COBURG UNIVERSITY
of applied sciences and arts

Faculty
Electrical Engineering and Computer Science

Term paper

Data Mining & Machine Learning

Introduction to Evolutionary Algorithms

Author: Lukas Höllein
lukashoellein@googlemail.com

Examiner: Markus Ring

Deadline: 27.06.2017

I Abstract

The goal of this seminar work is to provide readers with a broad overview about one component of machine learning in computer science, the so called evolutionary algorithms (EA).

Evolutionary algorithms are a core mechanism for developing artificial intelligence in computer systems today. Inspired by nature itself, it enables programs to evolve in certain ways. Its abilities allow it to be a relevant tool for many different aspects of software engineering.

Throughout this paper a core understanding of the principles and methods of EAs will be made comprehensible. Furthermore, readers shall gain the ability to recognize possible applications for EAs. It is not the goal to achieve full knowledge in every detail or the ability to create them practically. For further information on these issues, the literature in [1], [4] and [7] offer great starting points.

II Table of Contents

III Glossary

EA Evolutionary Algorithm

GA Genetic Algorithm

ES Evolution Strategies

EP Evolutionary Programming

GP Genetic Programming

MIT Massachusetts Institute of Technology

IV List of Figures

1 Motivation

Starting off with this seminar work, the first chapter will deal with the relevance of evolutionary algorithms. Declaring their usage and following possible achievements, a first grasp of EAs will be developed.

The following explanation is inspired by [1]. It provides an overview about the abilities of EAs.

All problems in computer science can roughly be broken down into two categories: P-problems and NP-problems. While the first ones are easy to solve with ordinary programming, the second ones usually fall into complexity classes of $O(n^2)$ or larger. As a result optimal solutions to these problems cannot be achieved in polynomial time, therefore making them practically insolvable. Nevertheless, todays technology is in need of providing answers to many of these problems. If there was no solution, many advanced services that are available to society, like navigation, would not operate. One way to achieve solutions can be through approximation. In many cases it is sufficient to get results that are close to being optimal. In these situations EAs can be applied.

As an example of NP-problems solvable through EAs, the "Travelling Salesman Problem" will be introduced. A salesman has to visit 20 places without visiting a place twice and is of course interested in taking the shortest possible route. Figure 1 presents this problem.

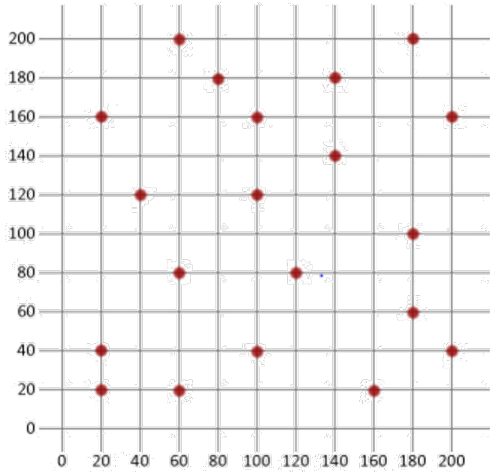

Figure 1: Travelling Salesman: example of 20 places to visit (taken from [2])

It is obvious that there are 20! possible solutions to this problem. Going through all of them requires too much time. However, navigation services like this exist already, so there must be solutions to finding good travelling routes in short time. Exactly this can be reached by the usage of evolutionary algorithms.

2 Getting Started: Overall View

What are the core principles of evolutionary algorithms? Into which class of algorithms can EAs be grouped? When can EAs be applied? Why are they called "evolutionary"?

All these questions will be answered in the following. Without explaining how they work, a classification of evolutionary algorithms will be developed throughout this chapter.

2.1 Characteristics and Behavior

Evolutionary algorithms are typically defined as searching algorithms [7]. They search a given domain of possibilites for optimal solutions. Thus, they can also be classified as a strategy to deal with optimization problems [1]. In [7] EAs are compared with finding peaks in multidimensional functions. All possible combinations of input variables to the function model, the domain of solutions and their output value can be viewed as measure of success: a higher output stands for a more optimal solution. This is visualized in figure 2.

