BSc thesis in Railway Engineering

Estimation of Railway Track Parameters Using Evolutionary Algorithms

Terrence Dahoe 2021

TU Delft

Estimation of Railway Track Parameters Using Evolutionary Algorithms

by

Terrence Dahoe

to obtain the degree of Bachelor of Science in Civil Engineering
at the Delft University of Technology,
defended on Monday June 21, 2021.

Student number: 4856198
Project duration: April 26, 2021 – June 18, 2021
Thesis committee:

Dr. Alfredo Núñez Vicencio,	TU Delft, 1st supervisor
Prof. dr. Zili Li	TU Delft, 2nd supervisor
Dr. Chen Shen	TU Delft

TU Delft

Estimation of Railway Track Parameters Using Evolutionary Algorithms
by Terrence Dahoe

Abstract

Keywords: Evolutionary Computation, Railway Engineering, Inverse Modelling

The estimation of railway track parameters from a target frequency response function is a non-convex optimization problem. The objective of this thesis is to evaluate new evolutionary optimization methods to solve the railway track optimization problem. We use a Python based platform, since this programming language is increasing in popularity and new evolutionary algorithms are being made available there. The purpose of the research is to find out which optimizer and objective function performs best. Before the railway track parameters are optimized, tests are conducted on benchmark problems to become familiar with the evolutionary optimization solutions. In this thesis we use Grey Wolf Optimization, Particle Swarm Optimization and Genetic Algorithm. Subsequently, the optimizers are applied on a numerical railway track model from [1]. We focus on the optimization of four parameters, stiffness and damping of both railpad and ballast. Objective function one J_1 includes the sum of the differences between estimated frequency response function and target frequency response function in the frequency range 0-12500 Hz. Objective functions two includes the sum of the differences between estimated frequency response function and target frequency response function in the frequency range 0-3418 Hz and gives extra weight to those differences in the frequency range of the track resonances. Objective function three J_3 includes the sum of differences of the estimated track resonances and the resonances from the target response. We compare the performance of these optimizers and three different objective functions. We show that it is possible to use all three optimizers to estimate the rail track parameters. We yield the best results with an objective function that only takes the frequency range 0-3418 Hz into account and applies a higher weight factor to differences within the frequency ranges of the track resonances. The compared algorithms behave differently for different objective functions. In most of the tested cases, GWO performed the best. The smallest difference between target parameters and optimized parameters were obtained with objective function two J_2. Damping of the railpad is the most difficult parameter to estimate. Stiffness of the ballast was estimated with an error of about 3.90%, stiffness of the railpad with 1,06% and damping of ballast with an error of 3.60%. Finally, part of the further research includes the analysis of other evolutionary computation algorithm, optimization of the whole track model, sensitivity of analysis of the optimization parameters, inclusion of real-life measurements and addressing stochasticities in the objective function.

Preface

This report is written for the purpose of obtaining the degree of Bachelor of Science in Civil Engineering at the Delft University of Technology.

The research topic was found together with my supervisor, Dr. Alfredo Núñez Vicencio. In this thesis I found the opportunity to apply computational techniques learned during the minor Computation Science and Engineering. Besides the opportunity is taken to explore the field of evolutionary computing in the thesis.

I would like to thank Dr. Alfredo Núñez Vicencio for the excellent guidance and support during this process. My thanks goes to ir. Chen Shen for making his numerical rail track model available and the support implementing it. Lastly I would like to thank ir. Kees Lemmens for the practical sessions in the course Scientific Computing. The learned skills turned out to be extremely helpful in this thesis. Especially the skill to link different programming languages formed the backbone of this thesis.

Terrence Dahoe
Delft, June 2021

Table of Contents

1 Introduction

To predict the behaviour of the railway track we use a physics based model. The computational model we use in this thesis is a simplified representation of the physical track. In this thesis we use the dependence of the dynamic response of the track on the railway track parameters to estimate these parameters. The challenge here is to find an effective way to estimate the railway track parameters. We focus on the optimization of four parameters, stiffness and damping of both railpad and ballast. The dynamic response of the railway track is represented by a frequency response function, and we obtain it using the railway track model proposed in [1]. Drastic changes in frequencies can represent a deterioration mechanism occurring in the railway track. As such, the measurement and understanding of the frequency response function is key for railway maintenance decisions. The estimation of railway track parameters from a target frequency response function is a non-convex optimization problem. The objective of this thesis is to evaluate new evolutionary optimization methods to solve the railway track parameters optimization problem. The purpose of the research is to find out which optimizer and objective function performs best.

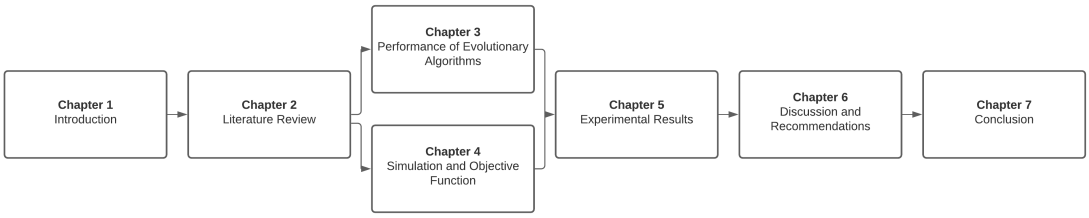


Figure 1: The structure of the thesis

The thesis starts after the introduction with a literature review, which contains a introduction to structural dynamics and the background of the computational railway track model. Besides, the literature review contains a description of the evolutionary algorithms: Grey Wolf Optimization, Particle Swarm Optimization and Genetic Algorithm. Section 3 is the next part of the thesis, which discusses the performance of the earlier introduced evolutionary algorithms. Section 4 consists of two parts. In the first part the experiment is discussed together and in the second part we discuss the three objective functions. Subsequently in Section 5 the experimental results are described. This section consists out of three subsections. Each part shows the results of an objective function. Finally in Section 6 the results are discussed and we do recommendations for further research. We end the thesis with a conclusion in Section 7.

2 Literature Review

As part of the thesis a literature study is conducted concerning dynamic features of structures, inverse modelling of railway tracks and evolutionary computing.

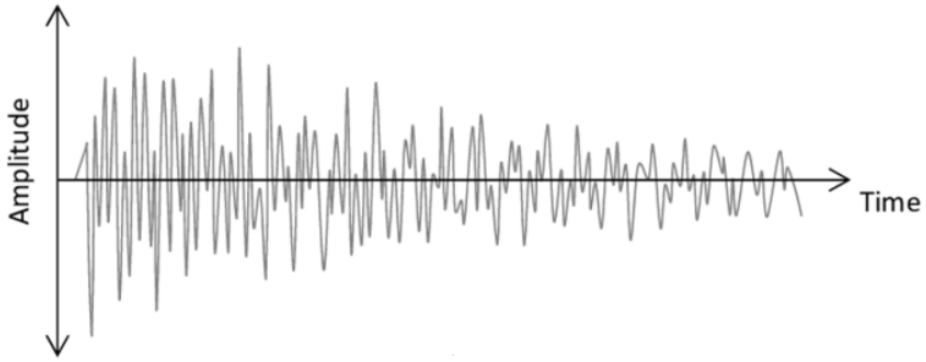
In 2.1 we discuss the relevant features of the dynamic response of structures. The identification of the railway track parameters from measurement is an inverse problem. The specific dynamic features serve as input for the inverse model to obtain these railway track parameters. The inverse model and the relation with the dynamic features is specified in 2.3.2. In the last part of the literature study 2.4 we discuss the methods to solve the inverse problem. There are iterative and non-iterative methods available. We discuss both methods, but focus on iterative evolutionary algorithms in particular.

2.1 Dynamic Response of Structures

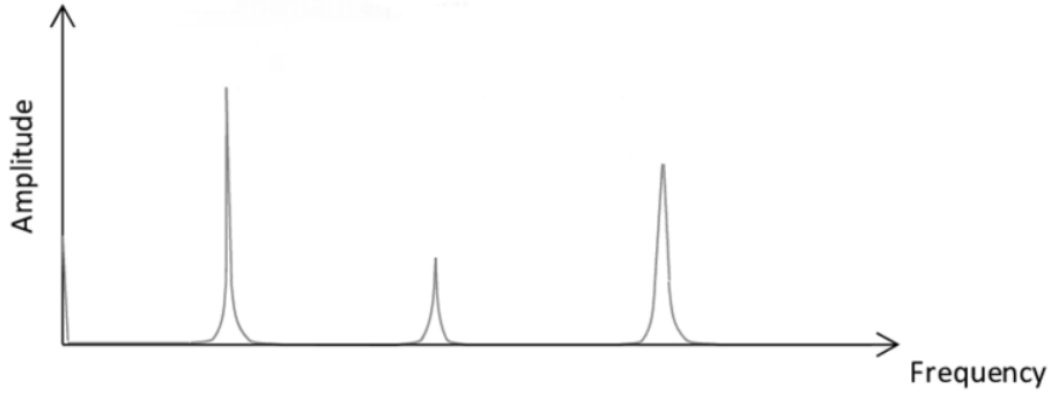
Structural dynamics is a part of structural analysis in which dynamic loads are taken into account. In general these are loads with an acceleration, in contrast to statics, where the loads do not move. An important part of structural dynamics is modal analysis. In modal analysis the structure is described in terms of dynamic characteristics as natural frequency, damping and modal shapes. The natural frequency is a frequency by which a structure vibrates when moved out of equilibrium. The modal shape is the deformation of a structure vibrating at natural frequency. A key concept in structural dynamics is that the amplitude varies under constant force with changing frequency. The amplitude amplifies when the frequency approaches the natural frequency of the structure. The response of a structure can be measured by placing accelerometers on the structure. In order to understand what a Frequency Response Function (FRF) is, we need to distinct between the time domain and frequency domain. Both domains show the dynamic response of a structure. The time domain is the most intuitive method in which the amplitude is a function of time. It is possible with the Fast Fourier Transform to convert a signal from the time domain to frequency domain. In Eq. 1 the transfer function is shown. This equation is used to model the dynamic response and is obtained from the time domain.

$$\mathrm{FRF}(f) = \frac{1}{(2\pi f)^2}\frac{S_{aF}(f)}{S_{FF}(f)} = \frac{1}{(2\pi f)^2}\frac{\sum_{n=1}^{N}\sum_{m=1}^{N-m-1} a[m+n]F[m]e^{-i(2\pi f)n}}{\sum_{n=1}^{N}\sum_{m=1}^{N-m-1} F[m+n]F[m]e^{-i(2\pi f)n}} \tag{1}$$

where S_{aF} is the cross-spectrum between the force and the acceleration, and S_{FF} the auto spectrum of the force, N the number of data points sampled [2]. The frequency domain shows the dynamic response, where the amplitude is a function of the oscillation frequency. Fig. 2 shows the the dynamic response of a structure in the time domain in Fig. 2a and the frequency domain in Fig. 2b.



(a) Time Domain



(b) Frequency Domain

Figure 2: Two possible representations of the dynamic response of a structure. Fig. 2a shows the dynamic response in the time domain. Fig. 2b shows the dynamic response in the frequency domain. The conversion between both domains can be done with the Fast Fourier Transform [3]

From the FRF we obtain information about the resonances, damping and mode shapes. The peaks in Figure 2b correspond with the natural frequencies of the structure. The amount of damping has a direct influence on the width of the peaks. The more damping, the wider the peak. Most signals in reality do not consist out of one a single frequency. These signals are a sum of sinusoidals with different frequencies. From the frequency response function in the frequency domain we can see out of which frequencies the signal consists. In practice, different methods are possible to identify resonance frequencies and dynamic responses. In this thesis, we consider the use of hammer impact excitation to obtain the FRF in a railway track. We describe next the formulation of FRF and some examples [4].

The FRF is a complex function and therefore consists of a real and an imaginary part. The FRF for frequency f is given by $\mathrm{FRF}(f) = a(f) + ib(f)$. The amplitude and phase are calculated with Eq. 2 and Eq. 3.

$$|\,\mathrm{FRF}(f)| = \sqrt{a(f)^2 + b(f)^2} \tag{2}$$

$$\angle\,\mathrm{FRF}(f) = \tan^{-1}\left(\frac{a(f)}{b(f)}\right) \tag{3}$$

with the amplitude we can determine the frequencies of the track resonances. With the phase we can determine whether the response of the structure is in phase with the excitation. To determine the mode shapes the amplitude as well the phase need to be taken into consideration. In this thesis we focus primarily on the amplitude of the FRF to estimate the railway track parameters.

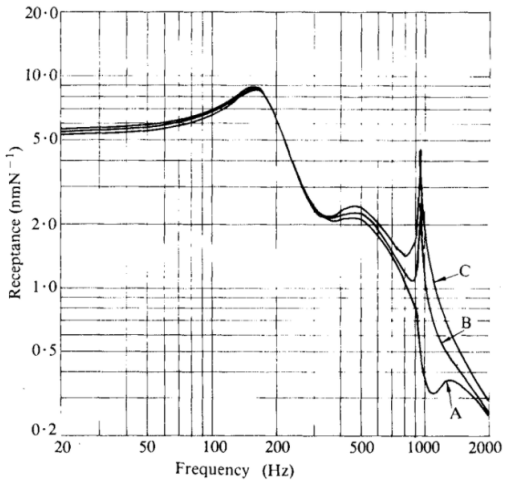


Figure 3: The magnitude plotted against the frequency [5]

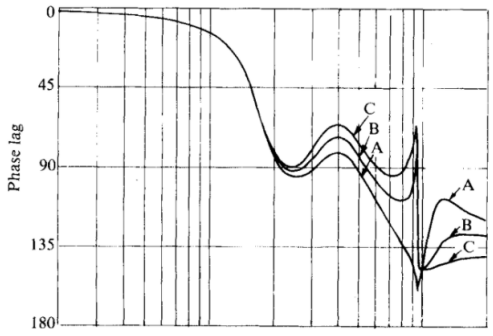


Figure 4: The phase plotted against the frequency [5]

Fig. 3 and Fig. 4 show three different FRF's. The FRF indicated as "A" represents a track response at the sleeper, "B" at a quarter distance span from sleeper, and "C" at mid-span (in between of two sleepers). Thus, the track response depends on the location of measurement.

2.2 Structural Health Monitoring of Railway Infrastructure

As a result of usage and environmental influences railway infrastructures are subject to wear. Failure of the rail could lead to catastrophic accidents. To prevent failure, the rail track needs to be maintained. An important question is when to perform maintenance and where. Deterioration is a process which is self accelerating in time. We need to prevent that the deterioration crosses the threshold where radical measures, such as replacement of the rails, are necessary to guarantee the safety. We increase the durability of the infrastructure by applying early state maintenance [6]. To apply early state maintenance we need adequate measuring techniques, which are the cornerstones of Structural Health Monitoring. In this section we discuss a technique by which we can examine the deterioration state of the rail track, the hammering test. We use the theory on Dynamic Characteristics from Section 2.1 to understand the physical principles behind the hammer test. With this background we discuss which data is obtained from the hammer test and how deterioration can be detected with this data.

2.2.1 Hammer Testing

Hammer testing is a non-destructive technique to perform modal analysis. We learned in section 2.1 that the objective in modal analysis is to describe the structure in terms of dynamic characteristics as natural frequency, damping and modal shapes. The principle behind hammer testing is that an impulse is applied to the structure. With this impulse we supply energy to the system and excite the response frequencies of the structure. The response is measured by the acceleration meters that are placed on the structure. We present the results of the experiment in an FRF diagram. Hammer testing falls into the category of impact testing. Another form of impact testing is the shaker test, in which a shaker is used to excite the structure. In the context of railway infrastructures the hammer test is better suited, because with the hammer test it is possible to obtain a broad range of frequencies by changing the type of hammer.

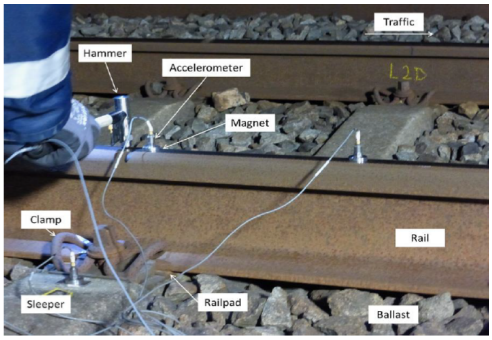


Figure 5: The test setup for the hammer test. The principle behind hammer testing is that an impulse is applied to the structure. With this impulse we supply energy to the system and excite the response frequencies of the structure. The response is measured by the acceleration meters that are placed on the structure. In this experiment 3 acceleration meters are used to measure the response of the track [6]

Fig. 5 shows a picture of a test setup for the hammer test [6]. It is important to take the effect of the hammer properties into account such as hardness of the tip and weight of the hammer. These properties determine which frequencies are excited in the structure. A hard hammer tip results in a short pulse and excites a broad frequency range, while a soft tip has a longer pulse and excites a frequency range that is more narrow. The weight property of the hammer has the

largest influence on the location of the frequency domain, rather than the size of the frequency domain. In [6] the hammer test is conducted with a large hammer (high mass) and a small hammer (low mass). This is done to have a broader frequency range. The small hammer has a better resolution for frequencies in the higher range, while the large hammer a better response in the lower range is obtained. Further, if the responses of interest are even lower, then a falling weight mechanism can be implemented, to excite those low frequency components. In this thesis, we will analyze frequency responses from 0-12500 Hz, with a focus on the responses between 0-3418 Hz. The analysis of track responses is interesting for understanding its behavior over time. Drastic changes in frequencies can represent a deterioration mechanism occurring in the railway track. As such, the measurement and understanding of the FRF's is key for railway maintenance decisions.

2.3 Modelling and Computational methods

To predict the behaviour of the rail track we use physics based models. In this section we focus on the dependence of the railway track parameters on the dynamic response of the track [1]. The computational model we use in this thesis is a simplified representation of the physical track. In traditional modelling we specify the track parameters and after simulation we analyse the FRF which represents the dynamic behavior of the track. In this section we also discuss inverse modelling in which we try to fit a computational model based on real observations. The challenge here is to find an effective way to approximate the railway track parameters. For the approximation of these parameters we use the FRF's described in the previous section.

2.3.1 Forward Modelling and Inverse Modelling

To analyse the dynamic response of the railway track we use a finite element model. The model is dependent on railway track parameters. The relevant parameters in this model are denoted as vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_8)^T$ in 8 dimensional space X. In Table 1 we find the railway track parameters of the rail track model. The input of the finite element model are the railway track parameters and the output an FRF, which we denote with $\mathbf{y}$. The finite element model of the rail track is denoted as f. Mathematically we can denote this forward relation as:

$$\boldsymbol{y} = f(\boldsymbol{x}), \boldsymbol{x} \in X \tag{4}$$

The main objective in [1] is to establish the inverse relation $g(\boldsymbol{y})$, based on observed an FRF. This relation can mathematically be defined as:

$$\boldsymbol{x} = g(\boldsymbol{y}), \boldsymbol{y} \in Y \tag{5}$$

It is important to note the difference between both relations. Eq. 4 is the forward relation between the railway track parameters and the FRF. In forward modelling the input parameters are known beforehand and we are simulating a process to collect the output data. Eq. 5 is the inverse relation between the FRF and the railway track parameters. In inverse modelling only the output data is known and we are adjusting the railway track parameters to fit the observations. The inverse modelling involves a fitting procedure to find unknown parameters. In this thesis, the procedure is conducted via a global optimization approach.

2.3.2 Global Sensitivity Analysis

The objective is to find the railway track parameters with the inverse relation in Eq. 5 by analysing the FRF features. The FRF's are obtained by performing field hammer tests [7]. The first step is to determine which FRF features are the most sensitive to changes in specific track parameters. To obtain these sensitive FRF features in relation to the railpad and ballast stiffness a global sensitivity analysis is performed. In a global sensitivity analysis the task is to quantify relative contribution of the input parameters in determining the output variable [8]. With the information from the global sensitivity analysis we need to find out which Track Resonances (TR) should be used to identify which railway track parameters.

Table 1: The railway track parameters that serve as input for the finite element rail track model. Table from [1]

Symbol	Track Parameter
x_1	Rail EI
x_2	Rail Intertia
x_3	Sleeper EI
x_4	Sleeper Inertia
x_5	Railpad stiffness
x_6	Railpad damping
x_7	Ballast Stiffness
x_8	Ballast Damping

Table 1 shows the railway track parameters that serve as input for the finite element rail track model.

To perform a sensitivity analysis, in [1] a sets of track parameters are evaluated. These parameters are taken randomly, uniformly distributed with a 5% increase of the nominal value as upper boundary and a 5% decrease of the nominal value as lower boundary. In Eq. 6 the railway track parameter x_i is denoted as an uniformly distributed stochastic variable with x_i^l as lower boundary and x_i^u as upper boundary.

$$x_i \sim U\left(x_i^u, x_i^l\right), i = 1, 2, \ldots, 8 \tag{6}$$

the next step is to generate N samples of the railway track parameters $\boldsymbol{x}^{(k)} \in X(k = 1, 2, \ldots, N)$. Since we presumed the railway track parameters x_i as stochastic variables, it can be assumed that each sample contains a different set of parameters. For each sample the dynamic response is simulated according the forward relation from Eq. 4.

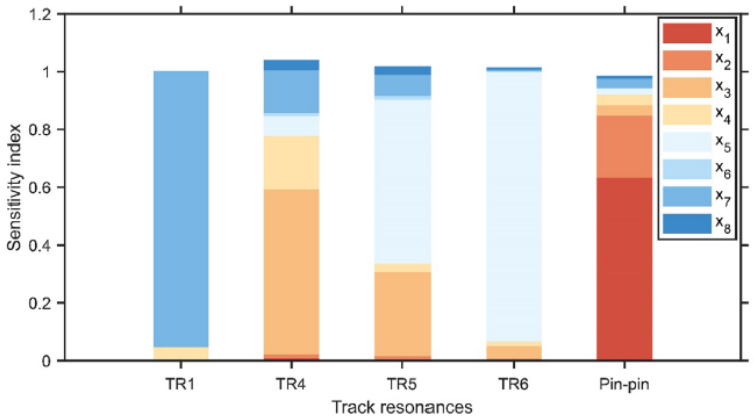


Figure 6: The figure shows in a bar plot the sensitivity of each track parameter per track resonance. From this figure we can see which railway track parameters have the largest influence on the track resonances [3]

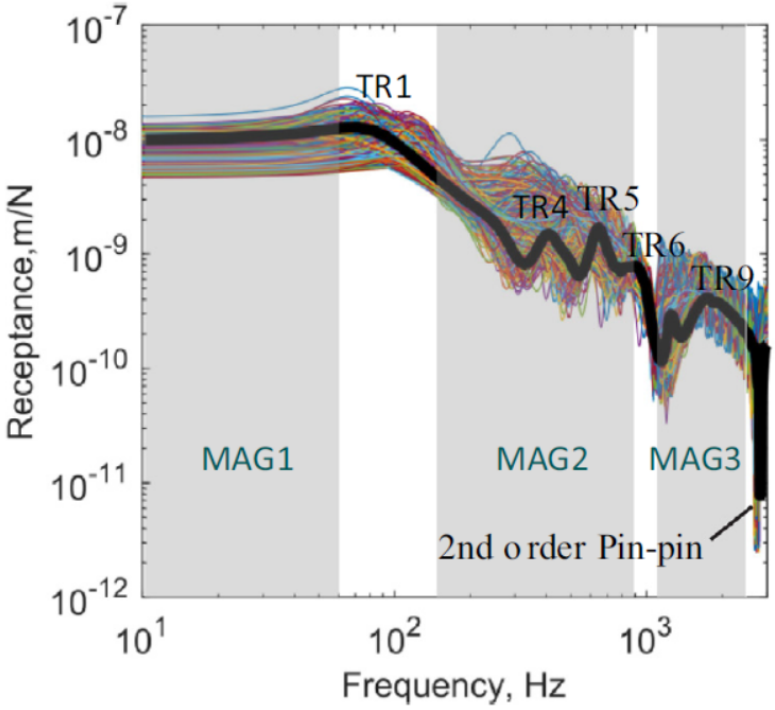


Figure 7: The dynamic responses of the rail track model [1]

As part of the sensitivity analysis, the FRF's of 5000 samples are plotted. The result is shown in Fig. 7. In the figure we see that each peak is labeled as a track resonance. The track resonances are chosen as the FRF features to analyze. Fig. 6 shows in a bar plot the sensitivity of each track parameter per track resonance. From the sensitivity analysis we can conclude that the pin-pin resonance is most sensitive to variation in the rail EI and the rail inertia, x_1 and x_2. We see that the track resonances TR4, TR5 and TR5 are most sensitive for a variation in the ballast stiffness x_7. The track resonances TR4, TR5, TR6 are the most correlated with the railpad stiffness x_5

2.4 Evolutionary Algorithms

In section 2.3 we discussed that the estimation of the railway track parameters from the measured FRF is an inverse problem. To obtain these railway track parameters the error between the measured FRF and the FRF of the simulated rail track model needs to be minimized. The search for the best railway track parameters has therefore turned into an optimization problem. We simply cannot try all the combinations, because that is computationally unfeasible. Therefore we need an optimization technique. The challenge here is to find optimization techniques that work on functions that are non-differentiable. To work around that problem we focus on the domain of evolutionary optimization techniques. These algorithms have in common that the mechanism behind these algorithms is inspired on nature. We see that the techniques we focus on in this section contain the same mechanisms found in Darwinian evolution, swarming behaviour and pack hunting. A potential bottleneck of evolutionary algorithms is that the fitness function needs to be evaluated quickly in order to work efficiently. This section contains the theoretical background of three famous evolutionary algorithms. In this section we have a closer look at Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO).

2.4.1 Nature-Inspired Metaheuristics

The classification of metaheuristics is complex and the categories are overlapping. In this section we focus on the classification of nature-inspired metaheuristics. Fig. 8 shows an overview of the categories within nature-inspired metaheuristics.

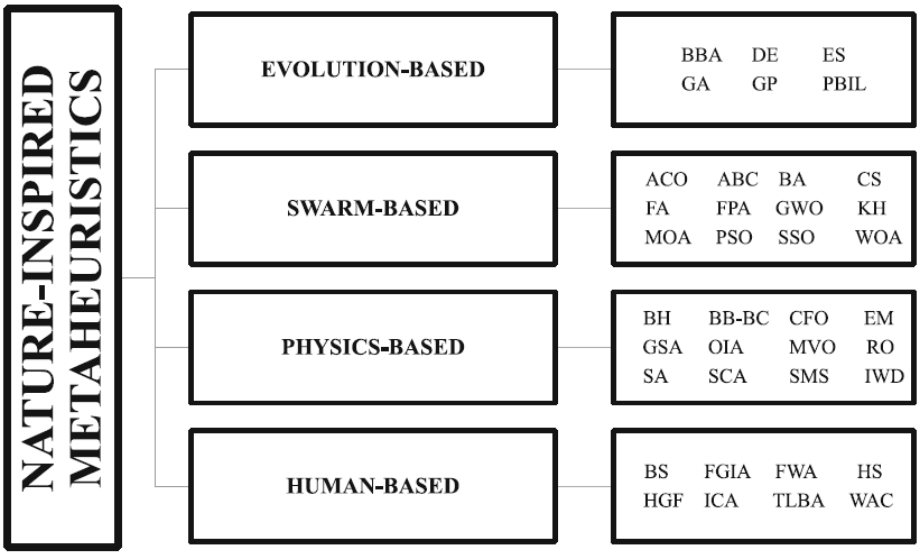


Figure 8: An overview of the categories within nature-inspired metaheuristics [9]

2.4.2 Evolution Based

Evolution-based methods are based on the Darwinian evolution theory. The general concept is that only the fittest individuals are able to 'reproduce' and form the population of the next iteration. These algorithms work with operators based on processes generally known in Darwinian evolution, such as crossover, mutation, and selection. Table 2 shows the most common evolution based metaheuristics.

Table 2: Overview of the most common evolution based metaheuristics

Algorithm	**Abbrevation**	**Citation**
Differential Evolution	DE	[10]
Evolutionary Strategies	ES	[11]
Genetic Algorithm	GA	[12]
Genetic Programming	GP	[13]
Population Based Incremental Learning	PBIL	[14]

2.4.3 Swarm Based

Swarm Based metaheuristics are based on the complex interaction between individuals that are observed in swarms. This thesis comprises partly of the comparison of two algorithms from this category: Particle Swarm Optimization and Grey Wolf Optimization. Table 3 shows the most common swarm based metaheuristics.

Table 3: Overview of the most common swarm based metaheuristics

Algorithm	**Abbrevation**	**Citation**
Ant Colony Optimization	ACO	[15]
Artificial Bee Colony	ABC	[16]
Bat Algorithm	BA	[17]
Cuckoo Search	CS	[18]
Firefly Algorithm	FA	[19]
Flower Pollination Algorithm	FPA	[20]
Grey Wolf Optimization	GWO	[21]
Krill Heard	KH	[22]
Magnetic Optimization Algorithm	MOA	[23]
Particle Swarm Optimization	PSO	[24]
Social Spider Optimization	SSO	[25]
Whale Optimization Algorithm	WOA	[26]

2.4.4 Physics Based

Physics based metaheuristics are techniques based on physical laws such as the laws of a black hole or the interaction between water drops. Sometimes in literature his category is subdivided in math based and chemical based meta heuristics. Table 4 shows the most common physics based metaheuristics.

Table 4: Overview of the most common swarm based metaheuristics

Algorithm	**Abbrevation**	**Citation**
Black Hole	BH	[27]
Big Bang-Big Crunch	BB-BC	[28]
Central Force Optimization	CFO	[29]
Expectation Maximization	EM	[30]
Gravitational Search Algorithm	GSA	[31]
Ortogonal Immune Algorithm	OIA	[32]
Multi-Verse Optimizer	MVO	[33]
Simulated Annealing	SA	[34]
Sine Cosine Algorithm	SCA	[35]
S Metric Selection	SMS	[36]
Intelligent Water Drops	IWD	[37]

2.4.5 Human Based

Human based meta heuristics are based on behavior that is typical for humans. Inspiration could be drawn from group interaction, but also from the psychological methodologies of problem solving. Table 5 shows the most common human based metaheuristics.

Table 5: Overview of the most common human based metaheuristics

Algorithm	**Abbrevation**	**Citation**
Bi-directional Search	BS	[38]
Football Game Inspired Algorithm	FGIA	[39]
Fireworks Algorithm	FWA	[40]
Harmony Search	HS	[41]
Human Group Formation	HGF	[42]
Imperialist Competitive Algorithm	ICA	[43]
Teaching and Learning Based Optimization	TLBA	[44]
Wisdom of Artificial Crowds	WAC	[45]

2.5 Genetic Algorithm

The idea of a Genetic Algorithm (GA) is based on the biological concept of evolution. Given a population, which could be vectors containing railway track parameters or individuals in the biological case. We randomly create a set of individuals, given an objective function to be maximised or minimized. Each individual has a fitness score which depends on the properties of the individual and the environmental selection pressure. Based on the outcome of this objective function applied on these individuals a fitness score is assigned. The individuals with the lowest fitness score tend to be eliminated from the population. The individuals with the highest fitness score will have the preference to 'reproduce' and pass their successful properties to the next iteration. The objective is to increase the fitness of the population each iteration. We see in Fig. 9 the simplified flowchart of evolutionary algorithms that explains in three steps the processes within an iteration [46].

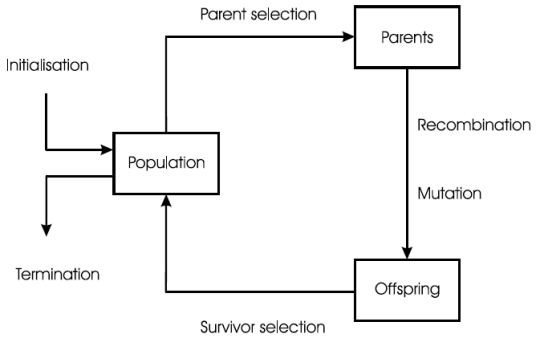


Figure 9: The simplified flowchart of a Genetic Algorithm explains in three steps the processes within an iteration. We start with a population of individuals. The objective function is applied to the individuals and a fitness score is computed. The individuals with the highest fitness score are selected as parent. Subsequently variation operators as recombination and mutation are applied to produce offspring. This offspring composes the new population and one iteration is completed [46]

To generate the new population variation operators are applied to the the parent solutions. The variation operators are recombination and mutation. Before we discuss what these operators do it is important to understand why they are necessary. The selected parents are the individuals in the population with the highest fitness score. These individuals have the properties that result in a high fitness score. The properties are called genes in accordance with the biological analogy. The objective is to find the individual with the best fitness score or with the best genes. It is unlikely that this individual is found in the first iterations, but we know which individuals contain the best properties. In order to find a better solution we combine the properties of the selected individual in the hope to obtain individuals which contain more desired properties and bring us closer to the best solution. This operation is called recombination, with the term breeding as biological equivalent. The operator needs at least two individuals and yields at least one individual. Fig. 10 show to the principle behind the recombination operator. The successful genes in the gene pool do not lead automatically to the best solution. There might be genes that are currently not in the gene pool, but would yield a better fitness score. Mutation is the mechanism behind the introduction of new genes. The mutation operator slightly alters a gene. The principle behind the operator is shown in Fig. 11.

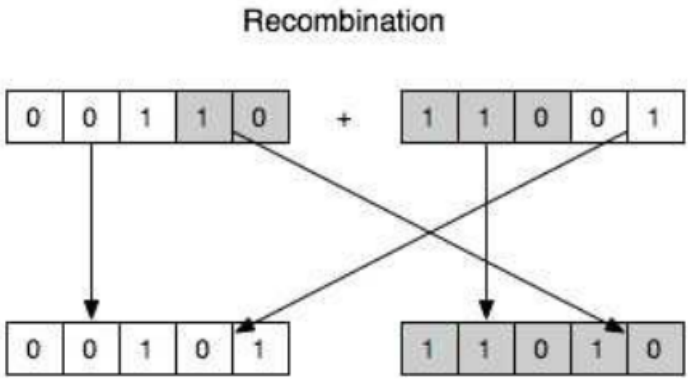


Figure 10: The principle behind the recombination operator. We see that the new individuals contain genes from both parents. The operator needs at least two individuals and yields at least one individual [47]

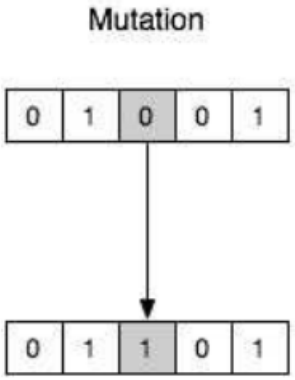


Figure 11: The principle behind the mutation operator. We see that the new individual contains an altered gene compared to the parent. The operator needs one individual and yields at least one individual [47]

2.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary algorithm that does not contain variant operators. The principle behind the algorithm is based on swarms in nature. It is important to note that the individual in a swarm is influenced by the rest of the swarm, but also has the freedom to move in its own direction. We iteratively simulate this behavior with a group of particles to find the global maximum of the objective function.

With PSO the particles move through the search space. This space is defined by the parameters of the problem. This means the number of parameters determines the dimension of the search space. The domain of the dimensions is determined by the possible values of the parameters. The concept of search space is explained with an example. Each possible solution is dependent on the x and y parameter. We obtain an area of possible x-y combinations, because each parameter can take a value independent from the other parameter. The limits of the search space are defined, because both parameters only can take a value in the interval [-3,3].

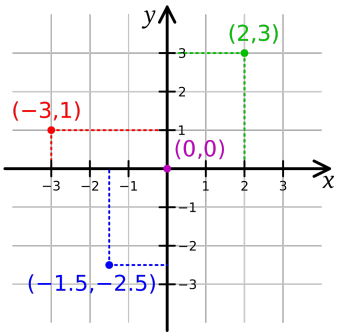


Figure 12: The figure shows four particles within the defined two dimensional search space.The limits of the search space are defined, because both parameters only can take a value in the interval [-3,3].The function needs to be dependent on x and y. We take the function $z = x + y$, which yields 5 for the green particle and -4 for the blue particle [48]

With PSO each particle represents a possible solution. The solution is dependent on the value of the parameters. These values are determined by the location of the particle in the search space. Fig 12 shows four particles within the above defined two dimensional search space. The function needs to be dependent on x and y. We take the function $z = x + y$, which yields 5 for the green particle and -4 for the blue particle.

We discuss first how to particles move though the search space before the algorithm is discussed. The movement of a particle is determined by the velocity vector which is influenced by different parameters. The velocity vector has a cognitive part which defines the best personal solution the particle has found so far and a social part which defines the best global solution that the swarm has found so far. Each particle has its own best personal solution and the best global solution of the swarm in memory. The memory of the particle is updated each iteration if necessary.

Each iteration a new velocity vector is calculated. The calculation of the velocity vector is done on basis of Eq. 7

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 \left(\hat{x}_i(t) - x_i(t)\right) + c_2 r_2 \left(g(t) - x_i(t)\right) \tag{7}$$

we define the following parameters:

- $v_i(t+1)$: Velocity vector of the next iteration
- ω : Inertial parameter which affects the propagation of the previous velocity vector
- c_1 : Weight coefficient for the best personal solution
- r_1 : Random number between 0 and 1
- $\hat{x}_i(t)$: The location of the best individual solution of the particle
- $x_i(t)$: The current location of the particle
- c_2 : Weight coefficient for the best global solution of the swarm
- r_2 : Random number between 0 and 1
- $g(t)$: The location of the best global solution of the swarm

The displacement is calculated with Eq. 8. Where $x_i(t+1)$ is the location of the particle in the next iteration.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{8}$$

in Fig. 13 we see a visual representation of the inertial, cognitive and social part in the calculation of the new particle location. The addition of these parts is visualized with the head-to-tail method [49].

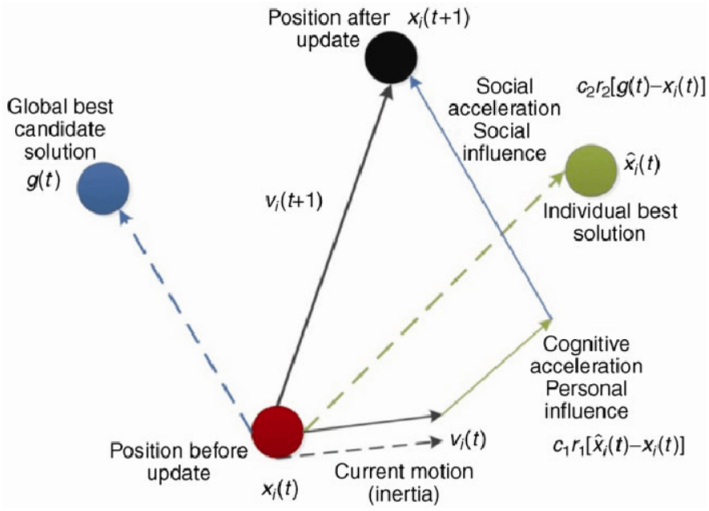


Figure 13: The visual representation of the inertial, cognitive and social part in the calculation of the new particle location. The addition of these parts is visualized with the head-to-tail method [49]

Before the iterative process the particles are initialized randomly from an uniform distribution. The fitness score or solution is calculated for each particle and saved in the memory. Subsequently the velocity vector is calculated and the particles are displaced. Again for each particle the fitness score is calculated. If a particle the obtains a new individual best the memory of that particle is updated. If the swarm obtains a new global best, the memory of all particles is updated. This process repeats itself until convergence or a predefined number of iterations.

2.7 Grey Wolf Algorithm

The Grey Wolf Algorithm (GWO) is an evolutionary algorithm inspired on the hunting pack behavior of wolves. GWO is an evolutionary algorithm that does not contain variant operators. The pack consists of an α-wolf, β-wolf and δ-wolf in descending order of dominance. The rest of the pack consists of ω-wolfs which follow the three dominant wolves. Hunting is the context of this algorithm means finding the best solution and the prey is the best solution within the search space. The algorithm consists out of two phases.

- Searching for prey
- Hunting the prey

The preys means in the context of the algorithm the best solution. We start with the random initialization of the wolfs. The wolfs are the agents that go though the search space looking for a prey. This is the first phase of the algorithm. Next, $\overrightarrow{\mathrm{A}}$, $\overrightarrow{\mathrm{C}}$ and $\overrightarrow{\mathrm{a}}$ are initialized according Eq. 9. The component of $\overrightarrow{\mathrm{a}}$ is linearly decreasing from 2 to 0 over the course of the iterations.

$$\begin{aligned} \vec{A} &= \vec{r}_1 \cdot 2 \cdot \vec{\mathrm{a}} - \vec{\mathrm{a}} \\ \vec{C} &= \vec{r}_2 \cdot 2 \end{aligned} \tag{9}$$

Subsequently the fitness score is evaluated for each wolf. This fitness score is dependent on the location of the wolf in the search space. The three best solutions are assigned to the α-wolf, β-wolf and the δ-wolf. It is assumed that the dominant wolves have superior knowledge about the location of the prey. The ω-wolves need to update their location according the location of the dominant wolves. The updating of the location is done according Eq. 10, Eq. 11 and Eq. 12. With $\overrightarrow{\mathrm{X}}_\alpha$, $\overrightarrow{\mathrm{X}}_\beta$ and $\overrightarrow{\mathrm{X}}_\delta$ denoting the position of the dominant wolves. The position of the wolfs in the next iteration is denoted with $\overrightarrow{\mathrm{X}}(t+1)$.

$$\begin{aligned} \overrightarrow{\mathrm{D}}_\alpha &= |\overrightarrow{\mathrm{X}}_\alpha \cdot \overrightarrow{\mathrm{C}}_1 - \overrightarrow{\mathrm{X}}| \\ \overrightarrow{\mathrm{D}}_\beta &= |\overrightarrow{\mathrm{X}}_\beta \cdot \overrightarrow{\mathrm{C}}_2 - \overrightarrow{\mathrm{X}}| \\ \overrightarrow{\mathrm{D}}_\delta &= |\overrightarrow{\mathrm{X}}_\delta \cdot \overrightarrow{\mathrm{C}}_3 - \overrightarrow{\mathrm{X}}| \end{aligned} \tag{10}$$

$$\begin{aligned} \overrightarrow{\mathrm{X}}_1 &= \overrightarrow{\mathrm{X}}_\alpha - (\overrightarrow{\mathrm{D}}_\alpha) \cdot \overrightarrow{\mathrm{A}}_1 \\ \overrightarrow{\mathrm{X}}_2 &= \overrightarrow{\mathrm{X}}_\beta - (\overrightarrow{\mathrm{D}}_\beta) \cdot \overrightarrow{\mathrm{A}}_2 \\ \overrightarrow{\mathrm{X}}_3 &= \overrightarrow{\mathrm{X}}_\delta - (\overrightarrow{\mathrm{D}}_\delta) \cdot \overrightarrow{\mathrm{A}}_3 \end{aligned} \tag{11}$$

$$\overrightarrow{\mathrm{X}}(\mathrm{t}+1) = \frac{(\overrightarrow{\mathrm{X}}_1 + \overrightarrow{\mathrm{X}}_2 + \overrightarrow{\mathrm{X}}_3)}{3} \tag{12}$$

It is possible that a solution is found by an ω-wolf that has a higher fitness score than the solutions in the current top 3. In that case that wolf attains dominance. The other ω-wolves update their locations according to the new dominant wolf/wolves. The transition from the searching phase to the hunting phase is mainly influenced with the parameter $\overrightarrow{\mathrm{a}}$. This value is linearly decreasing from 2 to 0 over the course of the iterations. The value of $\vec{A}$ is therefore between $[-2, 2]$. The next position of the wolf can be any position between the prey and the current position only if the value of $\vec{A}$ is between $[-1, 1]$. If $|\vec{A}| > 1$ the wolves are in the exploration phase and searching the search space diverging from the prey.If $|\vec{A}| < 1$ the wolves are in the exploitation and are converging toward the prey [50].

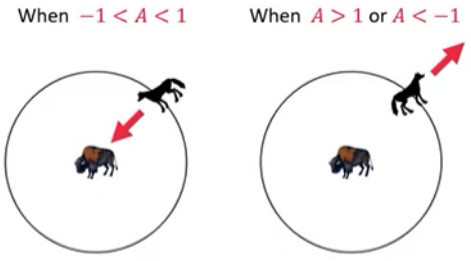


Figure 14: The figure shows that the searching and hunting behavior is dependent on the value of $\vec{A}$. If $|\vec{A}| > 1$ the wolves are in the exploration phase are diverging from the prey.If $|\vec{A}| < 1$ the wolves are in the exploitation and are converging toward the prey [51]

.

The threshold for transition between searching and hunting is shown in Fig. 14. The searching phase is also called exploration. It is important to have moved though the search space thoroughly to avoid ending up at the local maximum instead of the global maximum. The flowchart of the Grey Wolf Algorithm is shown in Fig. 15.

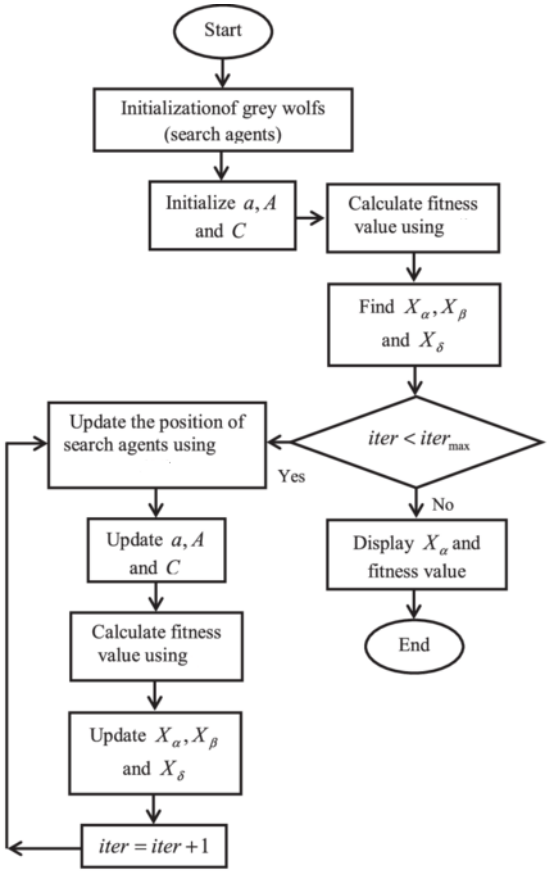


Figure 15: The flowchart of the Grey Wolf Algorithm [50]

3 Performance of Evolutionary Algorithms

As shown in Section 2.4 there are multiple evolutionary optimization techniques. The performance of these algorithms are dependent on different factors such as properties of the objective function and the tuning of the control parameters. Since the search agents are randomly initialized and the movement of the agents through the search space contains a random element. We have to take stochastic effects into account in order to draw a firm conclusion on the performance.

The performance of the evolutionary algorithms is tested by applying the evolutionary algorithms on test problems. Since we know beforehand the properties of the test functions such as the locations of the optima, number of optima and geometric shape we can draw a conclusion on the performance for different test functions. This information is necessary in the case of working with real engineering problems when these properties are not known beforehand.

In this section three experiments are performed. In the first experiment we want to find out what the percentage of convergence to the optimum solution for an increasing number of search agents. In the second experiment we study the rate of the convergence of each evolutionary algorithm for each test function. In the last experiment we want to find out what the computational costs are for each evolutionary algorithm.

The performance of the algorithms is also dependent on the control parameters. We can distinguish two groups of parameters. The first group contains the general parameters, which appear in all the optimizers. The parameters 'number of search agents' and 'number of iterations' belong to the first group. The second group is a collection of algorithm specific parameters. Examples are the weight coefficients for the cognitive behaviour c_1 and the social behaviour c_2, which are specific for Particle Swarm Optimazation described in Section 2.6.

Only the general parameters are modified while using the evolutionary algorithms. The algorithm specific parameters are kept constant on the default value. The modification of the specific parameters is considered outside the scope of this thesis. Table 6 shows the values of the parameters.

Table 6: The parameters of the evolutionary algorithms. The algorithm specific parameters are kept constant and default. The parameters 'Iterations' and 'Population size' are variable and modified during the performance tests

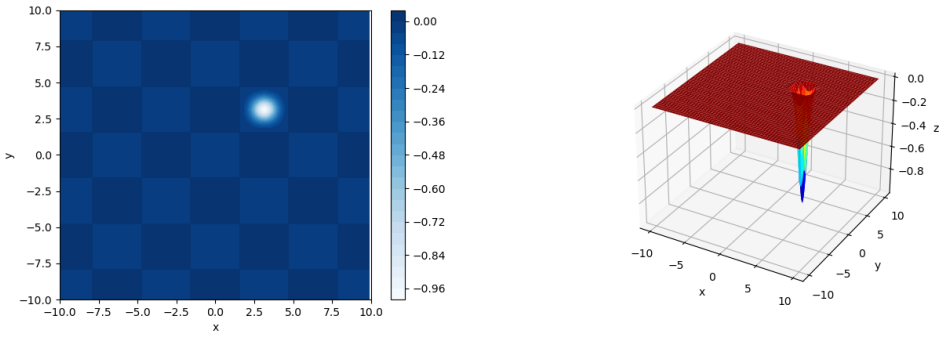
Name of algorithm	Parameters	Value
GWO	Iterations	Variable
	Population size	Variable
PSO	Iterations	Variable
	Population size	Variable
	ω_{max}	0.9
	ω_{min}	0.2
	c_1	2
	c_2	2
GA	Iterations	Variable
	Population size	Variable
	Crossover probability	1
	Mutation probability	0.01
	Keep	2

3.1 Test Functions

Test functions are used to study the performance of the evolutionary algorithms. We know beforehand the locations of the optima, the number of optima and the general geometric shape. Because the optimal solution is known, a check can be performed to test whether the evolutionary algorithm has converged or not. One reason to evaluate benchmarks is that those functions require few milliseconds to evaluate a solution. While the railway model can take up to various minutes when including complex characteristics. A simplified railway model can take various seconds. Therefore, by using these functions we get familiar with the optimized solutions. We select four functions with different modal properties. The degree of modality increases from uni-modal to multi-modal to highly multi-modal. The caveat with multi-modal functions is that the evolutionary algorithm gets stuck at a local optimum instead of the global optimum. The multi-modal functions are per definition non-convex. A function is convex if a line segment between any two points on the graph of the function lies above the graph between the two points.

3.1.1 Easom's Function - F1

Easom's function is an uni-modal function with the global optimum at $[\pi,\pi]$. In Fig. 16 the function is visualised by means of a 2D heatmap and 3D plot. Table 7 the characteristics of the function are schematized.



(a) Heatmap of Easom's Function

(b) 3D plot of Easom's Functions

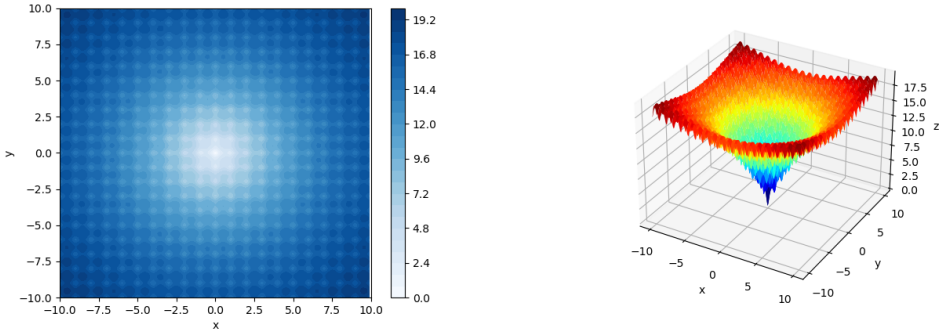
Figure 16: The figure shows a 2D heatmap and 3D plot of the Easom's Function. The Function is uni-modal and has a global optimum in $[\pi,\pi]$ and The global optimum has a small area compared to the search space

Table 7: Characteristics of the function

Name	Easom's function - F1
Domain	$[-10, 10]$,$[-10, 10]$
Optimal value(s)	$f(\pi, \pi) = -1$
Formula	$f(x, y) = -\cos(x)\cos(y)\exp\left(-\left((x-\pi)^2 + (y-\pi)^2\right)\right)$

3.1.2 Ackley's function - F2

Ackley's function is a multi-modal function with the global optimum at [0,0]. In Fig. 17 the function is visualised by means of a 2D heatmap and 3D plot. Table 8 the characteristics of the function are schematized.