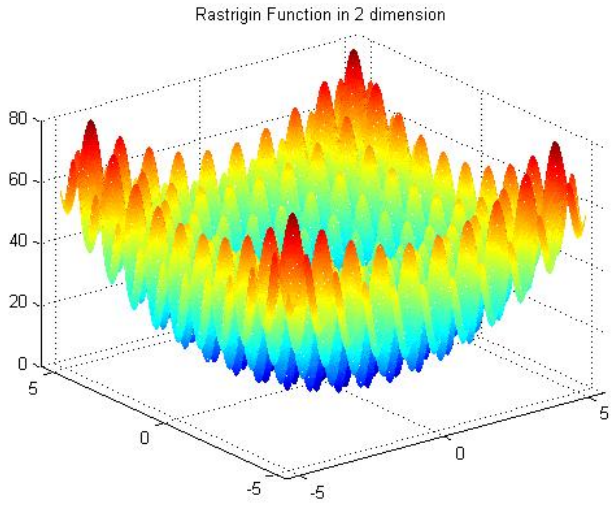

Figure 2: Comparison of EAs with multidimensional functions (taken from [5])

It is important to know that EAs usually will not produce the best solution for a given problem, instead they approximate and can find near optimal solutions. In figure 2 this can be viewed as finding input combinations resulting near to the global maximum, but never the exact values for the global maximum itself.

Another important aspect is that no input combination is wrong. It can only be very suboptimal for solving the problem. The global minimum in 2 would be such an example.

Whenever a domain exists that is characterized as described above, evolutionary algorithms can produce solutions. They become most useful when finding optimal solutions is not possible in polynomial time and approximations are a sufficient outcome.

Evolutionary algorithms exhibit two central characteristics while they are running [1]:

Generate and Test Evolutionary algorithms run in a loop. In each iteration a set of new solutions is produced and the previous set is discarded. The loop terminates when a predefined condition is raised.

Anytime Behavior Solutions grow better with time yet every solution is valid for solving the problem. While producing solutions non-deterministically, those in later iterations exceed the quality of previous solutions. In a manner of speaking they "evolve".

The most important trait of evolutionary algorithms is their origin [7]. Influenced by Charles Darwin and his evolutionary theory, they mimic evolution by nature and operate after the paradigm "Survival of the fittest". They copy many concepts and terms developed in Darwins' theory. According to [1] this is the key to why evolutionary algorithms are so successful in finding optimal solutions to hard problems in a short time.

2.2 Concepts and Terms

As stated in the last chapter, EAs operate on the paradigm "Survival of the fittest". Therefore, a number of concepts and terms from Darwins' evolutionary theory can be found in EAs in an adapted way. In the subsequent listing the most important ones are introduced. This is necessary in order to understand the whole concept of evolutionary algorithms. The terms will be used in successive explanations. All terms listed are explained in much more detail in [1] and [7].

Solution/Individual A solution, or in biological terms an individual, is one combination of input parameters that is solving the problem defined through a domain of inputs.

Population/Generation A population or generation is the set of all individuals the algorithm currently knows. It changes in each iteration and is the main data on which the algorithm operates.

Fitness Each individual can be measured by a fitness-function which allows comparison between different individuals. The higher the value of the fitness-function the "fitter" is one individual.

Recombination In the recombination process the new population is created on the basis of the old population. Old individuals, the parents, are "recombined" to new

individuals, the offspring.

Probability of Recombination Each individual holds a probability of recombination stating how likely it is to be chosen for recombination. Usually, the fitness is the central element for constructing the probability.

Selection In the selection process individuals are either kept for the new population or discarded. In a manner of speaking individuals can “survive“ or “die“ through selection.

Probability of Selection Each individual holds a probability of selection stating how likely it is to be chosen as a part of the new population. Usually, the fitness is the central element for constructing the probability.

It is important to note that the probability of recombination and the probability of selection are not the same: when a parent is chosen for recombination, it does not necessarily mean it is part of the next population as well.

Aswell, a high fitness value does not necessarily mean that the individual will be chosen for recombination and selection. It is more likely, but there is no guaranty. This mimics natural phenomenons where the most fittest individuals might still become extinct under certain circumstances.