(a) Heatmap of Ackley's function

(b) 3D plot of Ackley's function

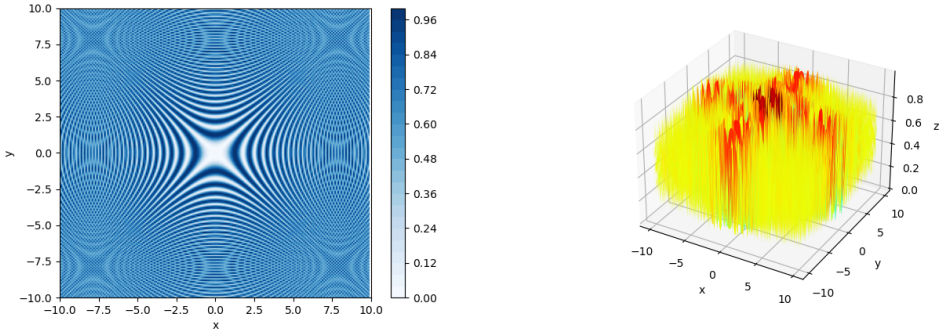
Figure 17: The figure shows a 2D heatmap and 3D plot of Ackley's function. The Function is multi-modal and has a global optimum in [0,0]

Table 8: Characteristics of the function

Name	Ackley's function - F2
Domain	[-10,10], [-10,10]
Optimal value(s)	$f(0,0) = 0$
Formula	$f(x,y) = -20\exp\left[-0.2\sqrt{0.5\left(x^2+y^2\right)}\right]$ $-\exp[0.5(\cos 2\pi x + \cos 2\pi y)] + e + 20$

3.1.3 Schaffer's function N.2 - F3

Schaffer 's function N.2 is a highly multi-modal function with the global optimum at [0,0]. In Fig. 18 the function is visualised by means of a 2D heatmap and 3D plot. Table 9 the characteristics of the function are schematized.



(a) Heatmap of Schaffer's function N.2

(b) 3D plot of Schaffer's function N.2

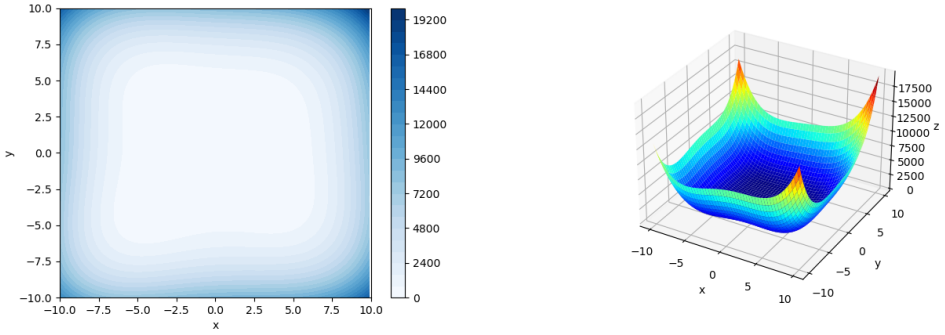
Figure 18: The figure shows a 2D heatmap and 3D plot of Schaffer function N.2. The Function is highly multi-modal and has a global optimum in [0,0]

Table 9: Characteristics of the function

Name	Schaffer's function N.2 - F3
Domain	[-10,10], [-10,10]
Optimal value(s)	$f(0,0) = 0$
Formula	$f(x,y) = 0.5 + \frac{\sin^2\left(x^2-y^2\right)-0.5}{[1+0.001(x^2+y^2)]^2}$

3.1.4 Himmelblau's function - F4

Himmelblau's function is a multi-modal function with the global optimum at four locations. In Fig. 19 the function is visualised by means of a 2D heatmap and 3D plot. Table 10 the characteristics of the function are schematized.



(a) Heatmap of Himmelblau's function (b) 3D plot of Himmelblau's

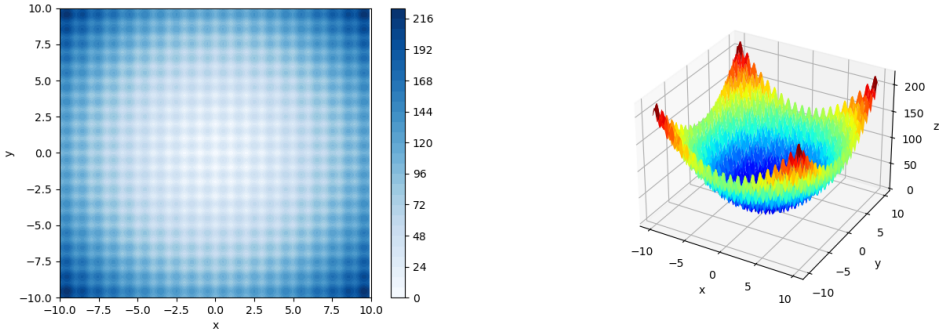
Figure 19: The figure shows a 2D heatmap and 3D plot of Himmelblau's function. The Function is multi-modal and has multiple modes

Table 10: Characteristics of the function

Name	Himmelblau's function - F4	
Domain	[-10,10], [-10,10]	
Optimal value(s)	$f(3.0, 2.0)$	$= 0.0$
	$f(-2.805118, 3.131312)$	$= 0.0$
	$f(-3.779310, -3.283186)$	$= 0.0$
	$f(3.584428, -1.848126)$	$= 0.0$
Formula	$f(x, y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right)^2$	

3.1.5 Rastrigin's function - F5

Rastrigin's function is a multi-modal function with the global optimum at four locations. In Fig. 20 the function is visualised by means of a 2D heatmap and 3D plot. Table 11 the characteristics of the function are schematized.



(a) Heatmap of Rastrigin's function

(b) 3D plot of Rastrigin's function

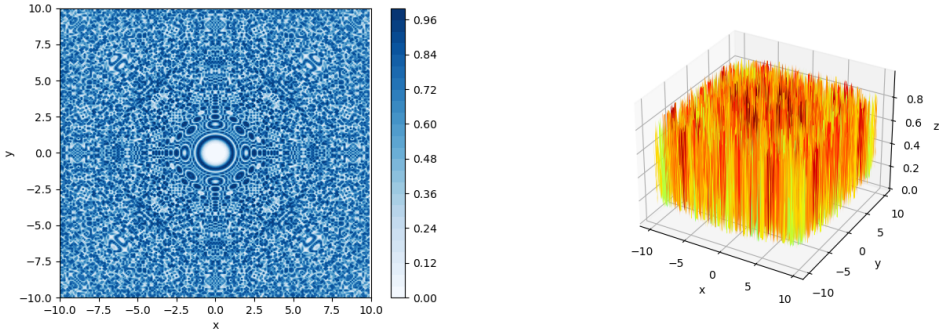
Figure 20: The figure shows a 2D heatmap and 3D plot of function F6. The function is multi-modal and has a global optimum in [0,0]

Table 11: Characteristics of the function

Name	Rastrigin's function - F6
Domain	[-10,10], [-10,10]
Optimal value(s)	$f(0,0) = 0$
Formula	$20 + \left(x^2 - 10\cos(2\pi x)\right) + \left(y^2 - 10\cos(2\pi y)\right)$

3.1.6 Altered Schaffer's Function - F6

The altered Schaffer's function is a highly multi-modal function with the global optimum at four locations. In Fig. 21 the function is visualised by means of a 2D heatmap and 3D plot. Table 12 the characteristics of the function are schematized.



(a) Heatmap of function F6

(b) 3D plot of function F6

Figure 21: The figure shows a 2D heatmap and 3D plot of function F6. The function is multi-modal and has a global optimum in [0,0]

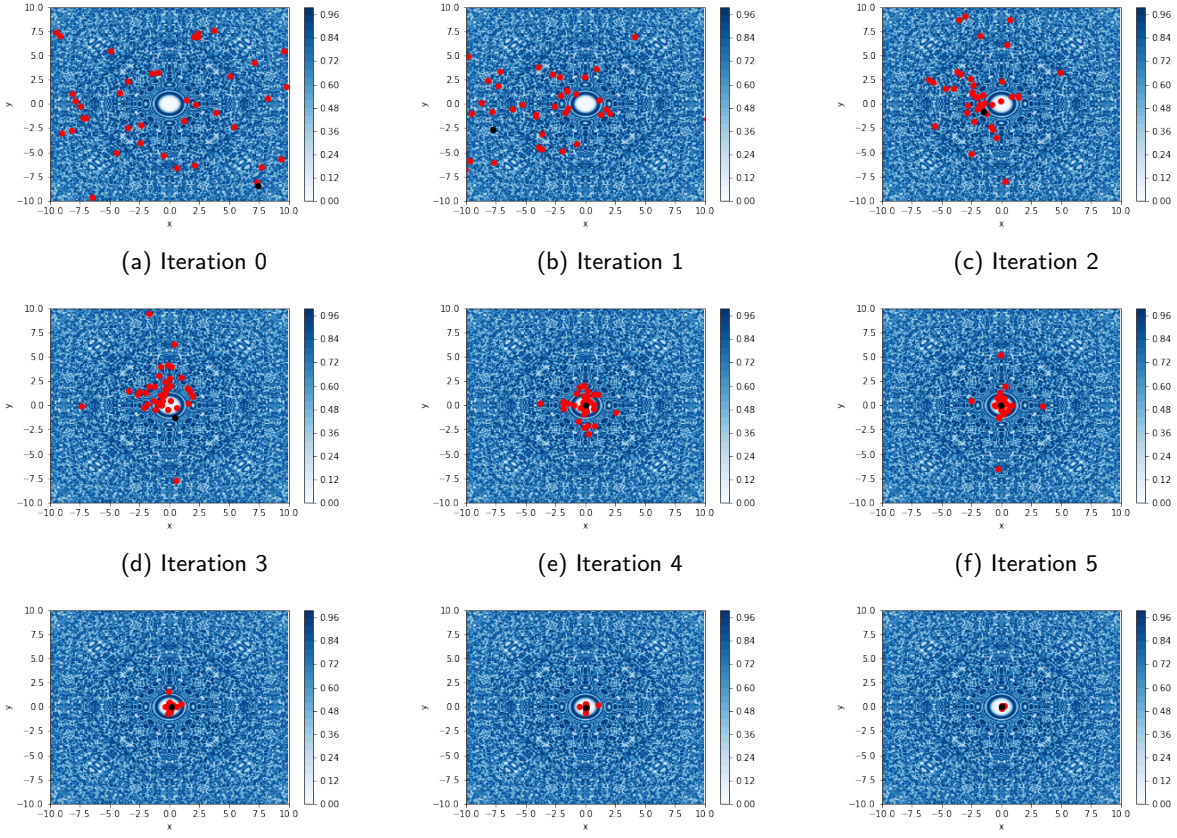
Table 12: Characteristics of the function

Name	Altered Schaffer's Function - F6
Domain	[-10,10], [-10,10]
Optimal value(s)	$f(0,0) = 0$
Formula	$0.5 + \frac{\sin^2\left(\left(x^2+y^2\right)^2\right)-0.5}{1+0.001(x^2+y^2)^2}$

3.2 Movement of the search agents

In this section we visualise the movement of the agents through the search space. Evolutionary computing in an iterative process and each iteration is visualised with a snapshot of the current state of the process. The only variable in this experiment is the optimizer. We consecutively conduct the experiment for GWO, PSO and GA. For all the optimizers we use 40 search agents. The optimizers are applied on the function Schaffer N.1. The GWO algorithm converges rather quickly. We show only the first 8 iteration to make proper visualisation of the visualisation possible. For PSO and GA 24 iterations are shown.

In Fig. 22, Fig. 24 and Fig. 26 the convergence process is simulated with GWO, PSO and GA applied on the function Schaffer N.1 with 40 search agents. To easily follow a search agent during the convergence process one search agent is colored black. In Fig. 23, Fig. 25 and Fig. 27 all the previous locations of the search agents are colored black. By doing so we create a map that indicates the part of the search space that is actually searched.



(a) Iteration 0 (b) Iteration 1 (c) Iteration 2

(d) Iteration 3 (e) Iteration 4 (f) Iteration 5

(g) Iteration 6 (h) Iteration 7 (i) Iteration 8

Figure 22: The convergence process simulated with GWO applied on the function Schaffer N.1 with 40 search agents

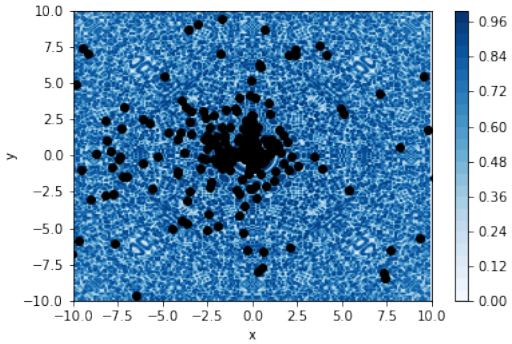
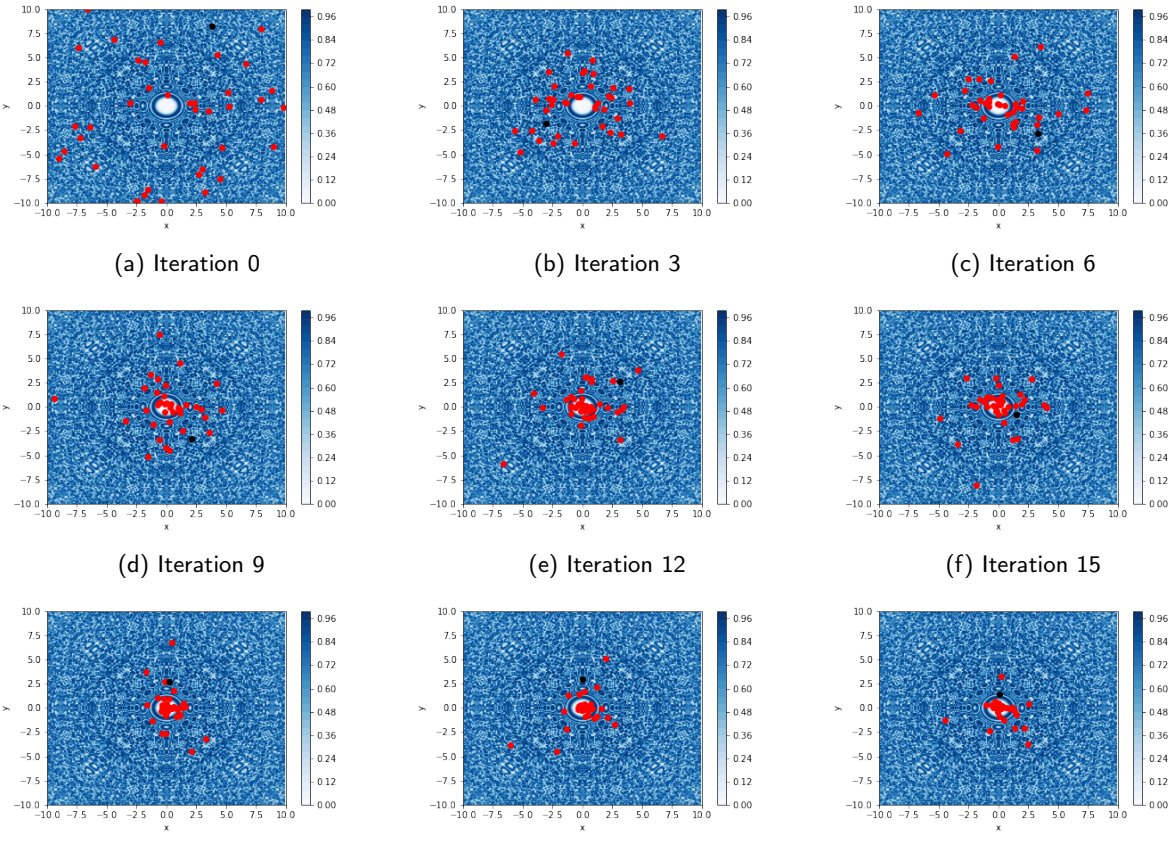


Figure 23: All solutions simulated with GWO applied on the function Schaffer N.1 with 40 search agents



(a) Iteration 0 (b) Iteration 3 (c) Iteration 6

(d) Iteration 9 (e) Iteration 12 (f) Iteration 15

(g) Iteration 18 (h) Iteration 21 (i) Iteration 24

Figure 24: All solutions simulated with PSO applied on the function Schaffer N.1 with 40 search agents

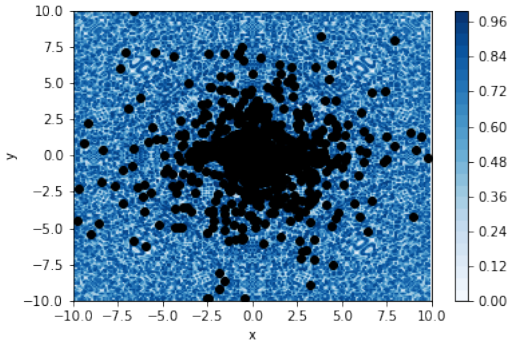
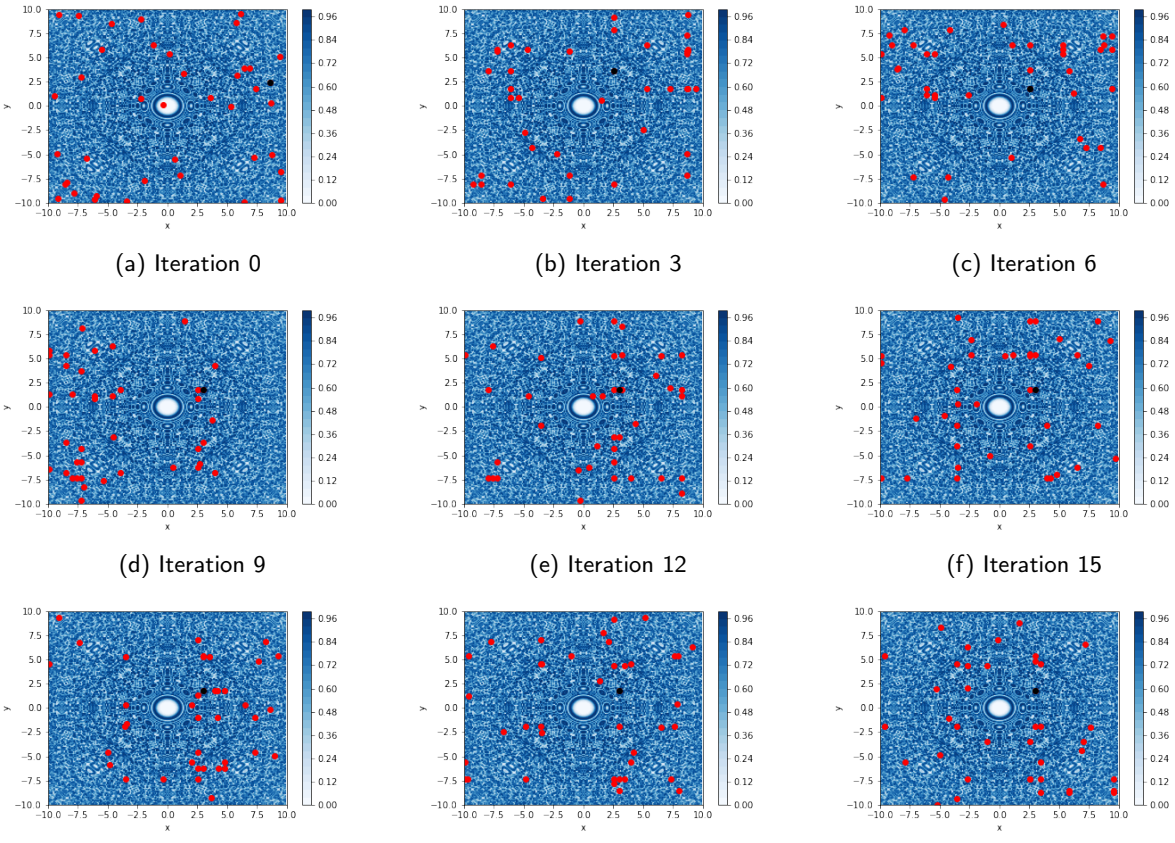


Figure 25: All solutions simulated with PSO applied on the function Schaffer N.1 with 40 search agents



(a) Iteration 0 (b) Iteration 3 (c) Iteration 6

(d) Iteration 9 (e) Iteration 12 (f) Iteration 15

(g) Iteration 18 (h) Iteration 21 (i) Iteration 24

Figure 26: All solutions simulated with GA applied on the function Schaffer N.1 with 40 search agents

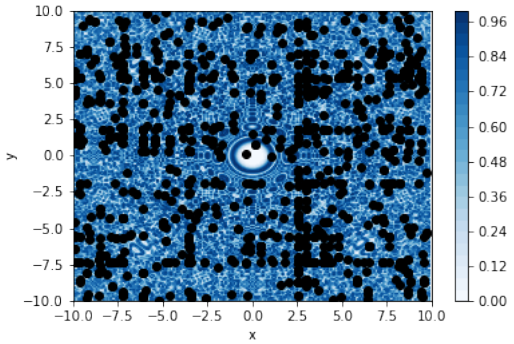


Figure 27: The convergence process simulated with GA applied on the function Schaffer N.1 with 40 search agents

3.3 Numerical comparison of the EA's

With the No Free Lunch Theorem it is proved that optimizer algorithms perform exactly the same, when averaged over all possible objective functions. For example when algorithm A outperforms algorithm B, then there must exist exactly as many other functions where algorithm B outperforms algorithm A [52]. This theorem implies that there is no best algorithm and that the performance of an algorithm is function dependent. In this section we compare the numerical results of the algorithms applied on the benchmark functions from Section 3.1.

Table 13: The numerical outcomes of the application of GWO on the benchmark functions

F	F *	Mean GWO	Best GWO	Worst GWO	STD GWO	MSE GWO
F1	−1	−0.9998818	−0.9999992	−0.9989307	0.0001224	0.0000000
F2	0	0.0000000	0.0000000	0.0000010	0.0000000	0.0000000
F3	0	0.0000273	0.0000000	0.0034934	0.0002723	0.0000001
F4	0	7.4511756	0.0000014	168.3136353	26.9980750	784.4160716
F5	0	0.9233187	0.0000000	30.4928559	3.7188431	14.6823117
F6	0	0.0028266	0.0000000	0.5498715	0.0350022	0.0012331

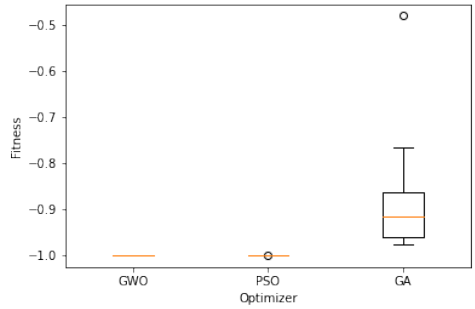
Table 14: The numerical outcomes of the application of PSO on the benchmark functions

F	F*	Mean PSO	Best PSO	Worst PSO	STD PSO	MSE PSO
F1	−1	−0.9999938	−1.0000000	−0.9998850	0.0000096	0.0000000
F2	0	0.0049219	0.0000875	0.0314262	0.0037114	0.0000380
F3	0	0.0000002	0.0000000	0.0000761	0.0000025	0.0000000
F4	0	0.0014365	0.0000001	0.4112068	0.0157075	0.0002488
F5	0	0.2021788	0.0000062	5.9562249	0.4457013	0.2395259
F6	0	0.0000000	0.0000000	0.0000011	0.0000001	0.0000000

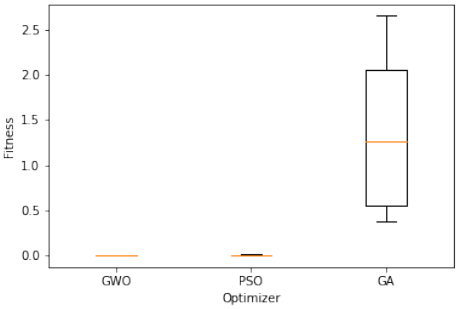
Table 15: The numerical outcomes of the application of GA on the benchmark functions

F	F*	Mean GA	Best GA	Worst GA	STD GA	MSE GA
F1	−1	−0.8651625	−0.9999726	−0.2605681	0.1280339	0.0345738
F2	0	1.0124667	0.0167889	3.6844148	0.8048388	1.6728542
F3	0	0.0021202	0.0000000	0.0161548	0.0028652	0.0000127
F4	0	0.7895123	0.0000413	7.0237208	0.8787829	1.3955891
F5	0	1.9194927	0.0105721	7.6715139	1.2890445	5.3460878
F6	0	0.0015269	0.0000000	0.0191067	0.0026996	0.0000096

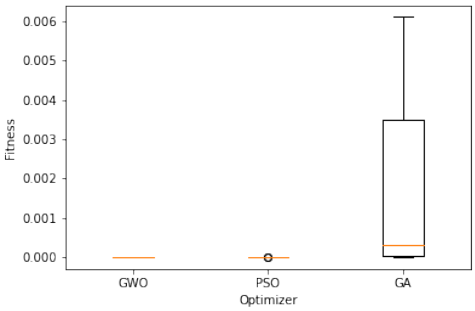
The experiment is conducted by applying the optimizers each a 1000 times on the specific benchmark function. The optimizers are applied with 40 search agents and 30 iterations. The best solution of each run is saved in an array. Subsequently statistic information is calculated on basis of that array. The results are shown in Table 13, 14, 15 and show the mean, best score, worst score, standard deviation and the mean squared error. The statistical information is visualised with boxplots in Fig. 28.



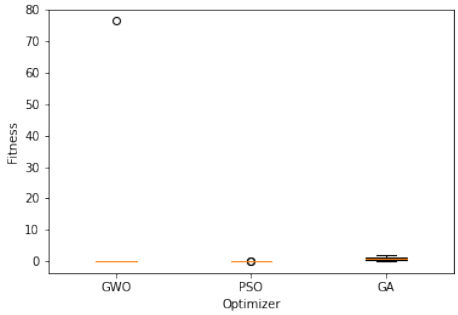
(a) F1 - Easom



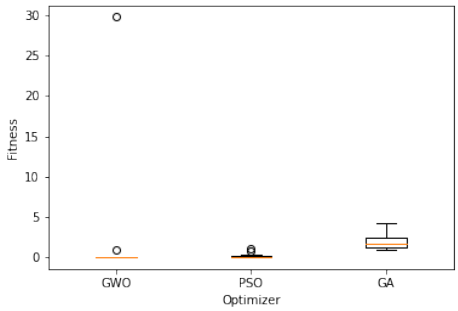
(b) F2 - Ackley



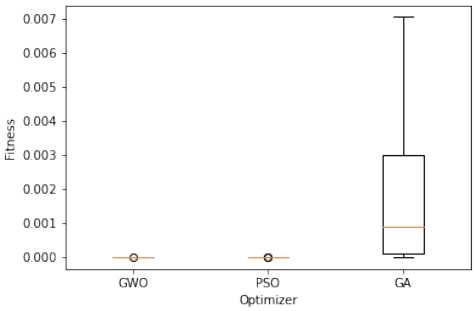
(c) F3 - Schaffer N.1



(d) F4 - Himmelblau



(e) F5 - Rastrigin



(f) F6 - Altered Schaffer

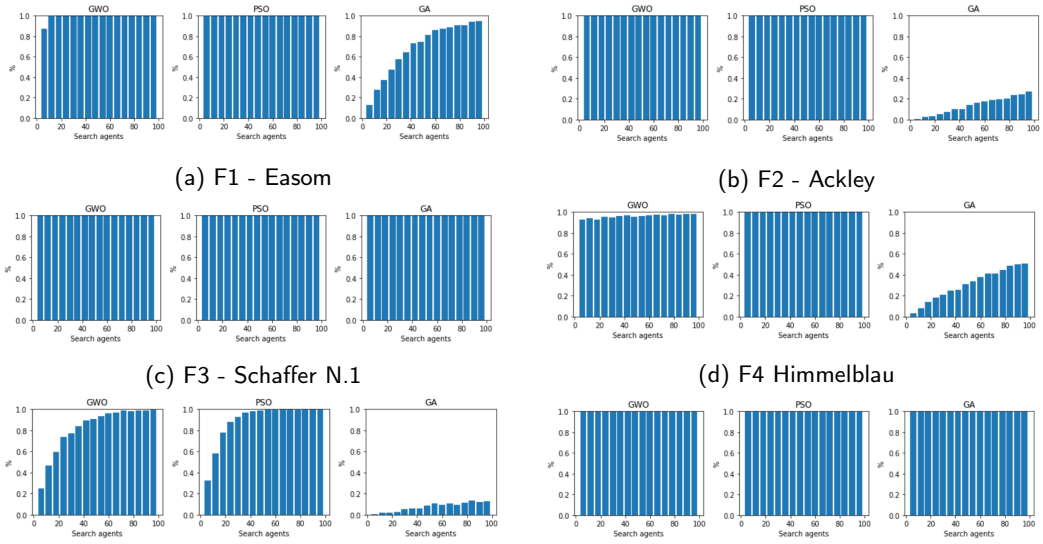
Figure 28: The statistical information from the array with the scores of 1000 runs is visualised with boxplots

3.4 Convergence and Convergence Rate

We conduct an experiment to study the convergence behavior of the EA's. This experiment consists of two parts. Each optimizer returns an array with the best score per iteration which shows the rate of convergence. In the first part we keep the number of iterations and search agents constant and we take the average convergence rate of 500 runs. Fig. 30 shows the rate of convergence for the benchmark functions.

In the second part of the experiment a test is performed to find out how the best solution changes with the number of search agents. From the theory we know that a larger part of the search space is searched with a larger number of search agents and the probability of convergence to the global optimum increases. An increment in the number of search agents increases the computational cost. Therefore it is important to gain insight on how the percentage of successful runs increases with the number of search agents.

If the difference between the best score and the global optimum is smaller than 0.1 we consider the run as successful. Fig. 29 shows the ratio between successful runs and total runs against the number of search agents. Again the ratio is averaged over 1000 runs to decrease stochastic influences. We see clearly that in some cases the number search agents influences the success rate. This effect is for all optimizers most prevalent in Fig. 29e.



(a) F1 - Easom

(b) F2 - Ackley

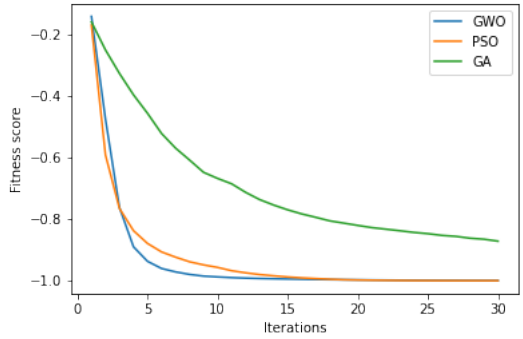
(c) F3 - Schaffer N.1

(d) F4 Himmelblau

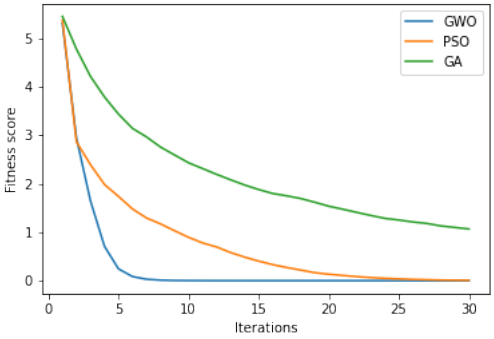
(e) F5 - Rastrigin

(f) F6 - Altered Schaffer

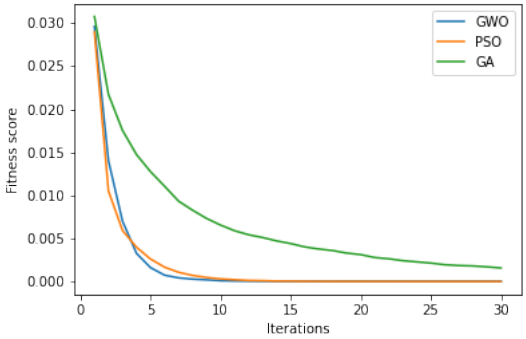
Figure 29: If the difference between the best score and the global optimum is smaller than 0.1 we consider the run as successful. The figure shows the ratio between successful runs and total runs against the number of search agents



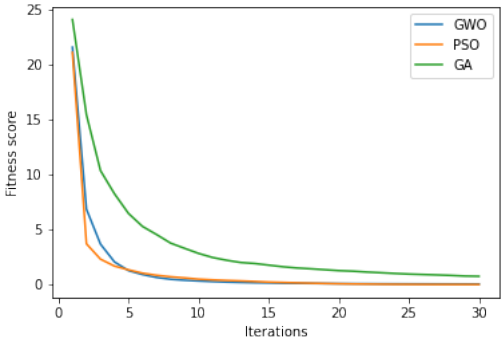
(a) F1 - Easom



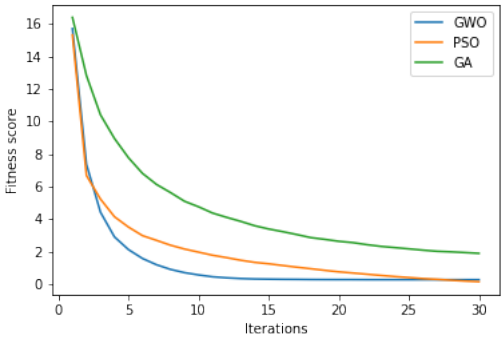
(b) F2 - Ackley



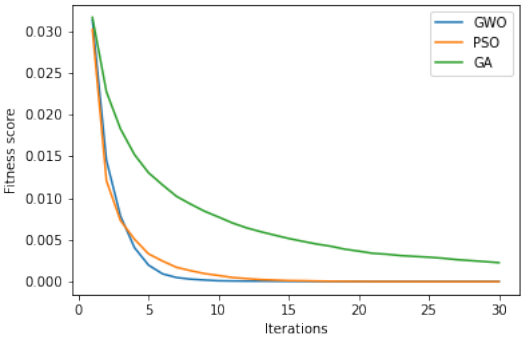
(c) F3 - Schaffer N.1



(d) F4 - Himmelblau



(e) F5 - Rastrigin

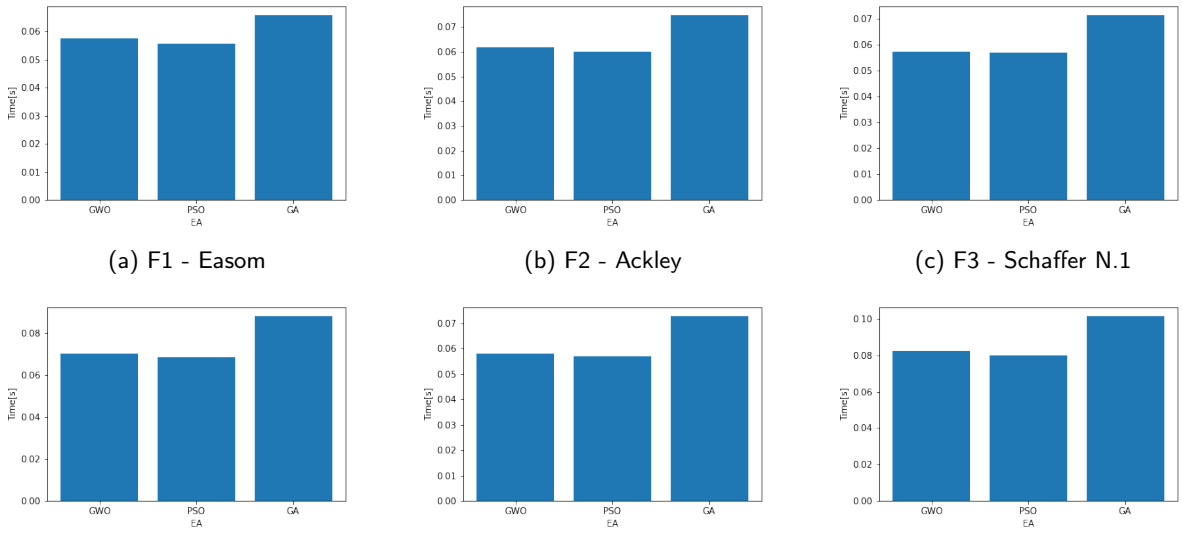


(f) F6 - Altered Schaffer

Figure 30: The rate of convergence for GWO, PSO and GA applied on the benchmark functions

3.5 Computational Costs

The computational cost of an evolutionary algorithm is an important performance metric. The computational cost is measured in the time it takes to execute the algorithm. In Section 3.4 we discussed that an computationally inexpensive EA that converges slowly may perform as well as an EA that is computationally expensive but converges fast. In the experiment the GWO, PSO and GA optimizers are applied on all test functions. Per test function we run the optimizer 500 times and take the average elapsed time for each EA. The results of the experiment are shown in Fig. 31.



(a) F1 - Easom (b) F2 - Ackley (c) F3 - Schaffer N.1

(d) F4 - Himmelblau (e) F5 - Rastrigin (f) F6 - Altered Schaffer N.1

Figure 31: The computational cost of each optimizer for the benchmark problems

What stands out is that the figures show strong resemblances. Prior to the experiment the expectation was that the ration in computation costs between the different algorithms would be the same for all benchmark functions, because the mathematical nature of each optimizer stays constant. This turns out to be true. It is unexpected that, besides ratio, also a strong resemblance occurs in the numerical values. A reason could be that the evaluation of the objective function has an insignificant effect on the computational costs.

4 Simulation and Objective Functions

We conduct experiments to gain insight in the performance of the optimizers to estimate the track parameters. In this section we discuss the experiments and what kind of results we extract from these experiments. We repeat the experiments with different objective functions. The objective functions used are J_1, J_2 and J_3. The objective functions are explained in detail in this section.

4.1 Experiments

The search space is determined by the railway track parameters. The position of the search agent within that search space represents a set of railway track parameters. The common factor of all the experiments is that for each set of railway track parameters an FRF is generated and compared with the target FRF. We compare these FRF's in different ways, which results in three objective functions. Before the objective functions are discussed in detail, it is important to first understand which data we obtain from the experiments. Within an experiment each optimizer is called 10 times. For each run we use 20 search agents and 6 iterations. For every optimizer a lower bound and upper bound is needed to define the search space. For the lower boundary $[1e9, 1e4, 1e7, 1e4]$ is used and $[10e9, 10e4, 10e7, 10e4]$ for the upper boundary. We used mostly the default parameters of the optimizers. A detailed sensitivity analysis can be conducted to tune the algorithms, which is a topic for further research. In the case of PSO, we tuned the maximum velocity to consider the different order of magnitudes in the search space. Alternatively, a normalization can be applied before the optimization, which performance can be analyzed.

4.1.1 Numerical Values and Boxplot

We run each algorithm a total number of 10 times. When the algorithm has completed a run, there is a best solution. For all the runs we store the fitness score of the best solution. After the experiment we have an array that contains the fitness score of the runs. As we have seen in Section 3 the search process involves stochastic elements. We expect that we obtain different fitness scores each run. We present the performance of the methods in different tables. The columns of a table include the mean fitness score, the best fitness score, the worst fitness score and the standard deviation. The numerical results are also visualised with a boxplot. With the boxplot we can easily compare the optimizers and see which optimizer has the best performance, for example the best fitness score. Besides the spread of the results could be observed.

4.1.2 Rate of Convergence

It is important to have insight into the rate of convergence of the optimizer. This information is crucial in choosing the number of iterations and in determining the cost of an optimizer. Within a run for each iteration the best score is stored in an array. Each experiment consists of 10 runs for each optimizer. We obtain therefore for each optimizer 10 convergence arrays. These arrays with the fitness scores of each iteration are plotted against the number of iterations.

4.1.3 Visualisation

In the experiment two methods are used to visualise the iterative process. The first method makes use of a parallel-data plot. This is a tool to visualise high-dimensional data. In the plot 4 vertical axes are found and each axis represents a railway track parameter. The set of railway

track parameters can be represented by a line that intersects these 4 axes on specific locations dependent on the value of that parameter. The lines of the search agents are red coloured and the target is coloured blue. When the algorithm is applied we can plot a parallel-data plot for each iteration. With the consecutive plots we can see what the behaviour of the optimizer is and whether the process converges or not. Another method of visualisation is to plot all the FRF's of a specific iteration in the same figure together with the target FRF. In this way we visualise the search space in terms of the FRF.

4.2 Objective Functions

Objective functions are used to assign a fitness score to the search agents. The goal of the optimizers is to obtain the best fitness score. In this section the 3 objective functions are described that we test in the experiments. Each optimizer has the same main parts. There is a part for agent initialization, where each search agent is put randomly in the search space. There is also a part where the fitness score of the search agents is evaluated. This part consists of a nested loop. The outer loop goes through the iterations and the inner loop goes through the number of search agents. This is where the objective function is called to evaluate the fitness score.

4.2.1 Objective Function J_1

Objective function J_1 is the most simple one. Here the target FRF is needed as input, where $\overline{FRF}(f_i)$ is the value of the target at frequency f_i. The FRF of the target is included in the objective function. The next step is that for a search agent the FRF is simulated. The error is calculated by summing the absolute differences between the target and the resulting FRF from the agent in the frequency range between 0 - 12500 Hz. The sum of absolute differences is the fitness score. In Fig. 32 the steps of the objective function are visualised in a flowchart. In Eq. 13, x^k is the search agent k, and $FRF\left(x^k, f_i\right)$ is the simulated frequency response with search agent k at frequency f_i. Important to note that in the simulation we consider 12801 data points that are in the range of 0 - 12500 Hz.

$$J_1\left(x^k\right) = \sum_{f_i=1}^{12801} \mid FRF\left(x^k, f_i\right) - \overline{FRF}(f_i) \mid \tag{13}$$

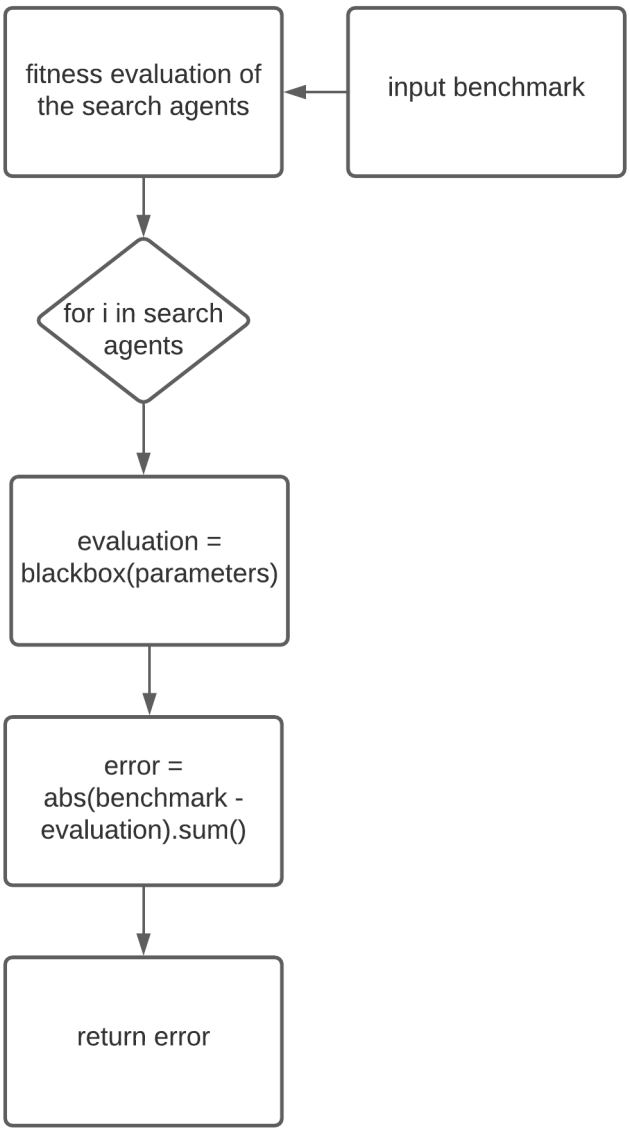


Figure 32: The flowchart of J_1

4.2.2 Objective Function J_2

Objective function J_2 is a more sophisticated version of J_1. From the theory we know that the more valuable information to estimate the railway track parameters is in the frequency range 0 - 3418 Hz. Besides there are frequency ranges of the track resonances which contain the most valuable information. With this objective function we only look in the frequency range of 0 - 3418 Hz and if the iteration variable of the inner loop is within the frequency range of a track resonance, the error is multiplied by 100. Before applying this objective function there needs to be a visual inspection of the target to estimate the ranges of the track resonances. For the experiments the following ranges are used $[20, 120], [350, 600], [600, 900], [900, 1250], [2000, 2700]$. Each range has an upper and lower bound of the estimated frequency range of a particular track resonance. In Fig. 33 the flowchart of the objective function is shown. In Eq. 14 we define J_2. With the symbol k defining the search agents.

$$J_2\left(x^k\right) = \sum_{f_i=1}^{3500} w(f_i) \cdot \mid FRF\left(x^k, f_i\right) - \overline{FRF}(f_i) \mid \tag{14}$$

$$w(f_i) = \begin{cases} 100 & \text{if } f_i \in [20, 120] \cup [350, 600] \cup [600, 900] \cup [900, 1250] \cup [2000, 2700] \\ 1 & \text{otherwise} \end{cases}$$

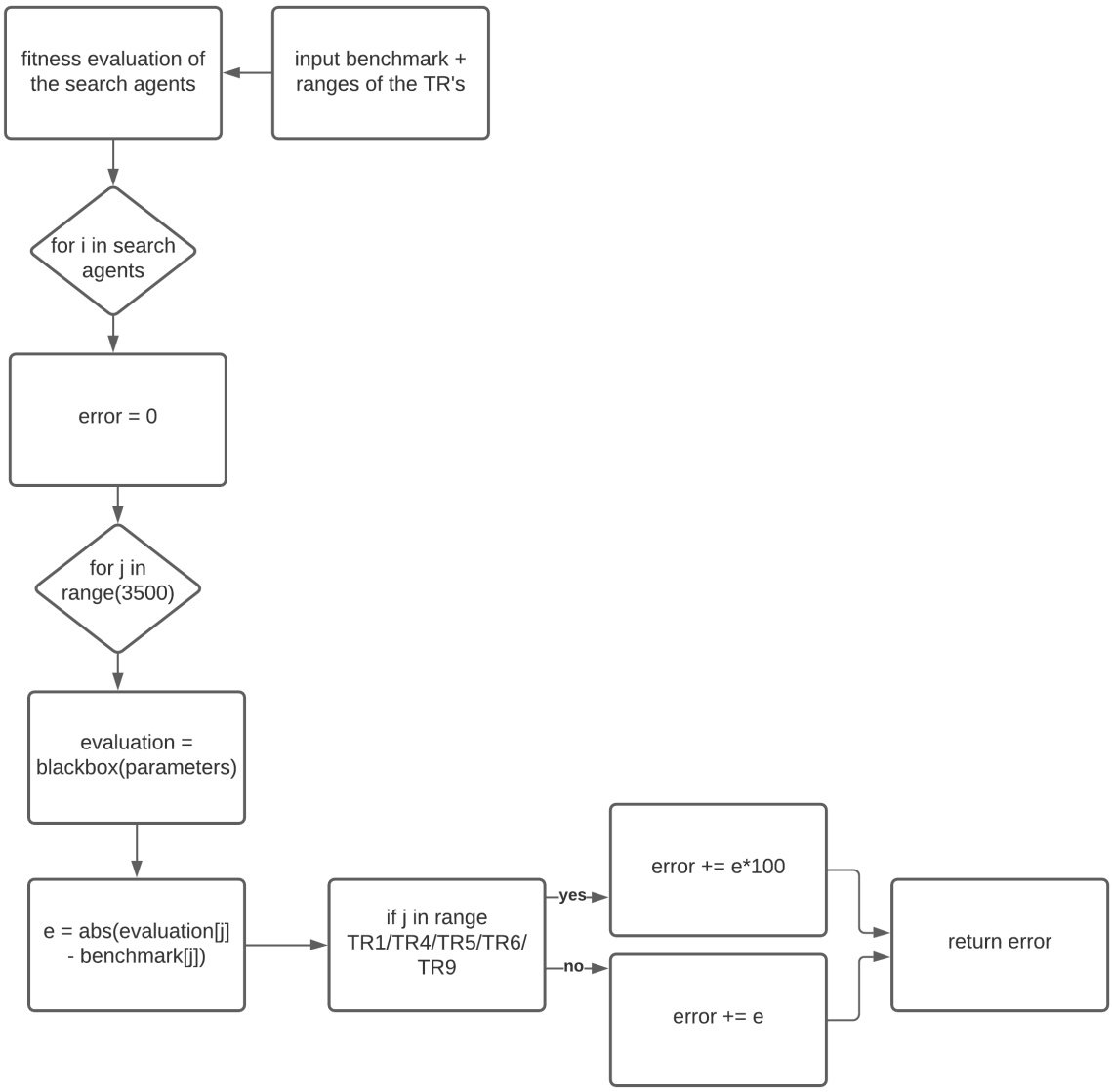


Figure 33: The flowchart of J_2

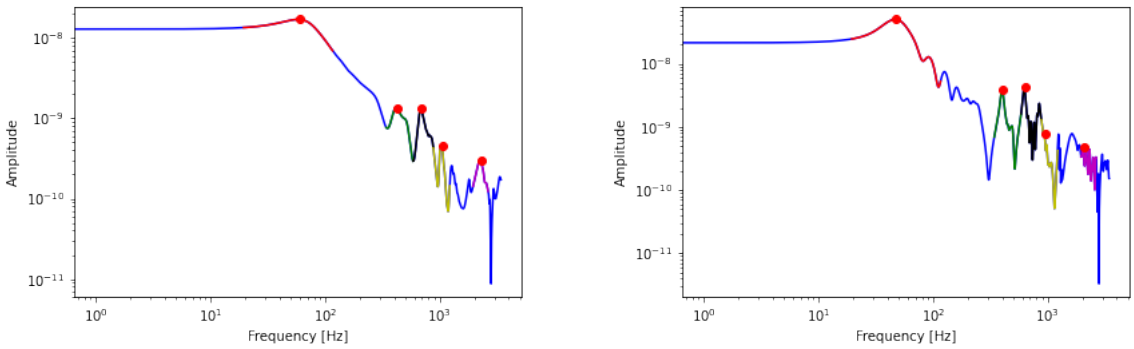
4.2.3 Objective Function J_3

Objective function J_3 also considers that the first 3500 data points contain more value information. In addition we also need a estimation of the TR ranges as input. The key difference is that the fitness is determined by the x-values of the resonance peaks rather than the y-values. The goal of the objective function is to minimize the x-distance of the resonance peaks. In Fig. 35 we see two different FRF's. The objective function is able to find the peaks within the specified range. It is assumed that the peaks of the evaluation are within the same frequency range as the target. The objective function find the indices of the resonance peaks. The fitness is obtained by summing the absolute differences of the index arrays. In Fig. 35 the flowchart of the objective function is shown. In Eq. 15 we define J_3. The variable TR_i^k is the frequency of track resonance i in the FRF, obtained with agent k. The variable $\overline{TR}_i$ is the frequency of track resonance i of the target FRF.

For finding the track resonance of the agents, we perform a local search. In this work, we consider the following ranges: $TR_1^k \in [20-120]$Hz, $TR_4^k \in [350-600]$Hz, $TR_5^k \in [600-900]$Hz, $TR_6^k \in [900-1250]$Hz and $TR_9^k \in [2000-2700]$Hz. We find the resonance frequencies of the target FRF, $\overline{TR}_i$ within the same frequency ranges.