Another important concept adapted from nature is the storage of information as genome or chromosome [1]. All information is stored in one or many independent chromosomes which can all be referenced. There are many ways to store information in chromosomes, which are listed in [1] and [7]. For further explanation in this seminar work, the so called binary representation will be chosen. In it, all information is expressed as a chain of zeroes or ones with fixed length. One chromosome is then equal to one chain of that kind. Each bit inside of that binary string has a predefined purpose. Figure 3 visualizes this definition.

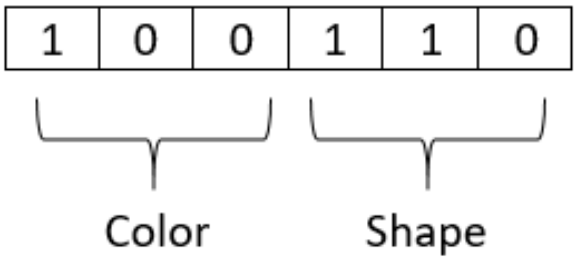

Figure 3: Binary representation of one chromosome

In this figure another term related to EAs can be seen: genotype to phenotype conversion. The genotype represents information at its encoded level, for example as a binary

representation of a chromosome. On the contrary, the phenotype represents information at a visible level [7]. In figure 3 this would be the color and shape of whatever object is expressed by this chromosome.

Having gained this knowledge, the process of recombination needs some further explanation. Evolutionary algorithms offer several ways to execute recombination of parents [7]. The two most important ones are explained in the following listing. Note that, again, both alternatives offer many different kinds of implementation. For further insight on these, chapter 4 may be advised.

Crossover Crossover is a process where offspring is created by recombining binary strings of p chromosomes of $2-n$ parents at its genotype representation to p new chromosomes for $1-n$ offspring. Figure 4 illustrates this example for $p = 1$ and two parents and offspring.

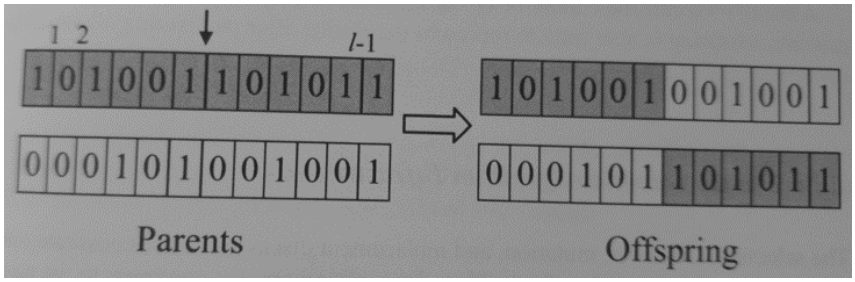

Figure 4: Crossover of two chromomsomes to two new chromosomes (taken from [7])

Mutation Mutation is a process where offspring is created by flipping $1-n$ bits of the binary string of p chromosomes. Figure 5 illustrates this for $p = 1$ and one bit to be flipped.

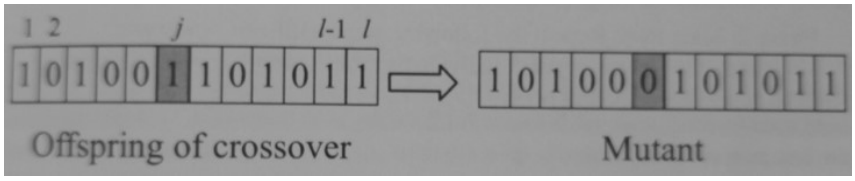

Figure 5: Mutation of one chromosome (taken from [7])

Having named the most important terms in evolutionary algorithms, chapter 2 concludes. All characteristics and concepts introduced are consecutively used as foundation for further explanations.

3 The Evolutionary Algorithm

Having established a core understanding for evolutionary algorithms, the goal of this chapter is to connect all previously introduced parts and use them to build the general process of any EA. In figure 6 this is illustrated.

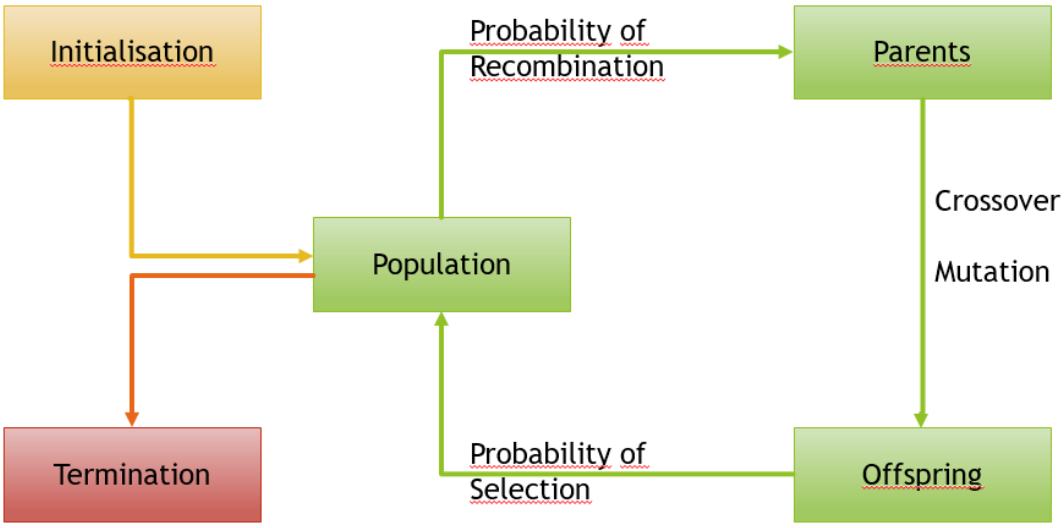

Figure 6: General process of any EA (inspired by [1])

Every EA can be described as combination of steps and transitions. Inspired by [7], all the steps and transitions of figure 6 can be explained as followed. Once more, chapter 4 is advised to be visited for further information on these components, as multiple implementations exist for every component.

Initialisation Every EA starts with building the first generation of individuals. Having constructed it anyhow, an EA can enter its main loop.

Population This state is the starting point of the main loop. If the population has not reached any condition for termination, the EA proceeds. Choosing n individuals from the population by using their probabilities of recombination, this state transists to its successor.

Parents Having gained n individuals, recombinations are operated on them. Predefined ways of recombination like crossover or mutation are performed until a fixed number of offspring is created.

Offspring All previously created offspring will be evaluated respectively to determine their fitness value. After that, the offspring and all individuals of the current population are chosen with their probability of selection to be part of the next generation.

Having concluded the selection, this state transists back to the starting point of the loop.

Termination If the population has reached any condition for termination, the main loop will be suspended. The last population is the resulting set of solutions gained by the EA.

All EAs share this identical structure, but differ in their implementation. When running an EA, the process shown above will sequentially produce better solutions for a given problem. As previously described, the "Anytime Behavior" is a central element of EAs, which is realized by using the structure of figure 6.

Experience has proven that every problem suited to be solved through EAs needs to be evaluated on key traits. Using that knowledge, the best suited implementations for each component, transition and even some terms like the probability of recombination need to be chosen respectively. This complex procedure is described in more detail in [1] and [7].

4 Diving Deeper: Implementation Details

This chapter is supposed to provide readers with a broad grasp about implementing evolutionary algorithms. Knowledge about different possibilities and general constraints of EAs is developed subsequently. It is noted, that these tasks constitute the main parts of [1] and [7]. Therefore, only an oversimplification can be provided in this seminar work.

4.1 Parameters and Coefficients

As previously stated, the general process of EAs as illustrated in 6 can be implemented in many ways. Over the years the following main types of evolutionary algorithms have emerged [1]:

- Genetic Algorithms (GA)
- Evolution Strategies (ES)
- Evolutionary Programming (EP)
- Genetic Programming (GP)
- Direction-based Search
- Differential Evolution (as seen in figure 2)

These types differ in implementations of specific states and transitions of the general EA processs illustrated in figure 6. They do not differ in the process itself.