Fig. 34 shows the target FRF and an FRF from an agent k. We consider the following values for the track resonances of the target, $\overline{TR}_1 = 60$Hz, $\overline{TR}_4 = 426$Hz, $\overline{TR}_5 = 700$Hz, $\overline{TR}_6 = 1068$Hz, and $\overline{TR}_9 = 2292$Hz. In the case of the agent k, $TR_1^k = 48$Hz, $TR_4^k = 409$Hz, $TR_5^k = 638$Hz, $TR_6^k = 953$Hz and $TR_9^k = 2066$Hz. The ranges of the local search are important, as for example in Fig. 34b. We see that the track resonances of $\overline{TR}_6$ and $\overline{TR}_9$ are outside the pre-defined frequency ranges. The algorithm takes the local optimum within the pre-defined range as optimum. This is more likely to occur when the rail track parameters have a larger difference compared with the target rail track parameters. More flexible ranges can be considered for this local search, or alternative methods for identification of resonances.

$$J_3\left(x^k\right) = \sum_{i\in\{1,4,5,6,9\}} \left|TR_i^k - \overline{TR}_i\right| \tag{15}$$



(a) Target FRF (b) FRF of agent k

Figure 34: It is assumed that the peaks of the evaluation are within the same frequency range as the target. The objective function find the indices of the resonance peaks. The fitness is obtained by summing the absolute differences of the index arrays

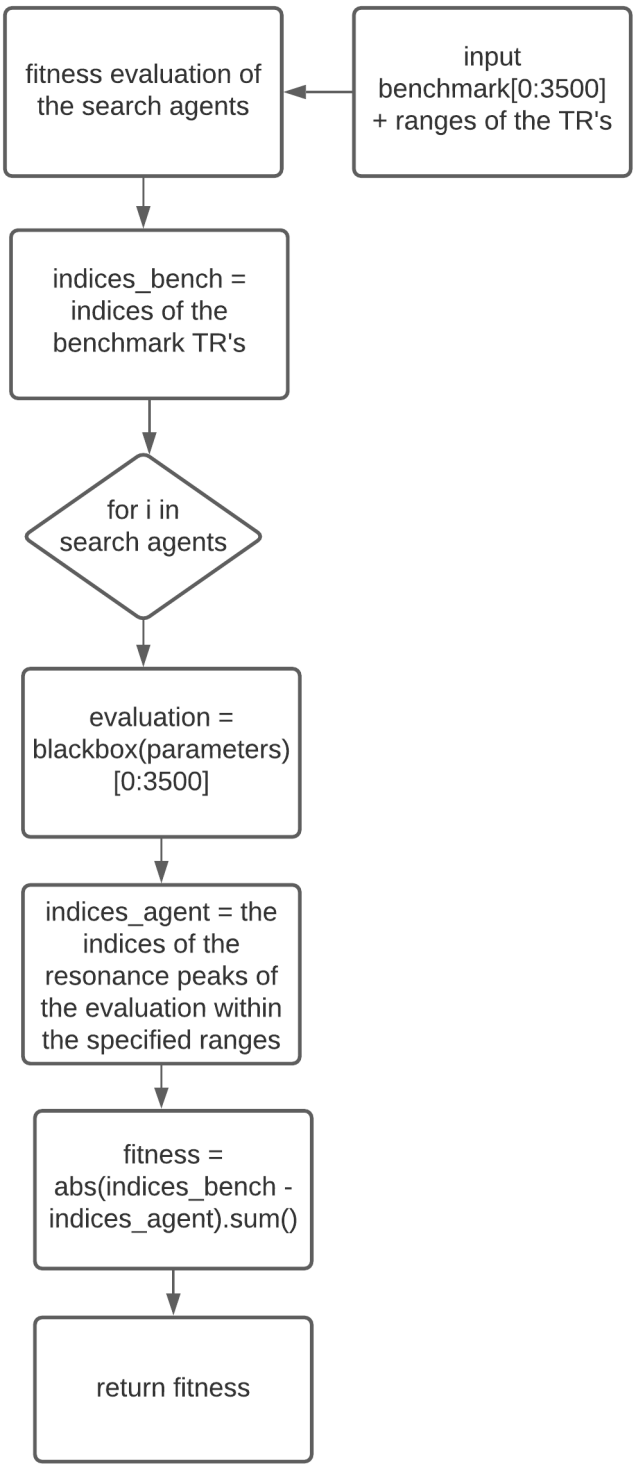


Figure 35: The flowchart of J_3

5 Experimental Results

The objective functions described earlier are tested in this chapter. We test these objective functions by applying the optimizers GWO, PSO and GA. The tested objective functions are J_1, J_2 and J_3. In this section the numerical results are represented in tables and in figures. The ultimate goal is to estimate the railway track parameters, therefore we have a look at the optimized railway track parameters in comparison with the target. We also visualise the convergence process of the optimizers. The multi-dimensional data is visualized by means of parallel-data plots. In the last subsection the positions of the last iteration from the best-overall solution of all optimizers of an objective function is put in the other two objective functions. We do this to gain insight whether a good fitness score with one objective function results automatically in a good fitness score with the other objective functions.

5.1 Experiment 1

In experiment 1 the performance of objective function J_1 is tested.

5.1.1 Numerical results

In this section the numerical results of experiment 1 are presented. The fitness values from the objective functions, the optimized railway track parameters and the relative differences between the optimized parameters and the target parameters are shown in Table 16, Table 17 and Table 18.

Table 16: The numerical outcomes of the application of the optimizers on objective function J_1.

J_1	Mean	Best	Worst	STD
GWO	$8.780e-8$	$3.508e-8$	$1.374e-7$	$2.738e-8$
PSO	$1.311e-7$	$4.642e-8$	$4.523e-7$	$1.218e-7$
GA	$1.808e-7$	$5.722e-8$	$4.114e-7$	$1.119e-7$

Table 17: The table shows for each optimizer the best optimized parameters. These parameters resulted in the best finess-score. The first row of the table shows the target parameters.

Optimizer	KP	CP	KB	CB
target	$4.300e+09$	$9.750e+04$	$6.00e+07$	$9.400e+04$
GWO	$3.130e+09$	$1.000e+05$	$5.087e+07$	$9.008e+04$
PSO	$3.288e+09$	$7.798e+04$	$5.002e+07$	$8.228e+04$
GA	$2.670e+09$	$9.516e+04$	$4.949e+07$	$9.326e+04$

Table 18: Relative difference of the estimated parameters compared to the target parameters for J_1

Optimizer	KP	CP	KB	CB	SUM
GWO	21.2%	10.6%	16.7%	7.3%	55.9%
PSO	23.5%	20.0%	16.6%	12.4%	72.6%
GA	23.4%	6.9%	15.7%	16.4%	62.5%

Fig. 36 shows the numerical data visualised with a boxplot.

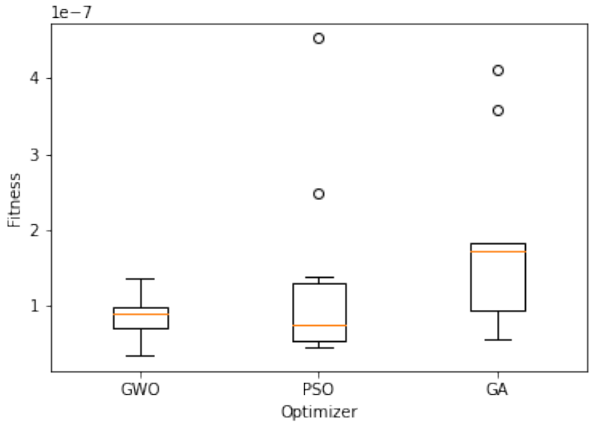
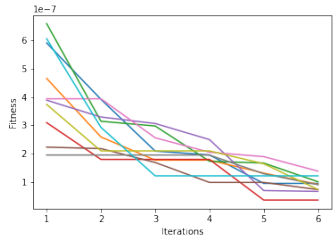


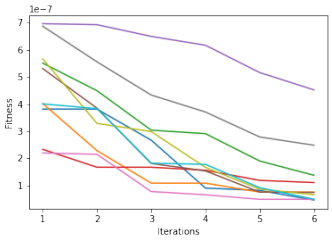
Figure 36: This figure shows the best solutions from 10 runs with each optimizer applied on J_1. From the boxplot we could determine the best solution and worst solution within the data set. Besides, the spread in results could be observed. We see that the results from GWO are less spread than the optimizers PSO and GA

5.1.2 Rate of convergence

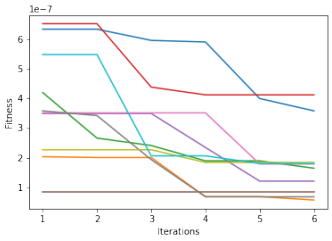
In this section we show the convergence rate per optimizer. The rate of convergence is shown in Fig. 37.



(a) Rate of convergence for GWO



(b) Rate of convergence for PSO

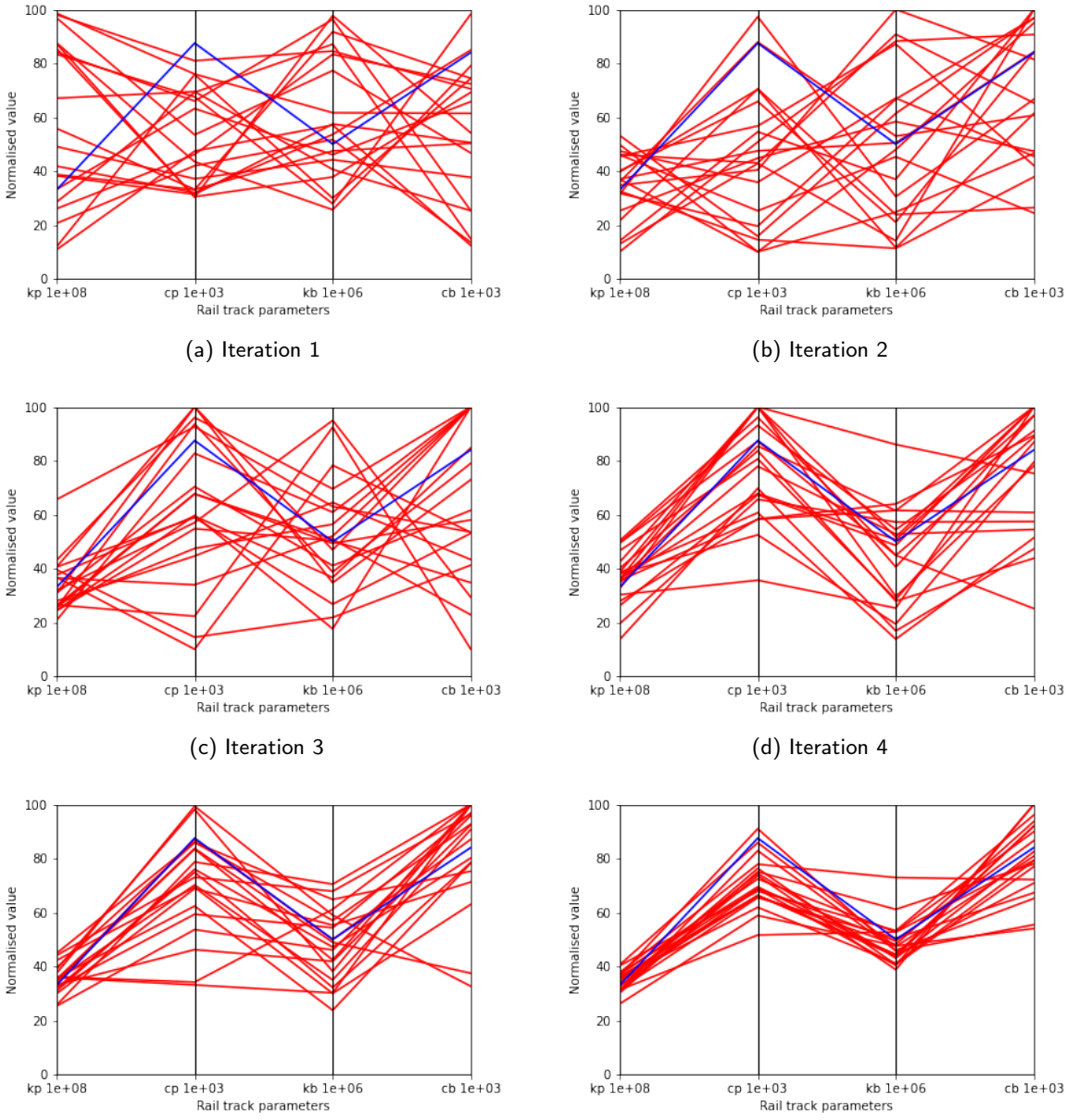


(c) Rate of convergence for GA

Figure 37: The convergence process of the optimizers applied on J_1

5.1.3 Visualisation

In this subsection the iterative process of the optimizers is visualized. Two different methods of visualization are applied. The first method is by means of a parallel-data plot. This is visualization method especially suited for high-dimensional data. In the second method we plot all the FRF's of the iteration in the same figure. The agents are colored red and the target is a coloured blue.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

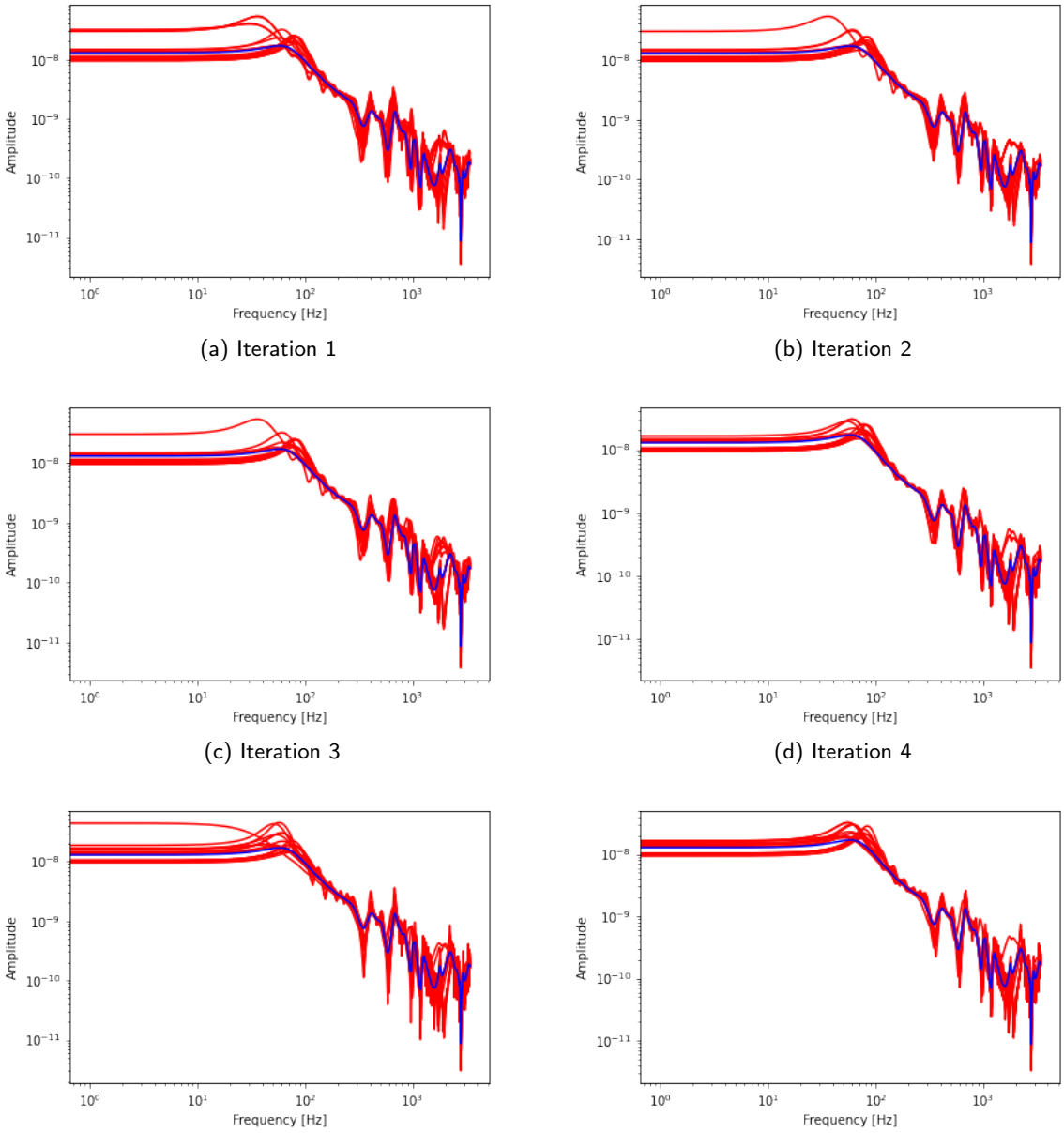
(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

Figure 38: The parallel-data visualization of the GWO optimizer applied on J_1

In Fig. 38 we see the parallel-data visualization of the GWO optimizer applied on J_1. Each red line corresponds with a set of railway track parameters. With this set of parameters we simulate the FRF in the next figure. In Fig. 39 we see the all the FRF's for each iteration plotted in one figure.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

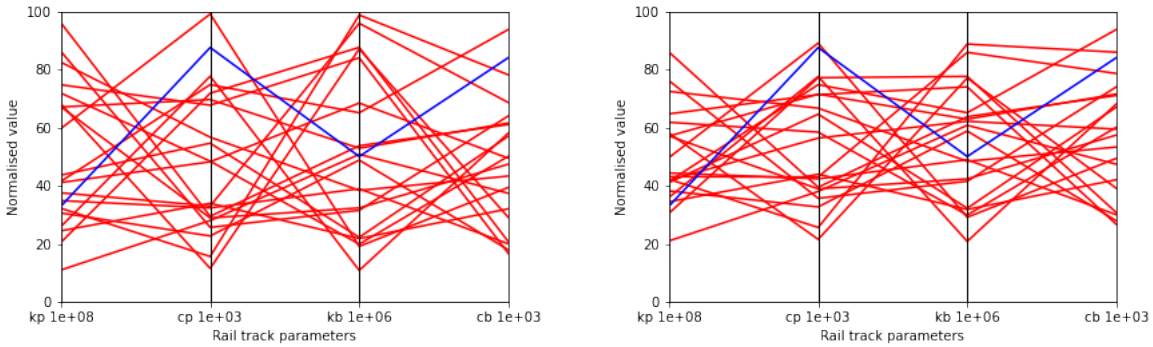
(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

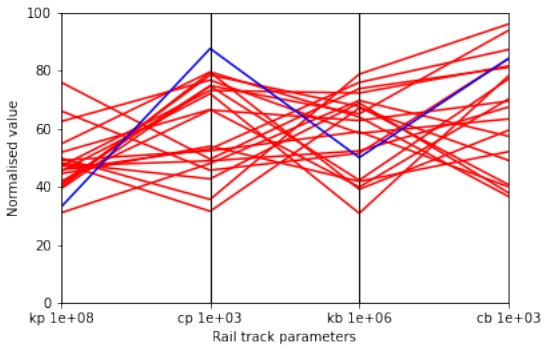
Figure 39: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the GWO optimizer is applied on J_1

In Fig. 40 we see the parallel-data visualization of the PSO algorithm applied on J_1.

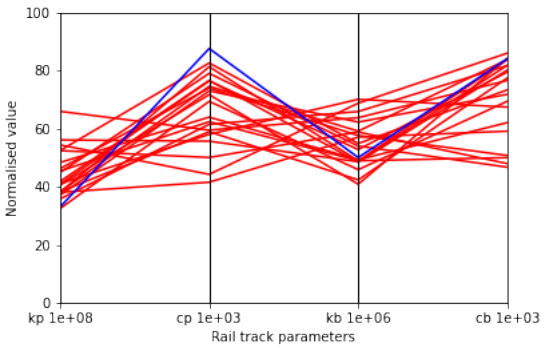


(a) Iteration 1

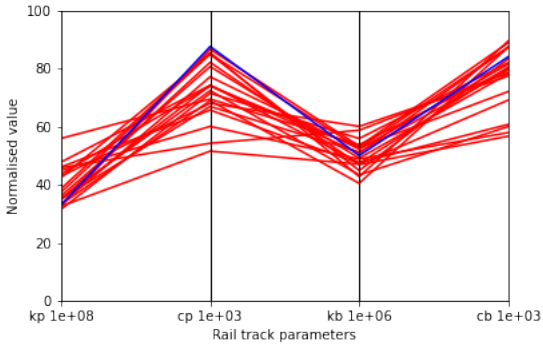
(b) Iteration 2



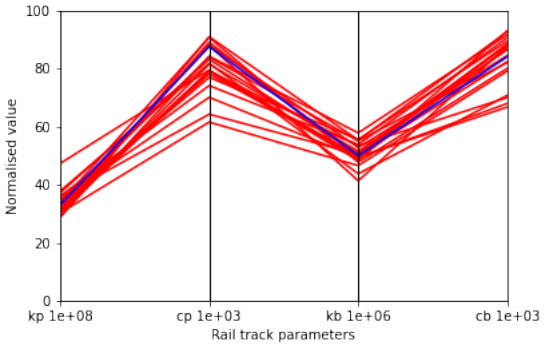
(c) Iteration 3



(d) Iteration 4



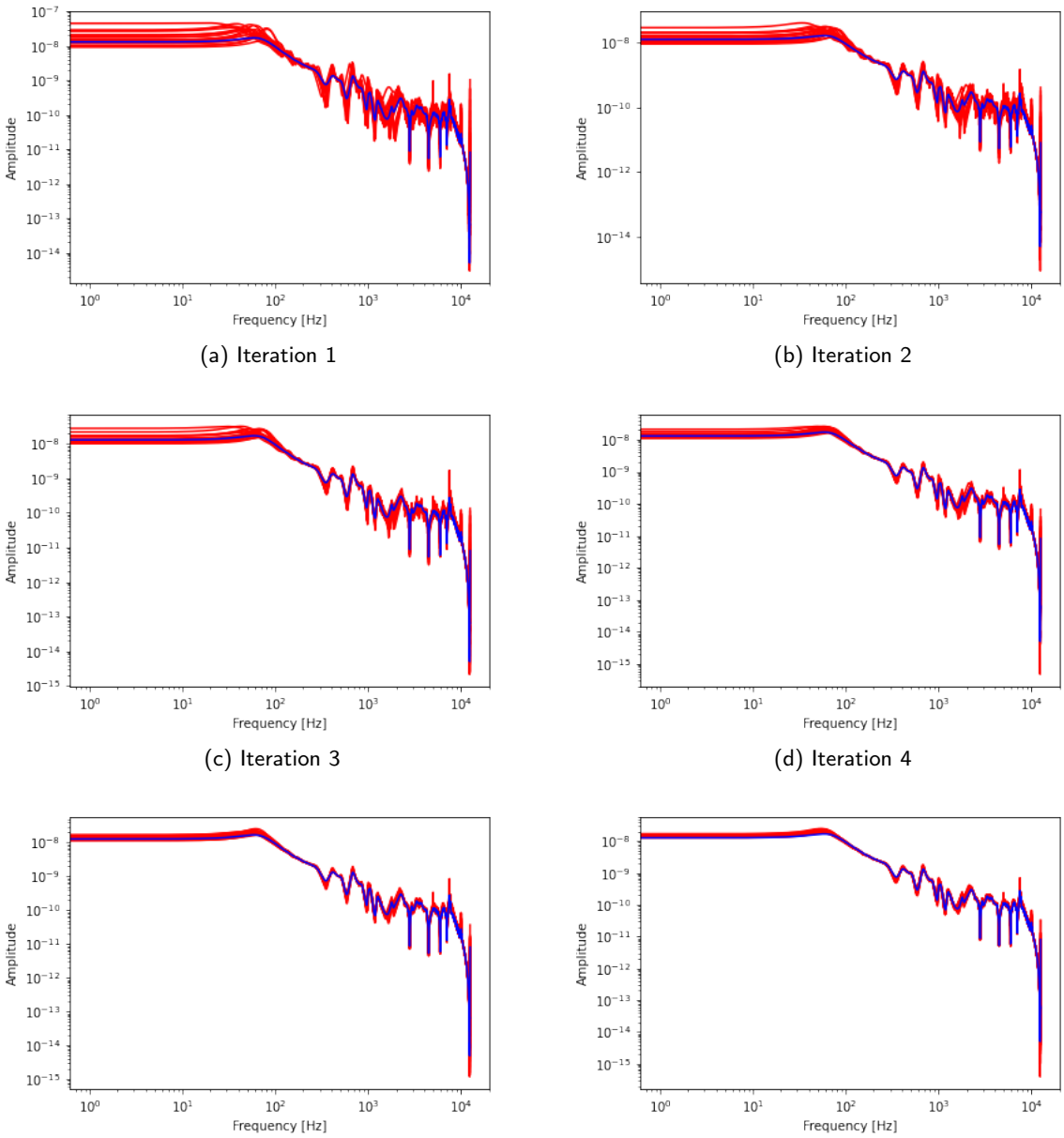
(e) Iteration 5



(f) Iteration 6

Figure 40: The parallel-data visualization of the PSO optimizer applied on J_1

In Fig. 41 the FRF's that are generated from the sets of parameters are shown.