In the following, example implementations for selected states and transitions will be conducted. All examples are taken from [7] and can be looked up there.

Probability of Recombination/Selection As stated in chapter 2, the fitness value f_i of individual i is usually used to construct its probabilities of recombination and selection. The simplest way to determine them is by constructing a relative fitness value p_i:

$$p_i = \frac{f_i}{\sum population f_i}$$

This can be visualised as seen in figure 7:

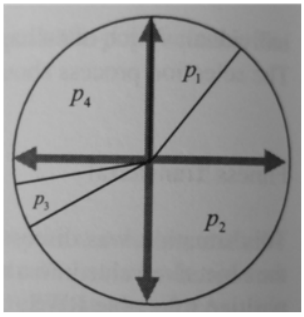

Figure 7: Visualisation of relative fitness values (taken from [7])

The parent selection and survivor selection can then be thought of as spinning a wheel with multiple sections as seen in figure 7. The section that is selected by the spinning process is then equal to its individual being selected for recombination or selection. It is obvious that fitter individuals are more likely to be chosen, but they might still be ignored. This concept is necessary to avoid premature convergence, a constraint of evolutionary algorithms that is introduced in chapter 4.2.

Another more refined way of constructing the previously mentioned probabilites is through so-called sigma truncation. The absolute fitness value f_i of an individual i is adapted in the following way prior to the calculation of probabilities by determination of relative fitness values for each individual.

$$f_i' = f_i - (f_{avg} - c * \sigma)$$

Sigma truncation scales the absolute fitness value with consideration of the standard deviation σ and the average fitness value f_{avg} of all fitness values in the population. The parameter c is a value between $[1, 5]$. The sigma truncation method can be analyzed on how it influences the recombination and selection processes respectively

to the number of iterations passed in one EA run:

- When the EA has just started, f_{avg} is relatively small and σ is relatively large. Therefore, a small number is subtracted or even added to the original absolute fitness value. This decreases selective pressure making it more likely for less fitter individuals to survive.
- When the EA has run for a while, f_{avg} is relatively large and σ is relatively small. Therefore, a large number is subtracted from the original absolute fitness value. This increases selective pressure leading to elimination of less fitter individuals.

Aside from sigma truncation, many advanced concepts exist, each of which influences the recombination and selections processes to push them in certain directions. Usually, it is necessary to analyze EAs while they are running and as a result performing a switch between different methods for an optimal outcome of EAs ([1] and [6]).

Selection Apart of the probability of selection, the definition of the space of individuals on which selection is operated is a method to implement EAs. The two most famous ways of implementing this space are the following:

- The new population is completely defined through all offspring created in recombination. All parents are discarded. This method is used by "Genetic Algorithms".
- The new population is defined by choosing the best μ individuals of μ parents and λ offspring. This method is used by "Evolution Strategies" and "Evolutionary Programming".

Recombination The recombination tools "Crossover" and "Mutation" can be utilized to execute recombination. While both are used by GAs and ESs, only mutation is used by EPs. On top of that, crossover and mutation exist in different variation. The following listing names the most common ones.

Crossover Single-point crossover (performed in figure 4), Multiple-point crossover

Mutation Bitflip-mutation (performed in figure 5), Uniform mutation, Normal mutation

Initialization Constructing the first population can either be performed through random generation of individuals or through gaining pre-optimized values with other methods than EAs.

Termination A termination is reached when either a fixed number of iteration and therefore a fixed number of generations is reached or when one individual in a population exceeds a fixed treshold of fitness.

When creating an EA, choosing implementation methods for each state and transition of the general EA process is the most work-intensive and important part ([1]). Most of the time, a general-purpose implementation of EAs is not sufficient to solve a specific problem, making construction of EAs time-intensive [7]. In [1] this characteristic is described as the "Tuning Problem".

4.2 Constraints

When EAs are finally constructed, it is interesting to analyze how they behave while they are running. Common constraints for the usage of EAs exist, which will be introduced in the following chapter. They are described in more detail in [1].