(a) Iteration 1

(b) Iteration 2

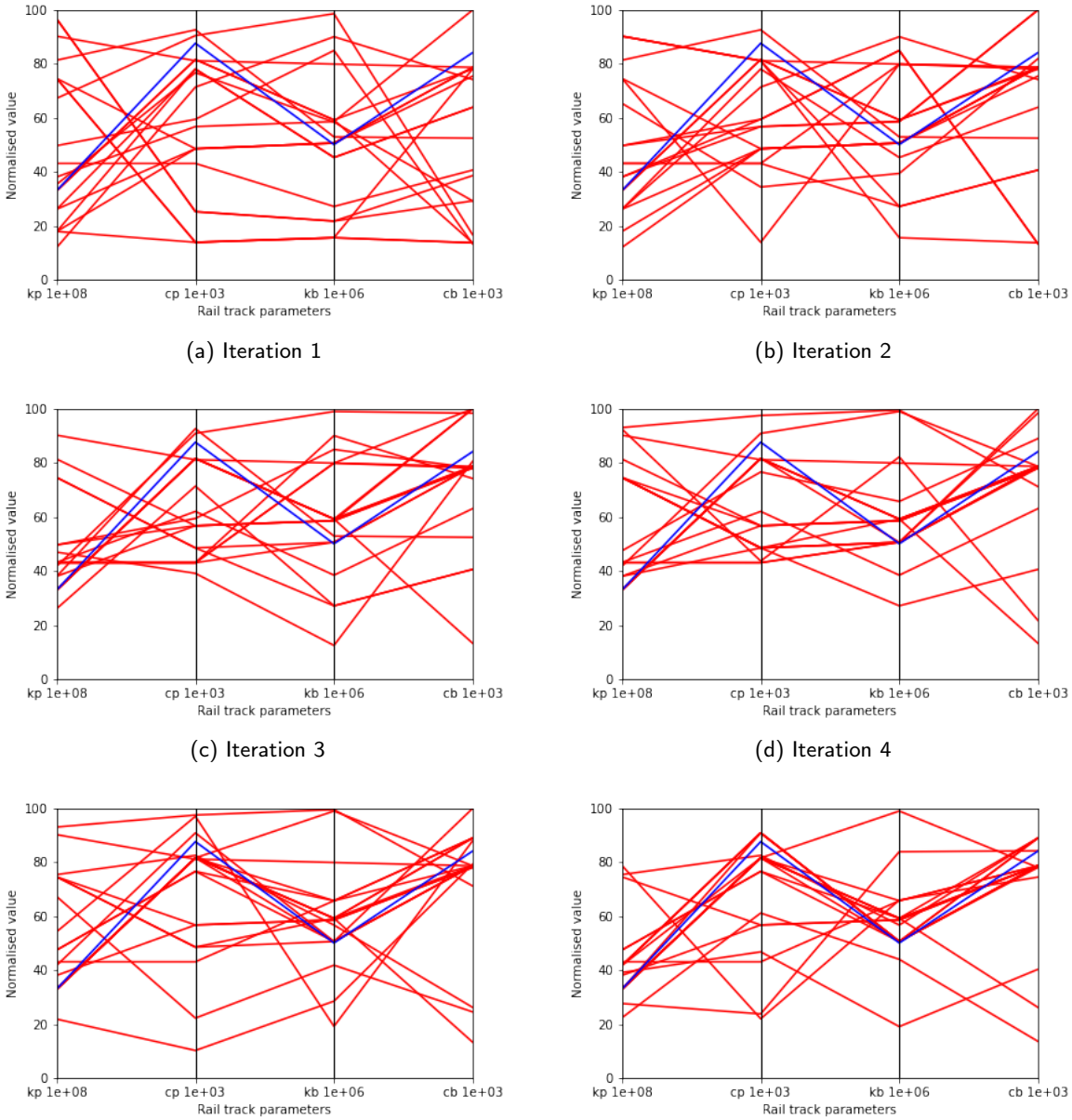
(c) Iteration 3

(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

Figure 41: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the PSO optimizer is applied on J_1



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

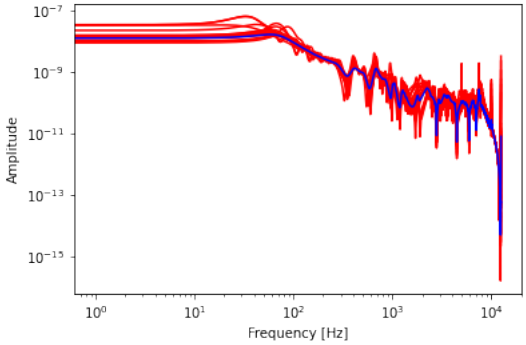
(d) Iteration 4

(e) Iteration 5

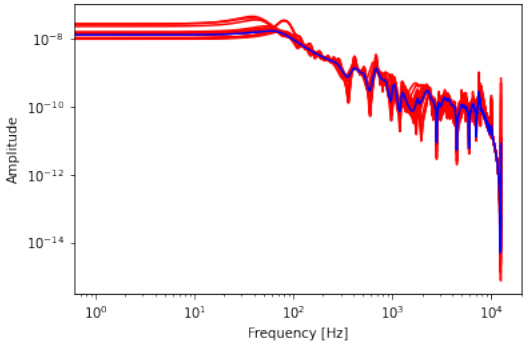
(f) Iteration 6

Figure 42: The parallel-data visualization of the GWO optimizer applied on J_1

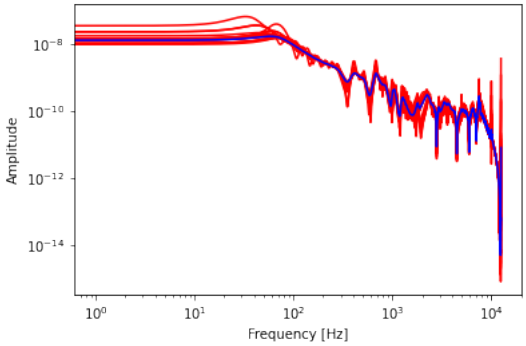
In Fig. 42 and Fig. 43 the iterations are visualised. Here the GA optimizers is applied on J_1.



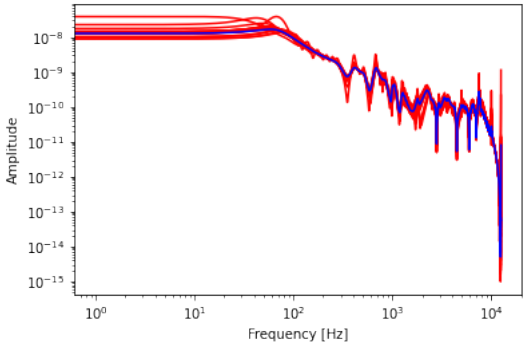
(a) Iteration 1



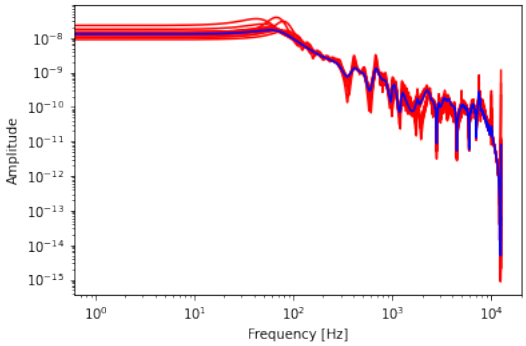
(b) Iteration 2



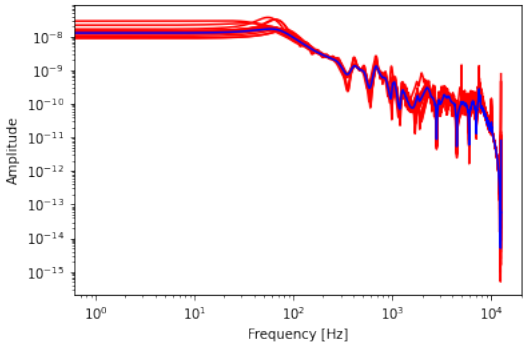
(c) Iteration 3



(d) Iteration 4



(e) Iteration 5



(f) Iteration 6

Figure 43: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the GA optimizer is applied on J_1

5.2 Experiment 2

In experiment 2 the performance of objective function J_2 is tested.

5.2.1 Numerical results

In this section the numerical results of experiment 2 are presented. Fig. 44 shows the numerical data visualised with a boxplot.

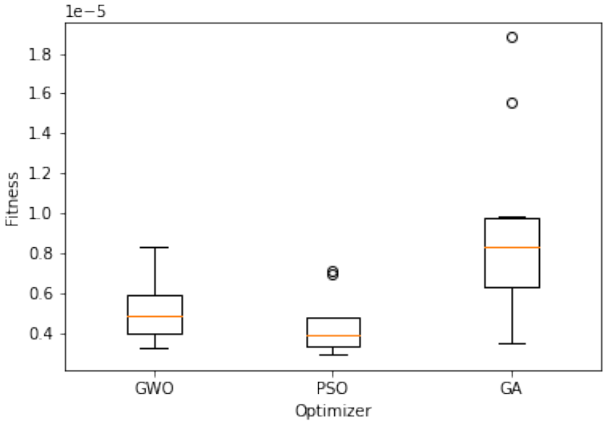


Figure 44: This figure shows the best solutions from 10 runs with each optimizer applied on J_2. From the boxplot we could determine the best solution and worst solution within the data set. Besides, the spread in results could be observed. We see that the results from GWO are less spread than PSO and GA

Table 19: The numerical outcomes of the application of the optimizers on objective function J_2

J_2	Mean	Best	Worst	STD
GWO	$2.161e-07$	$3.508e-08$	$6.595e-07$	$1.346e-07$
PSO	$2.700e-07$	$4.642e-08$	$6.964e-07$	$1.908e-07$
GA	$2.740e-07$	$5.722e-08$	$6.512e-07$	$1.692e-07$

Table 20: The table shows for each optimizer the best optimized parameters. These parameters resulted in the best finess-score. The first row of the table shows the target parameters

Optimizer	KP	CP	KB	CB
target	$4.300e+09$	$9.750e+04$	$6.00e+07$	$9.400e+04$
GWO	$4.287e+09$	$7.976e+04$	$6.005e+07$	$9.792e+04$
PSO	$4.225e+09$	$7.493e+04$	$5.988e+07$	$9.172e+04$
GA	$4.345e+09$	$3.899e+04$	$5.761e+07$	$9.742e+04$

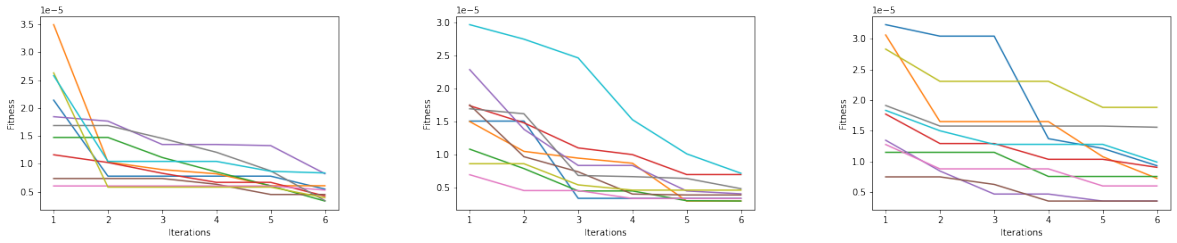
Table 21: Relative difference of the estimated parameters compared to the target parameters for J_2

Optimizer	KP	CP	KB	CB	SUM
GWO	0.28%	18.1%	0.08%	4.17%	22.7%
PSO	1.73%	23.1%	0.19%	2.4%	27.5%
GA	1.06%	7.6%	3.9%	3.6%	16.2%

The fitness values from the objective functions, the optimized railway track parameters and the relative differences between the optimized parameters and the target parameters are shown in Table 19, Table 20 and Table 21.

5.2.2 Rate of convergence

In this section we show the convergence rate per optimizer. The rate of convergence is shown in Fig. 45.

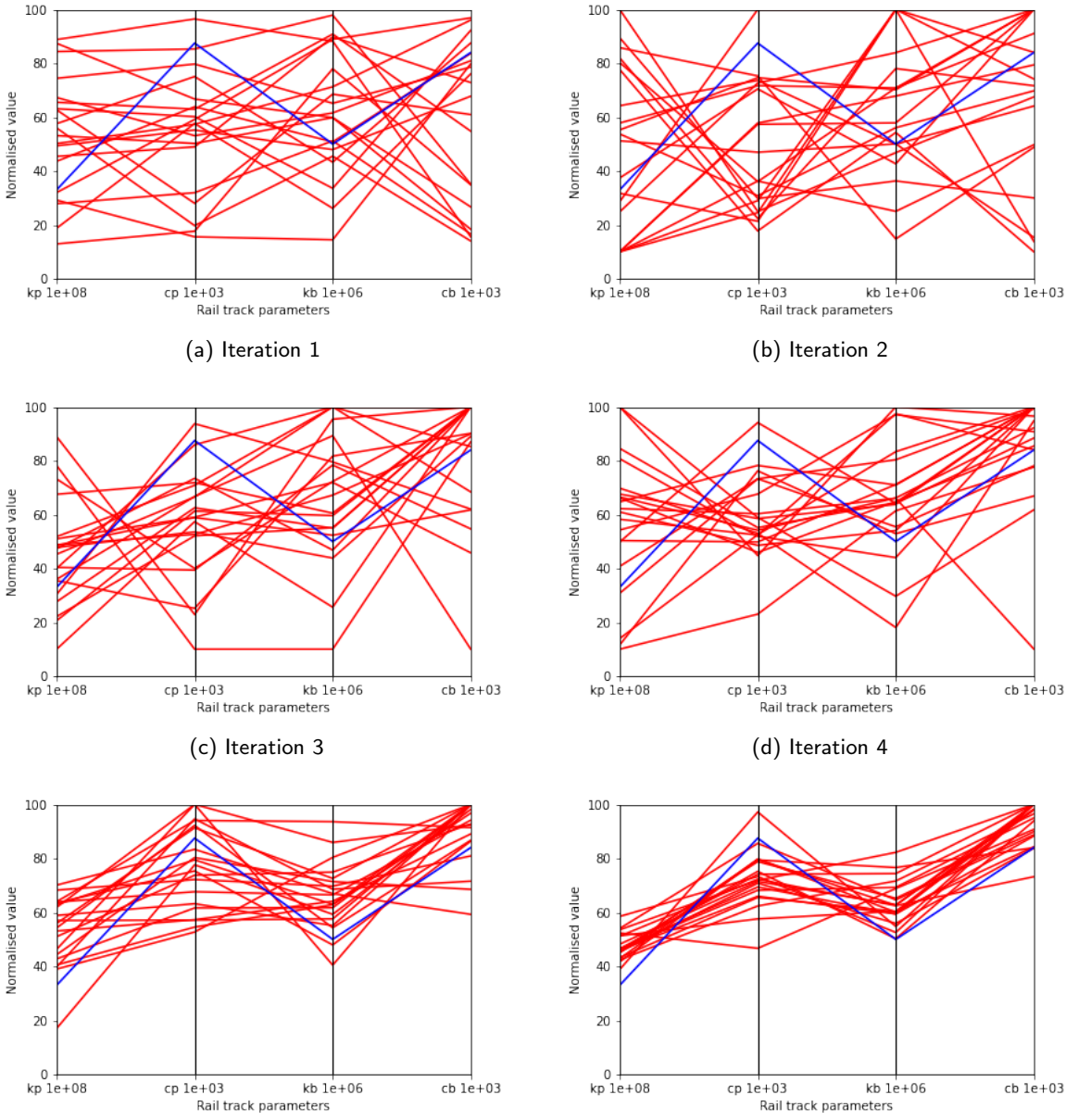


(a) Rate of convergence for GWO (b) Rate of convergence for PSO (c) Rate of convergence for GA

Figure 45: The convergence process of the optimizers applied on J_2

5.2.3 Visualisation

In this subsection the iterative process of the optimizers is visualized. Two different methods of visualization are applied. The first method is by means of a parallel-data plot. This is visualization method especially suited for high-dimensional data. In the second method we plot all the FRF's of the iteration in the same figure. The agents are colored red and the target is a coloured blue.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

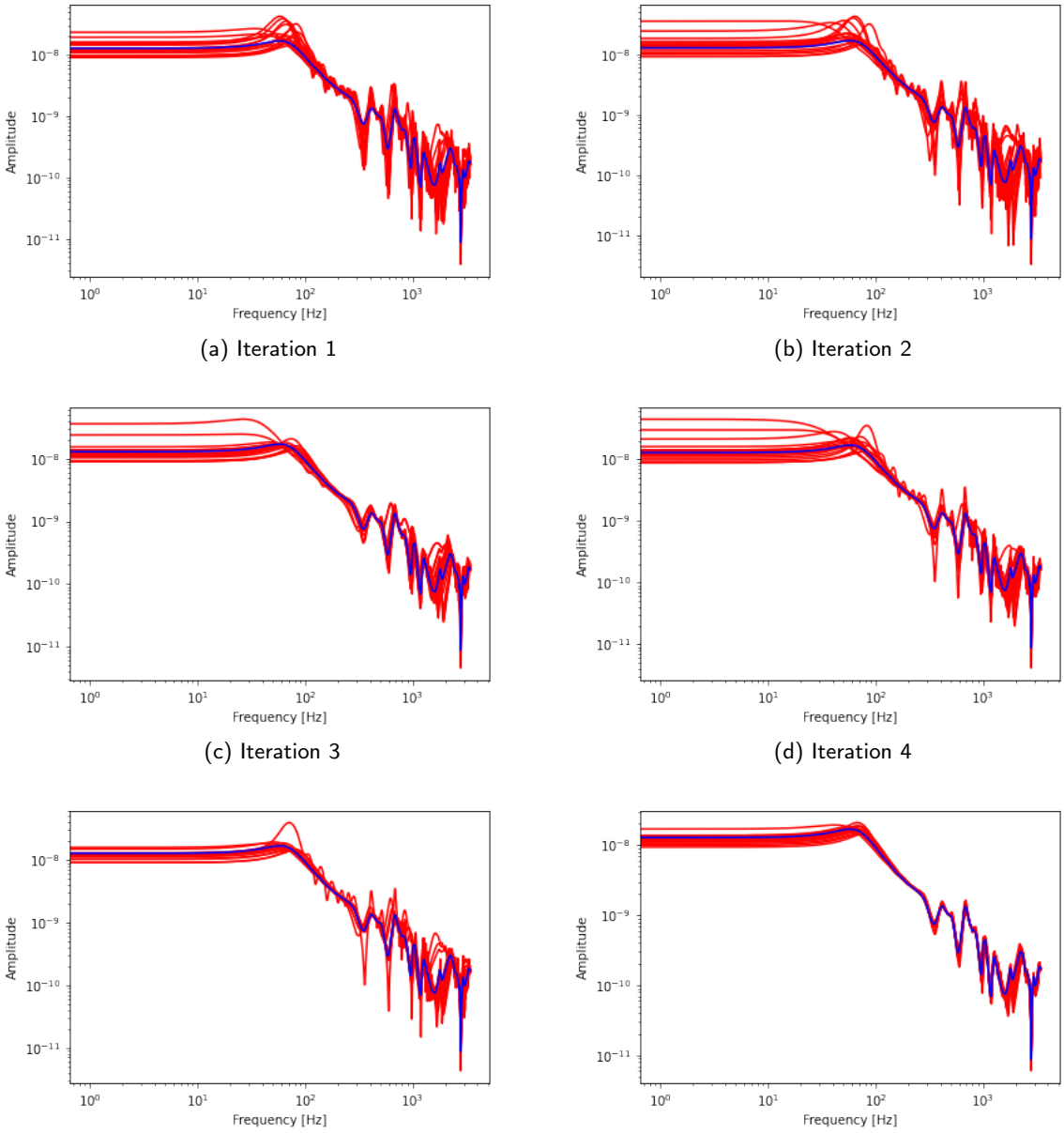
(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

Figure 46: The parallel-data visualization of the GWO optimizer applied on J_2

In Fig. 46 we see the parallel-data visualization of the GWO optimizer applied on J_2. Each red line corresponds with a set of railway track parameters. With this set of parameters we simulate the FRF in the next figure. In Fig. 47 we see the all the FRF's for each iteration plotted in one figure.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

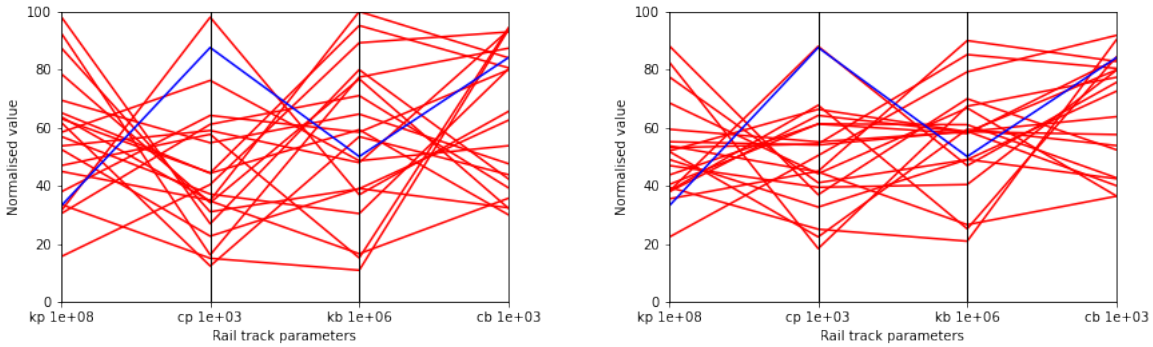
(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

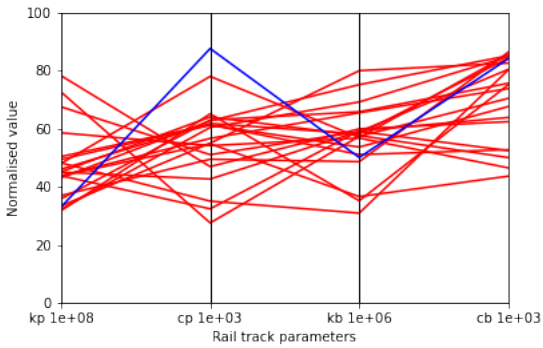
Figure 47: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the GWO optimizer is applied on J_2

In Fig. 48 we see the parallel-data visualization of the PSO algorithm applied on J_2.

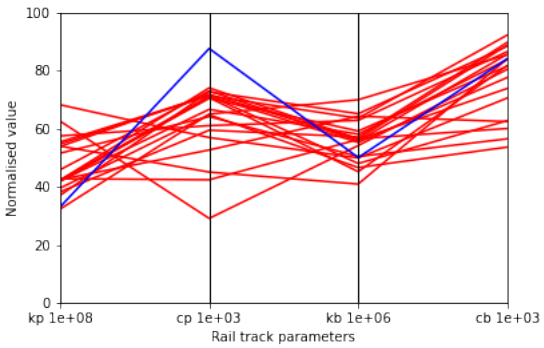


(a) Iteration 1

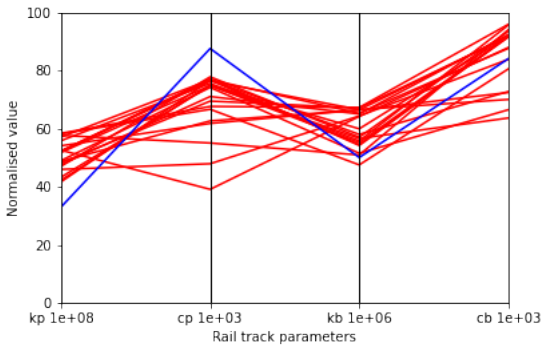
(b) Iteration 2



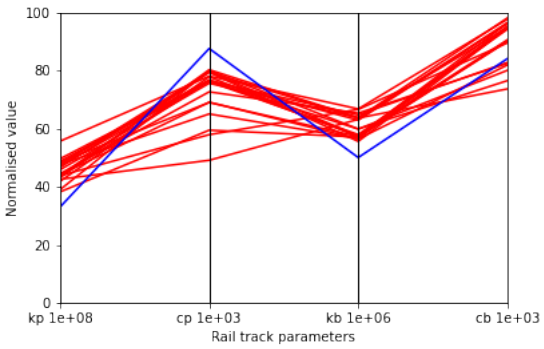
(c) Iteration 3



(d) Iteration 4



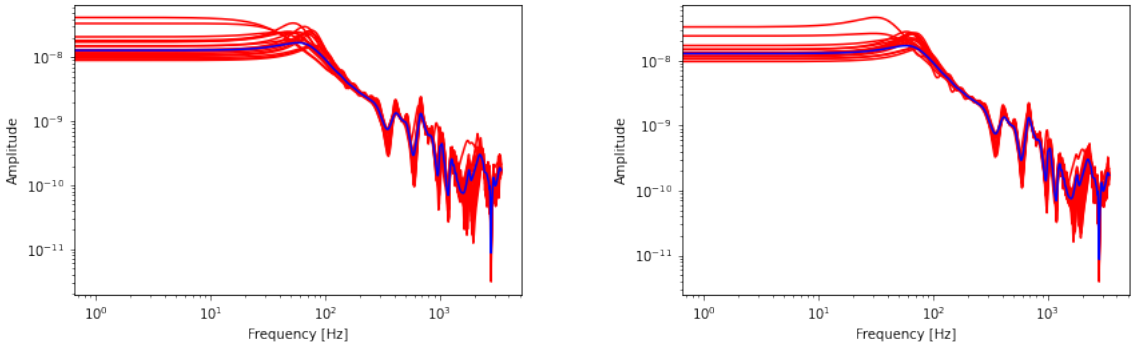
(e) Iteration 5



(f) Iteration 6

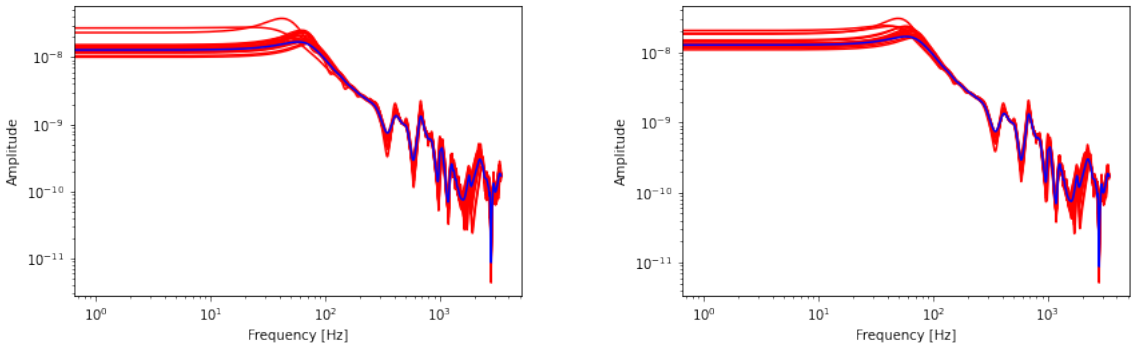
Figure 48: The parallel-data visualization of the PSO optimizer applied on J_2

In Fig. 49 the FRF's that are generated from the sets of parameters are shown.



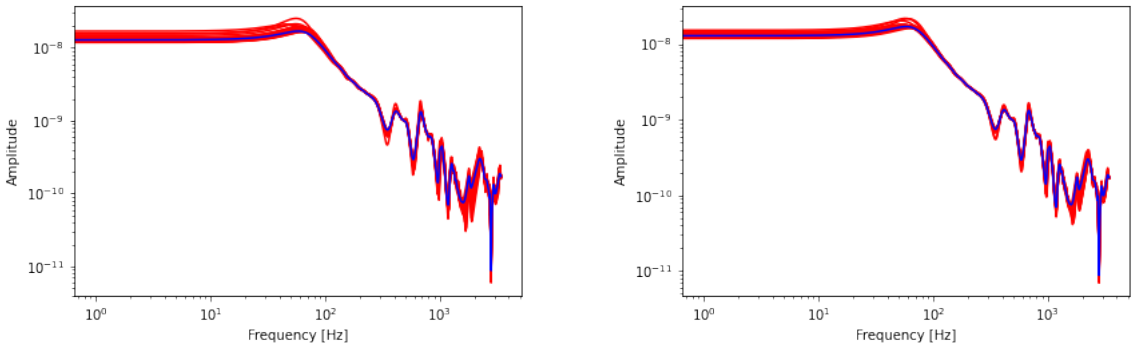
(a) Iteration 1

(b) Iteration 2



(c) Iteration 3

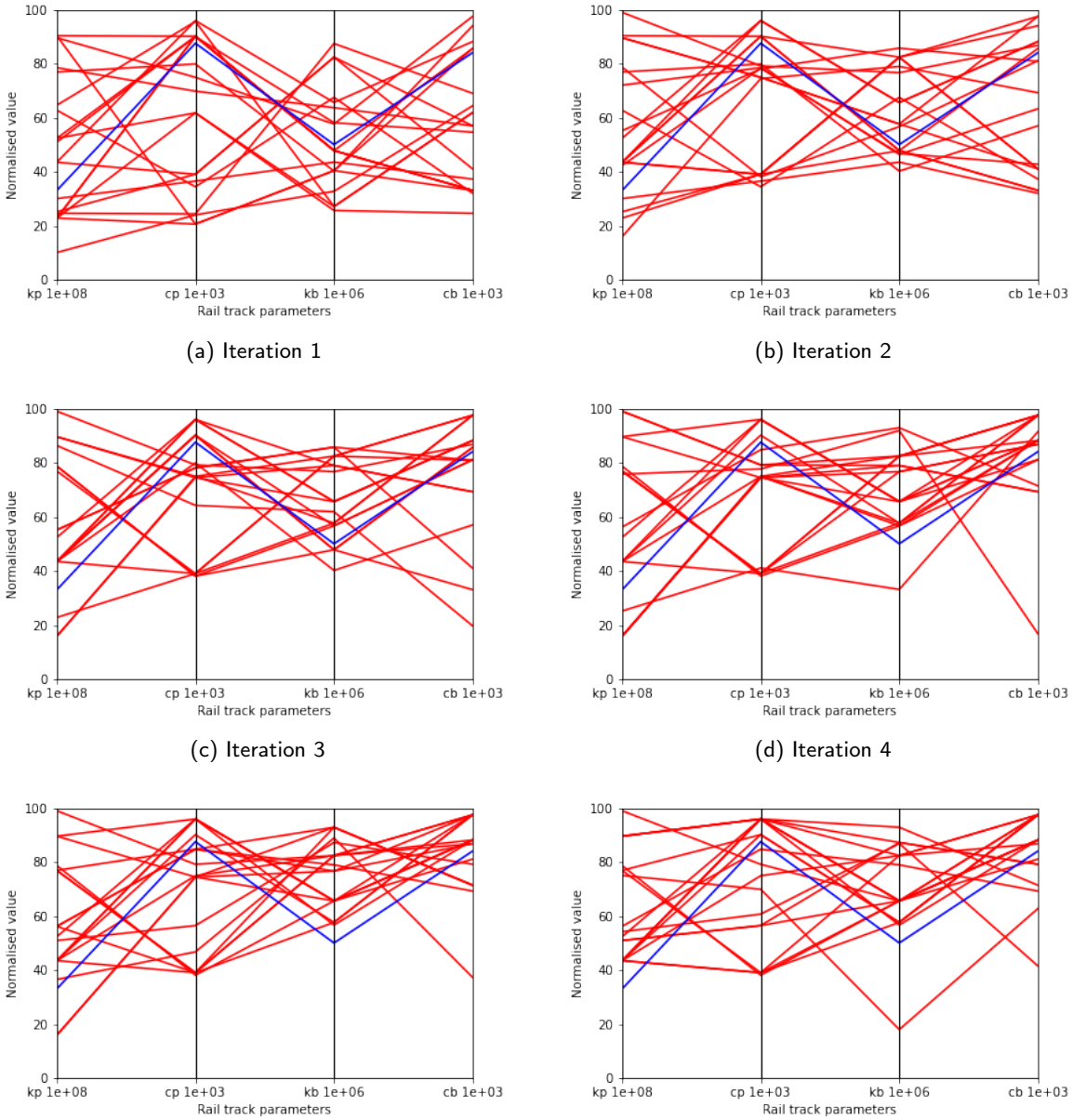
(d) Iteration 4



(e) Iteration 5

(f) Iteration 6

Figure 49: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the PSO optimizer is applied on J_2



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

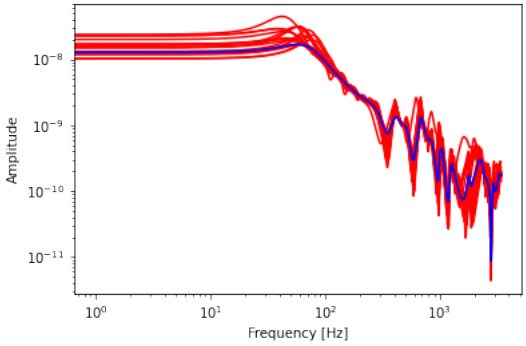
(d) Iteration 4

(e) Iteration 5

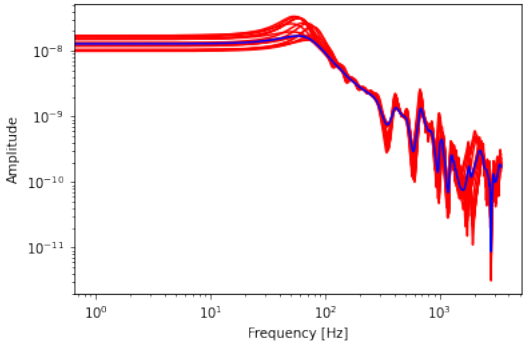
(f) Iteration 6

Figure 50: The parallel-data visualization of the GWO optimizer applied on J_2

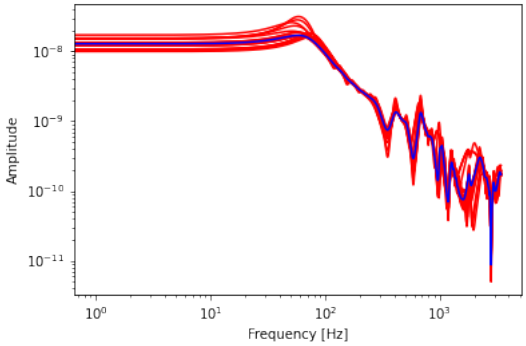
In Fig. 50 and Fig. 51 the iterations are visualised. Here the GA optimizers is applied on J_2.



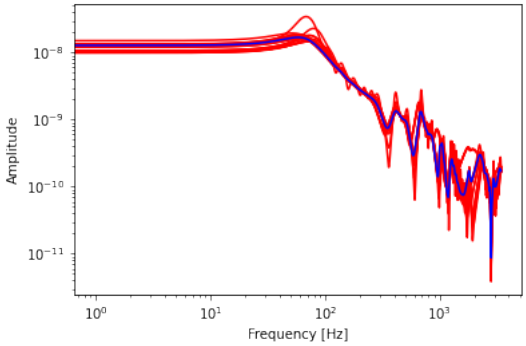
(a) Iteration 1



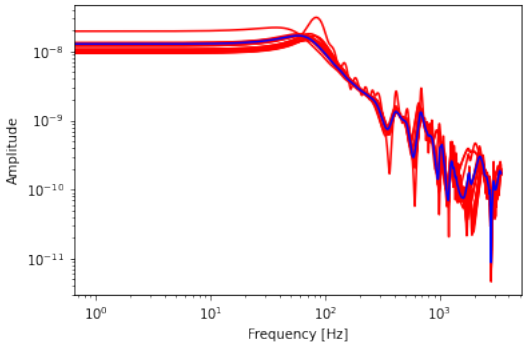
(b) Iteration 2



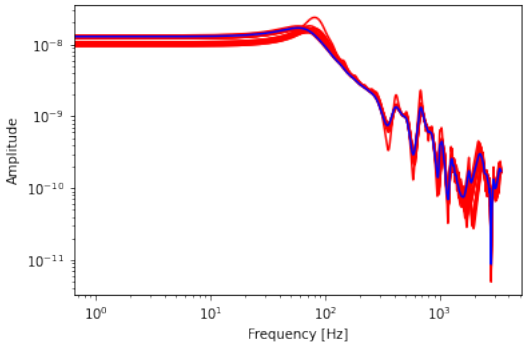
(c) Iteration 3



(d) Iteration 4



(e) Iteration 5



(f) Iteration 6

Figure 51: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the GA optimizer is applied on objective function J_2

5.3 Experiment 3

In experiment 3 the performance of objective function J_3 is tested.

5.3.1 Numerical results

In this section the numerical results of experiment 3 are presented. Fig. 52 shows the numerical data visualised with a boxplot.

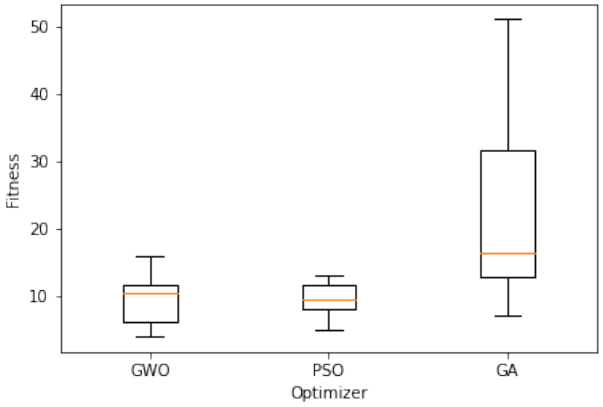


Figure 52: This figure shows the best solutions from 10 runs with each optimizer applied on J_3. From the boxplot we could determine the best solution and worst solution within the data set. Besides, the spread in results could be observed. We see that the results from GWO are less spread than PSO and GA

Table 22: The numerical outcomes of the application of the optimizers on objective function J_3

J_3	Mean	Best	Worst	STD
GWO	9.7	4.0	16.0	3.66
PSO	9.4	5.0	13.0	2.53
GA	22.4	7.0	51.0	13.12

Table 23: The table shows for each optimizer the best optimized parameters. These parameters resulted in the best finess-score. The first row of the table shows the target parameters

Optimizer	KP	CP	KB	CB
target	$4.300e+09$	$9.750e+04$	$6.00e+07$	$9.400e+04$
GWO	$4.233e+09$	$1.000e+05$	$7.057e+07$	$1.00e+05$
PSO	$4.637e+09$	$2.929e+04$	$5.179e+07$	$7.106e+04$
GA	$4.011e+09$	$7.327e+04$	$7.822e+07$	$9.913e+04$

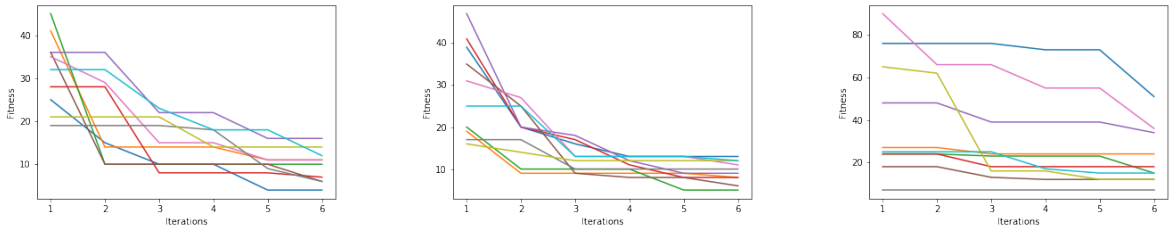
Table 24: Relative difference of the estimated parameters compared to the target parameters for J_3

Optimizer	KP	CP	KB	CB	SUM
GWO	4.41%	25.7%	0.48%	6.38%	37.0%
PSO	2.89%	14.5%	5.57%	21.0%	44.0%
GA	4.23%	36.5%	3.4%	1.6%	45.8%

The fitness values from the objective functions, the optimized railway track parameters and the relative differences between the optimized parameters and the target parameters are shown in Table 22, Table 23 and Table 24.

5.3.2 Rate of convergence

In this section we show the convergence rate per optimizer. The rate of convergence is shown in Fig. 53.

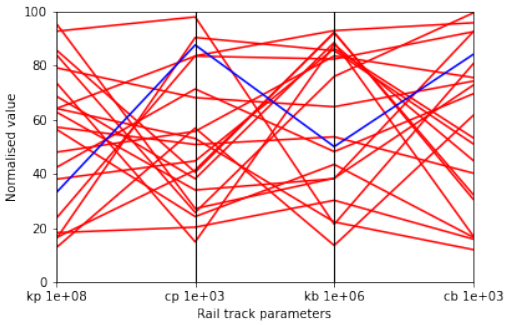


(a) Rate of convergence for GWO (b) Rate of convergence for PSO (c) Rate of convergence for GA

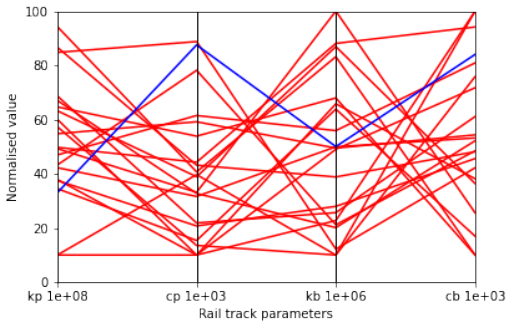
Figure 53: The convergence process of the optimizers applied on J_3

5.3.3 Visualisation

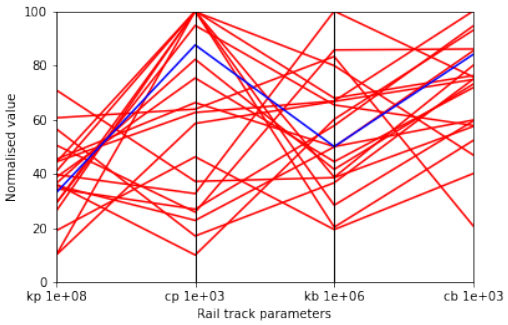
In this subsection the iterative process of the optimizers is visualized. Two different methods of visualization are applied. The first method is by means of a parallel-data plot. This is visualization method especially suited for high-dimensional data. In the second method we plot all the FRF's of the iteration in the same figure. The agents are colored red and the target is a coloured blue.



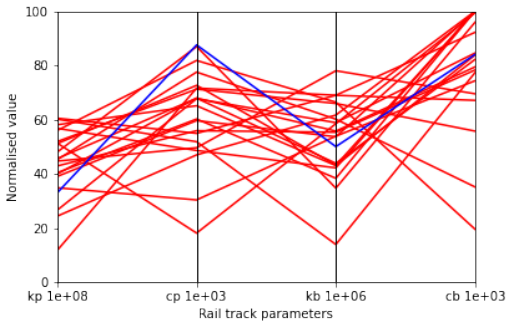
(a) Iteration 1



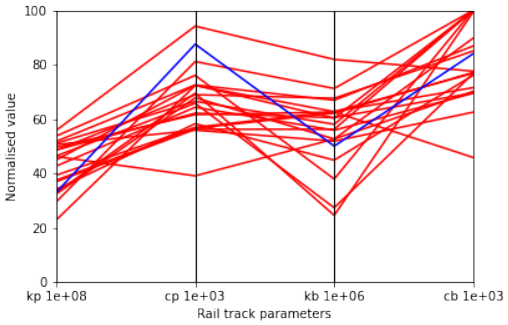
(b) Iteration 2



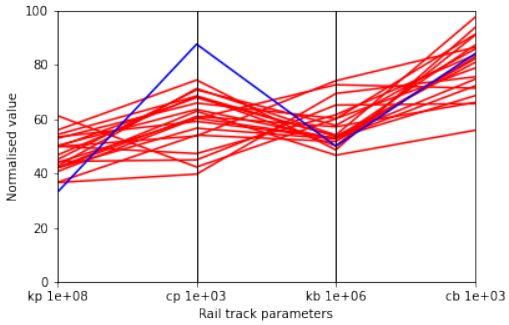
(c) Iteration 3



(d) Iteration 4



(e) Iteration 5

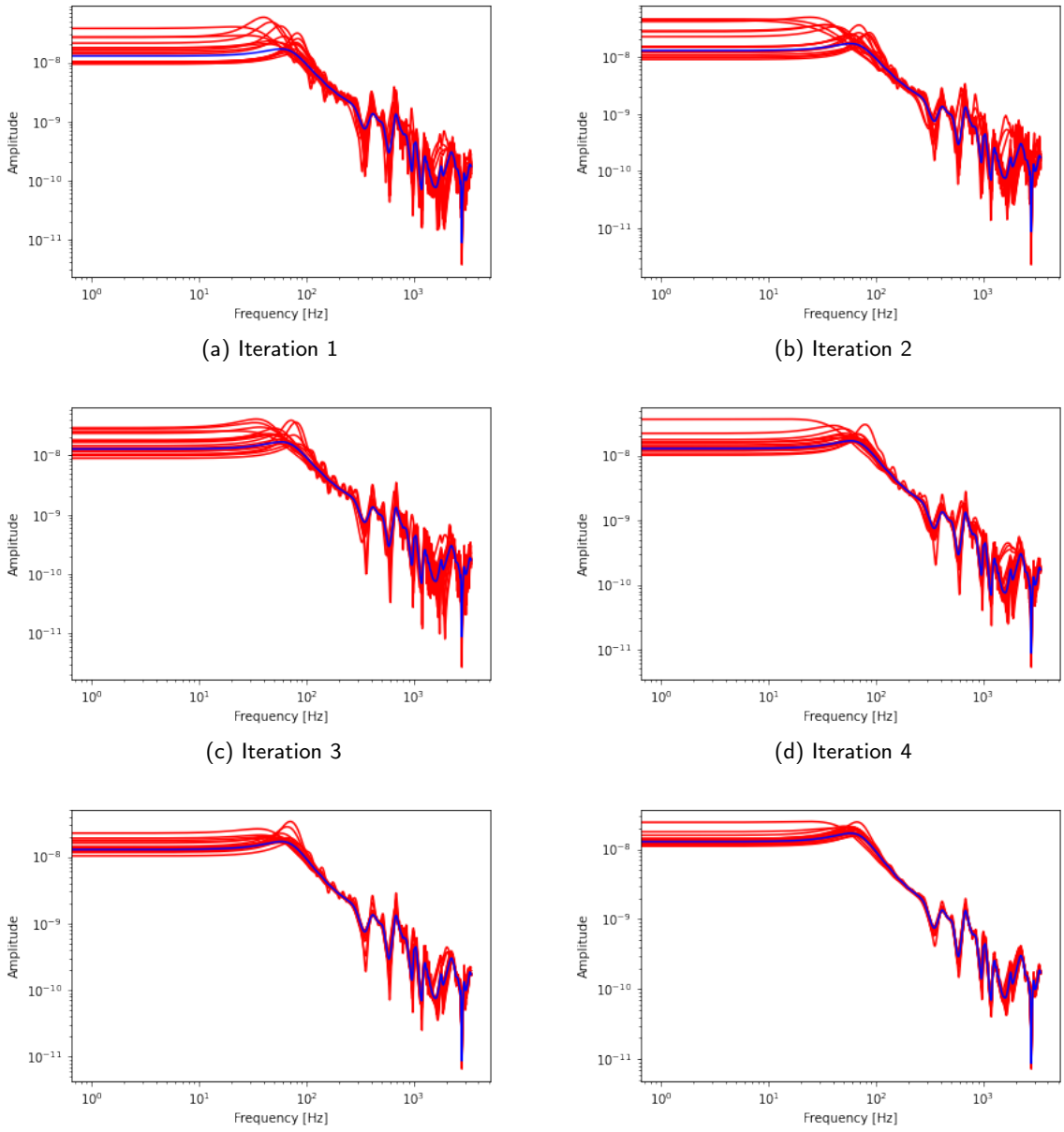


(f) Iteration 6

Figure 54: The parallel-data visualization of the GWO optimizer applied on J_3

In Fig. 54 we see the parallel-data visualization of the GWO optimizer applied on J_3. Each

red line corresponds with a set of railway track parameters. With this set of parameters we simulate the FRF in the next figure. In Fig. 55 we see the all the FRF's for each iteration plotted in one figure.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

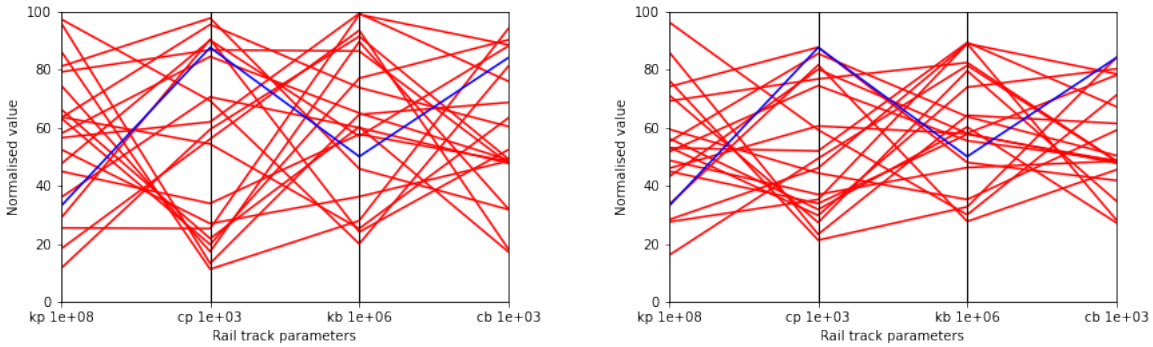
(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

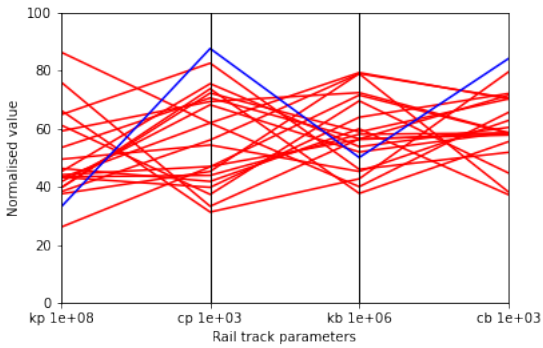
Figure 55: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the GWO optimizer is applied on J_3

In Fig. 56 we see the parallel-data visualization of the PSO algorithm applied on J_3.

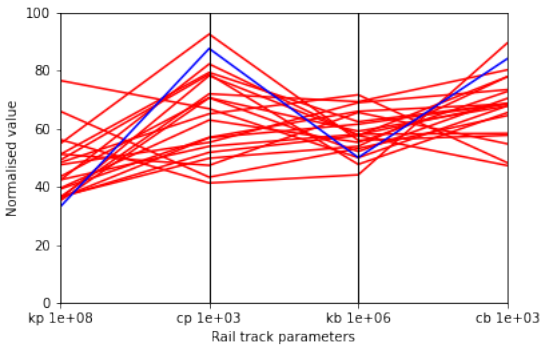


(a) Iteration 1

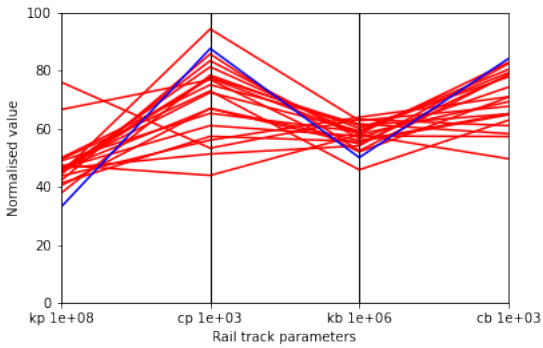
(b) Iteration 2



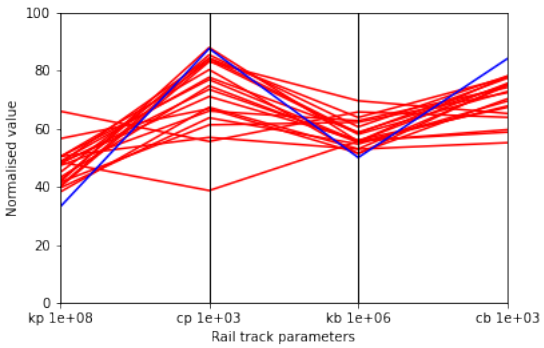
(c) Iteration 3



(d) Iteration 4



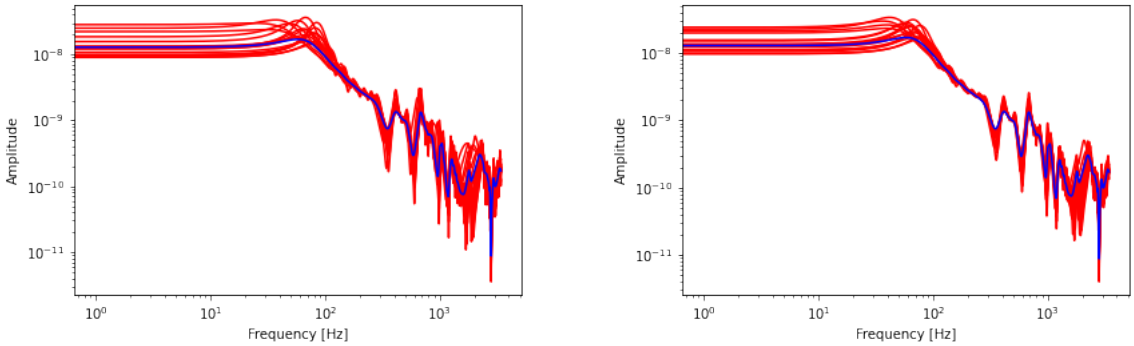
(e) Iteration 5



(f) Iteration 6

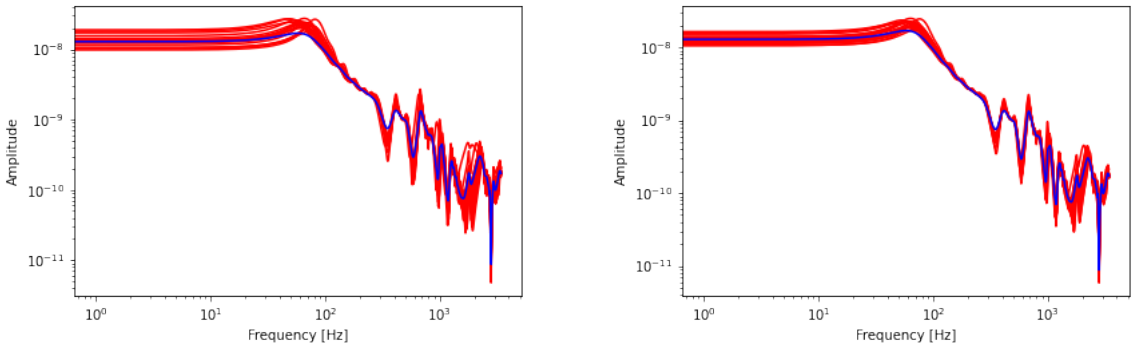
Figure 56: The parallel-data visualization of the PSO optimizer applied on J_3

In Fig. 57 the FRF's that are generated from the sets of parameters are shown.



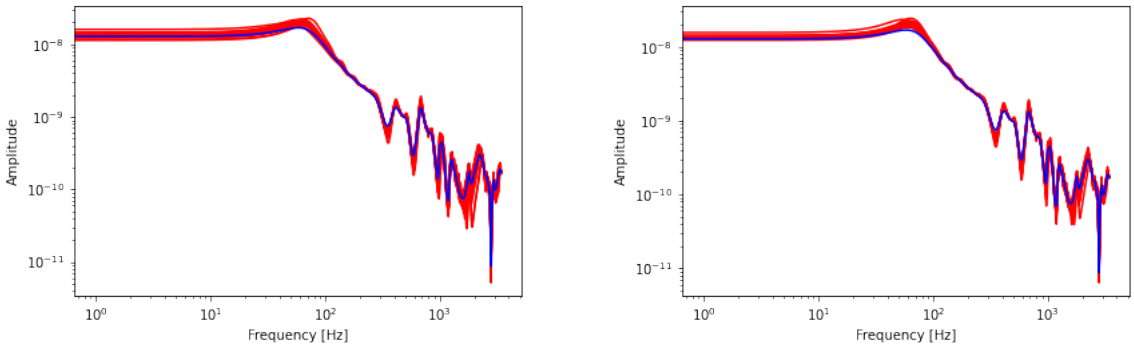
(a) Iteration 1

(b) Iteration 2



(c) Iteration 3

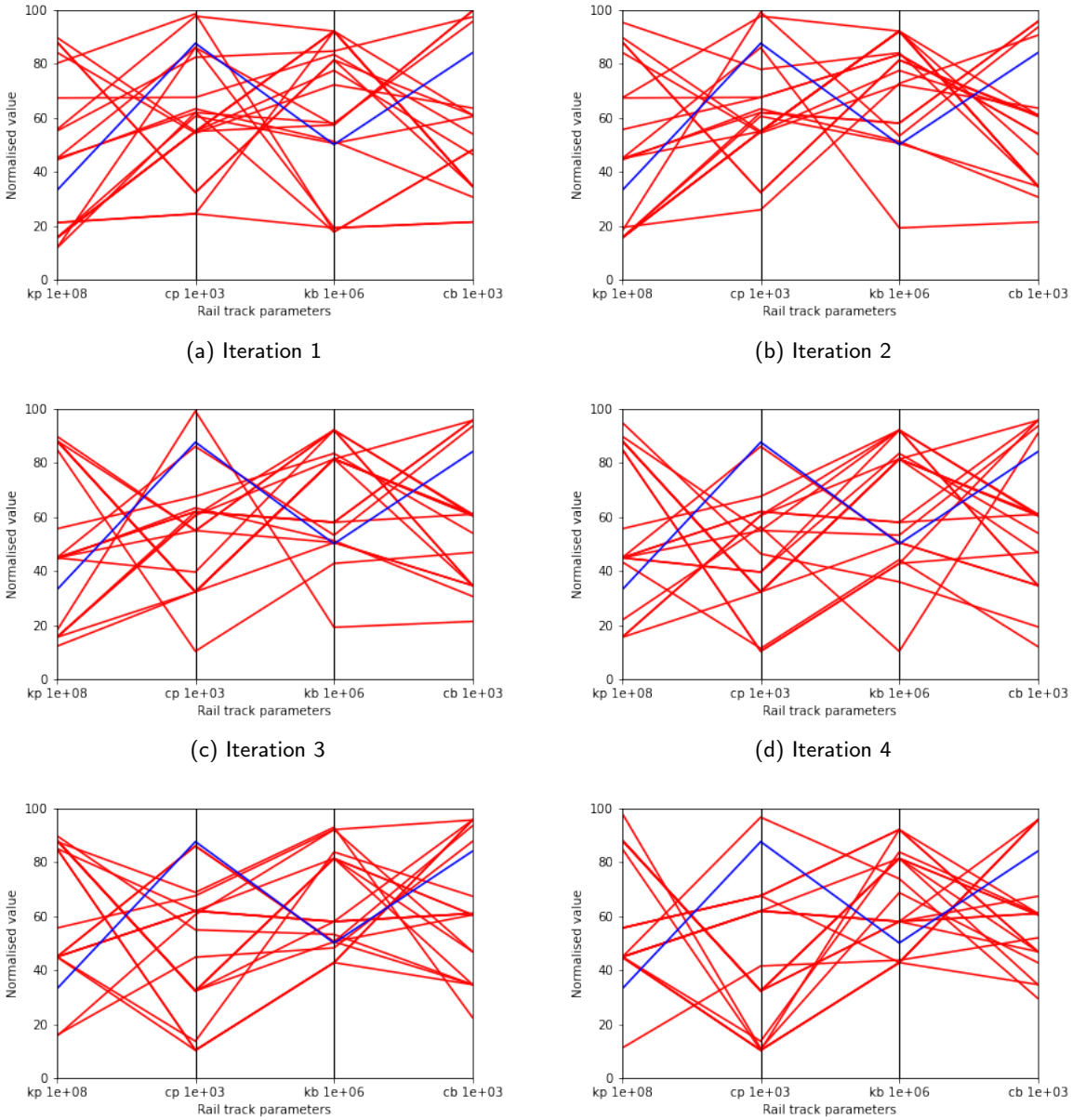
(d) Iteration 4



(e) Iteration 5

(f) Iteration 6

Figure 57: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the PSO optimizer is applied on J_3



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

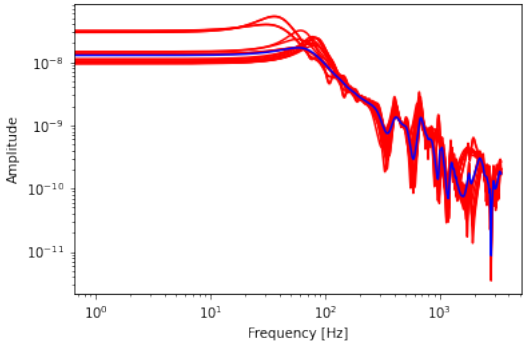
(d) Iteration 4

(e) Iteration 5

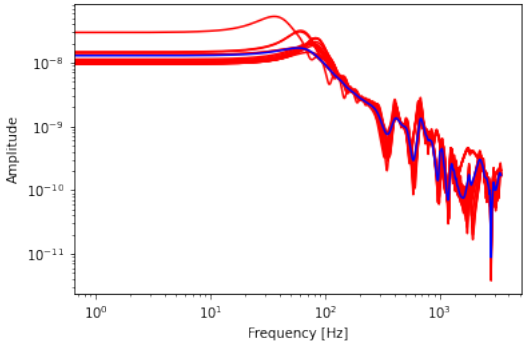
(f) Iteration 6

Figure 58: The parallel-data visualization of the GWO optimizer applied on J_3

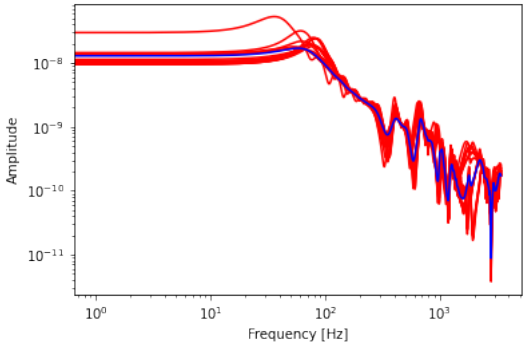
In Fig. 58 and Fig. 59 the iterations are visualised. Here the GA optimizers is applied on J_3.



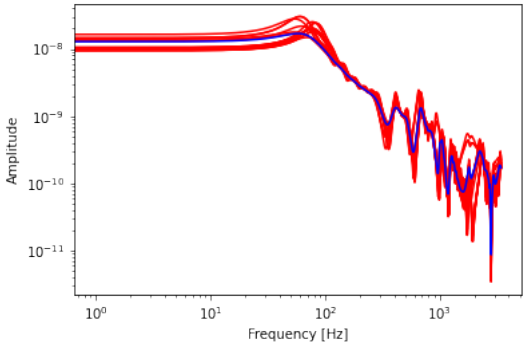
(a) Iteration 1



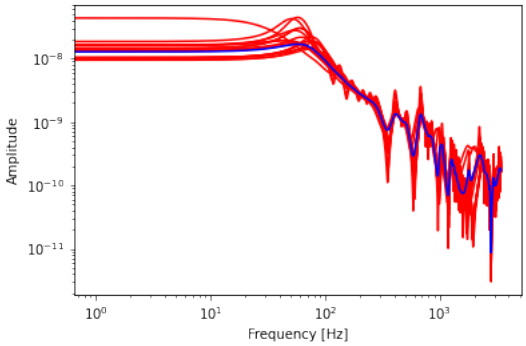
(b) Iteration 2



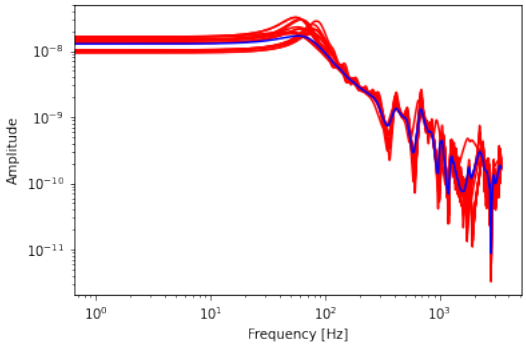
(c) Iteration 3



(d) Iteration 4



(e) Iteration 5



(f) Iteration 6

Figure 59: The search agents in this figure are colored red and the target in indicated with blue. In this figure also the GA optimizer is applied on J_3

5.4 Comparison of objective functions

The values in Table 25 are obtained by computing the fitness score of the best optimized railway track parameters for each objective function with the other objective functions.

Table 25: The columns of the table represent the best optimized railway track parameters. In the rows of the table the fitness of these parameters is reevaluated with all the objective functions J_1, J_2 and J_3

	Obtained J_1	Obtained J_2	Obtained J_3
Fitness J_1	$9.992e-08$	$1.756e-05$	143
Fitness J_2	$4.237e-07$	$6.269e-06$	21
Fitness J_3	$4.778e-07$	$1.168e-05$	10

6 Discussion and Recommendations

The objective of this thesis is to find out what the influence is of different evolutionary algorithms and objective functions on the estimation of railway track parameters. In this study we performed three experiments with different objective functions. Within each experiment three optimizers are tested. In this section the results are discussed. The most important characteristics obtained from the numerical experiments are discussed.

6.1 Rate of convergence

The rate of convergence is defined as the speed with which an optimizer reaches the global solution. For the experimental results we see that GWO has a high rate of convergence and the algorithms needs a relative low amount of iterations to reach the global optimum. The rate of convergence of GA is the lowest compared to the other optimizers. The rate of convergence of PSO is in between GWO and GA, but tends more to be like GWO. This difference in convergence rate may be the result of different ways information is shared within a population. This difference in information sharing originates from the classification of the optimizers. PSO and GWO search solutions considering the best solutions obtained so far, while GA is an evolution based optimizer that does not include information about previous performance of an agent.

6.2 Time Performance

The time performance is an important metric to determine the applicability of an algorithm. The time is measured by making use of the time module of Python, but the results are not presented. The reason behind this is that only one computer was available for the experiments. Also, conducting the experiments is computationally expensive, therefore it was necessary to run the experiments day and night. During the night more processors are available than during the day when other processes also need computation power. Therefore it was not possible to conduct the time experiment under the same conditions. This is a topic for further research.

6.3 Numerical results

The purpose of the thesis is to gain insight in which optimizer and objective functions performs best in estimating the railway track parameters. From the numerical results we can conclude that the second objective function J_2 yields the best results. Within experiment 2 we obtain the best results with GA. The second best results we obtain with J_3, this result is obtained with GWO. Based on the results we can not conclude that superior results are obtained with a specific optimizer, but we can conclude that J_2 yields the best estimation. It is interesting to look at the spread of the solutions. The spread of GA solutions is significantly larger than the spread of PSO and GWO, this means that a larger part of final results is of low quality. From this we can conclude multiple runs are a larger necessity for GA than for the other optimizers. This is an important conclusion for the computational costs of parameter estimation.

The relative error of the optimized parameters shows an interesting pattern. We see that the largest part of the error is caused by the damping of the railpad, CP. A possible explanation for this is that this parameter has no significant influence on the FRF and is therefore not optimised and assigned a random value. This is aligned with the findings of [1], where it is discussed that some track parameters are more easily identified by some track resonances responses than with others. A more tailored design of the objective function to identify particular parameters might lead into better solutions. Further research should be done to test this hypothesis.

Each experiment has been conducted by performing 10 runs. The number of runs is not sufficient to avoid stochastic influences in the results. 10 runs are performed because more runs are computationally not feasible to perform on a laptop. In order to improve the results the experiments should be conducted on a large cluster when we can do ten thousands of runs instead of 10. To move the experiments to a cluster is something for further research.

7 Conclusion

The contributions of this thesis are that we have designed a platform in Python to optimize the railway track parameters. In addition three evolutionary algorithms and three objective functions have been evaluated for the railway track parameter identification problem. The purpose of the research is to find out which optimizer and objective function performs best. In the simulation we yield the best optimized railway track parameters with objective function J_2. These results are obtained with GA. With the other objective functions the optimizer GWO generally performed best. We have found out that the parameter, damping of the railpad, is hard to optimize. In [1] we see that this parameter is relatively insensitive. Further research is needed to develop a model and method capable to better capture the parameters of railpads. Other suggestions for future research are a sensitivity analysis of optimization parameters to increase the performance of the optimization. Since the simulations are computationally expensive it is recommended to run a large number of evaluations on a cluster to avoid stochastic influences on the results. In addition we could implement other objective functions, such as combinations of J_2 and J_3 with an inclusion of logarithm in the error estimation or other ways to weight differently large deviation values. Also, real-life FRF measurements can be considered as the target FRF, for which new objective functions that can include stochasticities of the measurements and parameter estimations can be investigated. Finally, with the developed platform in this thesis, other evolutionary computation optimizers can easily be tested and compared.

References

[1] Chen Shen, Rolf Dollevoet, and Zili Li. "Fast and robust identification of railway track stiffness from simple field measurement". In: *Mechanical Systems and Signal Processing* 152 (2021), p. 107431. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2020.107431. URL: https://www.sciencedirect.com/science/article/pii/S0888327020308177.

[2] AA Nunez, M Oregui, and M Molodova. "An expensive optimization based computational intelligence method for railway track parameter identification". English. In: *Proceedings of the Proceedings of the 12th international conference on computational structures technology, Stirlingshire, UK*. Ed. by BHV Topping and P Ivanyi. Vol. 106. paper 173; 12th international conference on computational structures technology, Naples, Italy ; Conference date: 02-09-2014 Through 05-09-2014. United Kingdom: Civil-Comp Press, 2014, pp. 1–21. DOI: 10.4203/ccp.106.173.

[3] Eric Giannini et al. "Nondestructive Evaluation of In-service Concrete Structures Affected by Alkali-Silica Reaction (ASR) or Delayed Ettringite Formation (DEF), TxDOT Project 0-6491: Final Report – Part I". In: (Apr. 2013).

[4] URL: https://community.sw.siemens.com/s/article/what-is-a-frequency-response-function-frf.

[5] Stuart Grassie, R Gregory, and K Johnson. "The Dynamic Response of Railway Track to High Frequency Vertical Excitation". In: *ARCHIVE: Journal of Mechanical Engineering Science 1959-1982 (vols 1-23)* 24 (June 1982), pp. 97–102. DOI: 10.1243/JMES_JOUR_1982_024_018_02.

[6] M. Oregui, Z. Li, and R. Dollevoet. "Identification of characteristic frequencies of damaged railway tracks using field hammer test measurements". In: *Mechanical Systems and Signal Processing* 54-55 (2015), pp. 224–242. DOI: 10.1016/j.ymssp.2014.08.024.

[7] M. Oregui, Z. Li, and R. Dollevoet. "Identification of characteristic frequencies of damaged railway tracks using field hammer test measurements". In: *Mechanical Systems and Signal Processing* 54-55 (2015), pp. 224–242. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2014.08.024. URL: https://www.sciencedirect.com/science/article/pii/S0888327014003410.

[8] Andrea Saltelli. "Making best use of model evaluations to compute sensitivity indices". In: *Computer Physics Communications* 145.2 (2002), pp. 280–297. ISSN: 0010-4655. DOI: https://doi.org/10.1016/S0010-4655(02)00280-1. URL: https://www.sciencedirect.com/science/article/pii/S0010465502002801.

[9] Fernando Fausto et al. "From ants to whales: metaheuristics for all tastes". In: *Artificial Intelligence Review* 53 (Jan. 2020). DOI: 10.1007/s10462-018-09676-2.

[10] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. "Differential Evolution: A Survey of the State-of-the-Art". In: *IEEE Transactions on Evolutionary Computation* 15.1 (2011), pp. 4–31. DOI: 10.1109/TEVC.2010.2059031.

[11] Hans-Georg Beyer and Hans-Paul Schwefel. "Evolution Strategies –A Comprehensive Introduction". In: *Natural Computing: An International Journal* 1.1 (May 2002), pp. 3–52. ISSN: 1567-7818. DOI: 10.1023/A:1015059928466. URL: https://doi-org.tudelft.idm.oclc.org/10.1023/A:1015059928466.

[12] Annu Lambora, Kunal Gupta, and Kriti Chopra. "Genetic Algorithm- A Literature Review". In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019, pp. 380–384. DOI: 10.1109/COMITCon.2019.8862255.

[13] Annu Lambora, Kunal Gupta, and Kriti Chopra. "Genetic Algorithm- A Literature Review". In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019, pp. 380–384. DOI: 10.1109/COMITCon.2019.8862255.

[14] Fakhri Karray and Clarence De Silva. *Soft Computing and Tools of Intelligent Systems Design: Theory and Applications*. Pearson Addison Wesley, 2004. ISBN: 0321116178.

[15] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization". In: *IEEE Computational Intelligence Magazine* 1.4 (2006), pp. 28–39. DOI: 10.1109/MCI.2006.329691.

[16] Dervis Karaboga and Bahriye Basturk. "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems". In: vol. 4529. Jan. 2007, pp. 789–798. ISBN: 978-3-540-72917-4. DOI: 10.1007/978-3-540-72950-1_77.

[17] Iztok Fister jr et al. "Bat algorithm: Recent advances". In: *CINTI 2014 - 15th IEEE International Symposium on Computational Intelligence and Informatics, Proceedings* (Jan. 2014), pp. 163–167. DOI: 10.1109/CINTI.2014.7028669.

[18] A.S. Joshi et al. "Cuckoo Search Optimization- A Review". In: *Materials Today: Proceedings* 4 (Jan. 2017), pp. 7262–7269. DOI: 10.1016/j.matpr.2017.07.055.

[19] Nur Johari et al. "Firefly Algorithm for Optimization Problem". In: *Applied Mechanics and Materials* 421 (Apr. 2013). DOI: 10.4028/www.scientific.net/AMM.421.512.

[20] Mehar-un-Nisa Khursheed et al. "Review of Flower Pollination Algorithm: Applications and Variants". In: *2020 International Conference on Engineering and Emerging Technologies (ICEET)*. 2020, pp. 1–6. DOI: 10.1109/ICEET48479.2020.9048215.

[21] Haiqiang Liu et al. "An Intelligent Grey Wolf Optimizer Algorithm for Distributed Compressed Sensing". In: *Computational Intelligence and Neuroscience* 2018 (Jan. 2018), pp. 1–10. DOI: 10.1155/2018/1723191.

[22] Asaju La'aro Bolaji et al. "A comprehensive review: Krill Herd algorithm (KH) and its applications". In: *Applied Soft Computing* 49 (2016), pp. 437–446. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2016.08.041. URL: https://www.sciencedirect.com/science/article/pii/S1568494616304355.

[23] M.-H. Tayarani-N. and M.-R. Akbarzadeh-T. "Magnetic-inspired optimization algorithms: Operators and structures". In: *Swarm and Evolutionary Computation* 19 (2014), pp. 82–101. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2014.06.004. URL: https://www.sciencedirect.com/science/article/pii/S2210650214000509.

[24] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[25] Changseok Bae et al. "A new simplified swarm optimization (SSO) using exchange local search scheme". In: *International Journal of Innovative Computing, Information and Control* 8 (June 2012).

[26] Seyedali Mirjalili and Andrew Lewis. "The Whale Optimization Algorithm". In: *Advances in Engineering Software* 95 (2016), pp. 51–67. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2016.01.008. URL: https://www.sciencedirect.com/science/article/pii/S0965997816300163.

[27] Santosh Kumar, Deepanwita Datta, and Sanjay Singh. "Black Hole Algorithm and Its Applications". In: *Studies in Computational Intelligence* 575 (Dec. 2015), pp. 147–170. DOI: 10.1007/978-3-319-11017-2_7.

[28] Ghaith Jaradat and Masri Ayob. "Big Bang-Big Crunch optimization algorithm to solve the course timetabling problem". In: Nov. 2010, pp. 1448–1452. DOI: 10.1109/ISDA.2010.5687114.

[29] G.M. Qubati, Richard Formato, and Nihad Dib. "Antenna benchmark performance and array synthesis using central force optimisation". In: *Microwaves, Antennas Propagation, IET* 4 (June 2010), pp. 583–592. DOI: 10.1049/iet-map.2009.0147.

[30] Enrico Guiraud, Jakob Drefs, and Jörg Lücke. "Evolutionary Expectation Maximization". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '18. Kyoto, Japan: Association for Computing Machinery, 2018, pp. 442–449. ISBN: 9781450356183. DOI: 10.1145/3205455.3205588. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/3205455.3205588.

[31] Esmat Rashedi, Elaheh Rashedi, and Hossein Nezamabadi-pour. "A comprehensive survey on gravitational search algorithm". In: *Swarm and Evolutionary Computation* 41 (2018), pp. 141–158. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2018.02.018. URL: https://www.sciencedirect.com/science/article/pii/S2210650217303577.

[32] Maoguo Gong et al. "Immune algorithm with orthogonal design based initialization, cloning, and selection for global optimization". In: *Knowledge and Information Systems* 25 (2009), pp. 523–549.

[33] Muhamad Lokman et al. "Multi-verse optimization based evolutionary programming technique for power scheduling in loss minimization scheme". In: *IAES International Journal of Artificial Intelligence (IJ-AI)* 8 (Sept. 2019), pp. 292–298. DOI: 10.11591/ijai.v8.i3.pp292-298.

[34] Habib Youssef, Sadiq M. Sait, and Hakim Adiche. "Evolutionary algorithms, simulated annealing and tabu search: a comparative study". In: *Engineering Applications of Artificial Intelligence* 14.2 (2001), pp. 167–181. ISSN: 0952-1976. DOI: https://doi.org/10.1016/S0952-1976(00)00065-8. URL: https://www.sciencedirect.com/science/article/pii/S0952197600000658.

[35] Seyedali Mirjalili. "SCA: A Sine Cosine Algorithm for solving optimization problems". In: *Knowledge-Based Systems* 96 (2016), pp. 120–133. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2015.12.022. URL: https://www.sciencedirect.com/science/article/pii/S0950705115005043.

[36] Boris Naujoks, Nicola Hochstrate, and Michael Emmerich. "Multi-objective optimisation using S-metric selection: application to three-dimensional solution spaces". In: vol. 2. Oct. 2005, 1282–1289 Vol. 2. ISBN: 0-7803-9363-5. DOI: 10.1109/CEC.2005.1554838.

[37] Hamed Shah-Hosseini. "Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem. International Journal of Intelligent Computing and Cybernetics, 1, 193-212". In: *International Journal of Intelligent Computing and Cybernetics* 1 (June 2008), pp. 193–212. DOI: 10.1108/17563780810874717.

[38] James B.H. Kwa. "BS: An admissible bidirectional staged heuristic search algorithm". In: *Artificial Intelligence* 38.1 (1989), pp. 95–109. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(89)90069-6. URL: https://www.sciencedirect.com/science/article/pii/0004370289900696.

[39] Elyas Fadakar and Masoud Ebrahimi. "A new metaheuristic football game inspired algorithm". In: Mar. 2016, pp. 6–11. DOI: 10.1109/CSIEC.2016.7482120.

[40] Haoran Luo, Weidi Xu, and Ying Tan. "A Discrete Fireworks Algorithm for Solving Large-Scale Travel Salesman Problem". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, pp. 1–8. DOI: 10.1109/CEC.2018.8477992.

[41] Joong Hoon Kim. "Harmony Search Algorithm: A Unique Music-inspired Algorithm". In: *Procedia Engineering* 154 (2016). 12th International Conference on Hydroinformatics (HIC 2016) - Smart Water for the Future, pp. 1401–1405. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2016.07.510. URL: https://www.sciencedirect.com/science/article/pii/S1877705816318999.

[42] Marco Alberto Javarone and Daniele Marinazzo. "Evolutionary dynamics of group formation". In: *PLOS ONE* 12.11 (Nov. 2017), pp. 1–10. DOI: 10.1371/journal.pone.0187960. URL: https://doi.org/10.1371/journal.pone.0187960.

[43] Seyedmohsen Hosseini and Abdullah Al Khaled. "A survey on the Imperialist Competitive Algorithm metaheuristic: Implementation in engineering domain and directions for future research". In: *Applied Soft Computing* 24 (2014), pp. 1078–1094. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2014.08.024. URL: https://www.sciencedirect.com/science/article/pii/S1568494614003895.

[44] D.S.N.M. Rao and Niranjan Kumar. "Comparisional Investigation of Load Dispatch Solutions with TLBO". In: *International Journal of Electrical and Computer Engineering* 7 (Dec. 2017), pp. 3246–3253. DOI: 10.11591/ijece.v7i6.pp3246-3253.

[45] Roman Yampolskiy, Leif Ashby, and Lucas Hassan. "Wisdom of Artificial Crowds—A Metaheuristic Algorithm for Optimization". In: *Journal of Intelligent Learning Systems and Applications* 4 (Jan. 2012), pp. 98–107. DOI: 10.4236/jilsa.2012.42009.

[46] Thomas Bäck. *Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, 1996, pp. I–XII, 1–314. ISBN: 978-0-19-509971-3.

[47] André Restivo. "Dynamic Scenario Simulation Optimization". In: (Jan. 2006).

[48] *Cartesian coordinate system*. Apr. 2021. URL: https://en.wikipedia.org/wiki/Cartesian_coordinate_system.

[49] Iran Macedo. *Implementing Particle Swarm Optimization (PSO) Algorithm in Python*. Dec. 2018. URL: https://laptrinhx.com/implementing-particle-swarm-optimization-pso-algorithm-in-python-4085551428/.

[50] Dipayan Guha, Provas Roy, and Subrata Banerjee. "Load Frequency Control of Large Scale Power System using Quasi-Oppositional Grey Wolf Optimization Algorithm". In: *Engineering Science and Technology, an International Journal* 19 (July 2016). DOI: 10.1016/j.jestch.2016.07.004.

[51] thealimirjalili. *Mathematical models for the Grey Wolf Optimizer*. Oct. 2020. URL: https://www.youtube.com/watch?v=wA5FPPOYOKk&t=1099s.

[52] D. Wolpert and W. Macready. "No Free Lunch Theorems for Search". In: 1995.