Exploration and Exploitation All EAs usually start by exploring unknown regions of values. After that, this region is exploited and the best values inside that space are determined. Choosing different parameters can influence whether exploration or exploitation is performed by an EA. In [6] MIT Professor Patrick Winston at first lets his EA explore different regions and exploit them individually. Eventually, the global maximum is found and approximated by the EA.

Premature Convergence This is the most common drawback of all evolutionary algorithms. When the configuration of parameters and coefficients is insufficient the EA might not find the global maximum. Instead, it will find a local optimum and exploit it as explained previously. It then never breaks out of the region of the local optimum. The final outcome of the EA can therefore be far away from the actual optimal solution.

Looking at statistics from [1], the following figures 8 and 9 explain the progress and suitability of EAs.

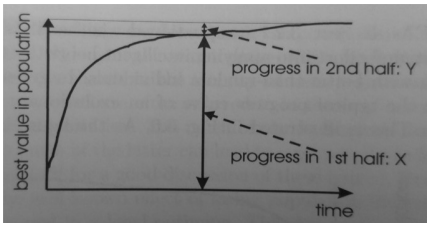

Figure 8: Visualisation of fitness values in progress of EAs (taken from [7])

In figure 8 it can be seen that 90% of the optimization in EAs happens in 50% of the time

they are running. This leads to the consideration of how optimized solutions need to be for specific problems. Having less sufficient solutions can greatly shorten the runtime of EAs.

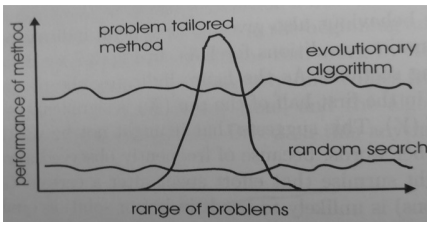

Figure 9: Comparison of EAs with other methods (taken from [7])

In figure 9 the previously mentioned thesis that EAs need to be generated for each problem respectively instead of building one general-purpose EA is confirmed.

Having mentioned some constraints of EAs, chapter 4 concludes.

5 Combination with Neural Networks

Before concluding with this seminar work, this prospect of another usage of EAs will be given. In [4] the combination of EAs with neural networks, so-called neuroevolution, is comprehensively explained. It shall be mentioned, that a combination of these methods can result in astonishing outcomes and many systems using artifical intelligence today rely on a combination of these two tactics.

In figure 10 the general concept of neuroevolution is visualized.

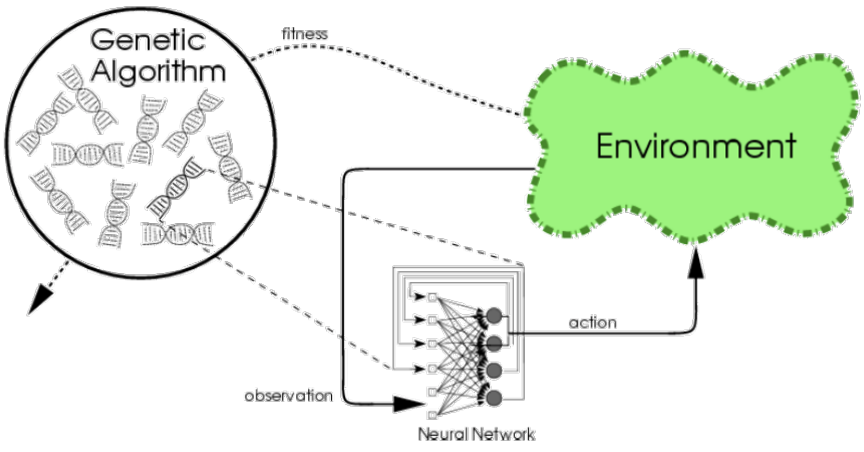

Figure 10: General concept of neuroevolution (taken from [3])

Evolutionary algorithms - in figure 10 GAs were chosen - are used to train neural networks to perform better in their environment. All general concepts like fitness, recombination

and selection are applied on neural networks. Therefore, it must be possible to define a domain of input variables and solutions as described in chapter 2. Interested readers are advised to look into [4] for further information on neuroevolution. In this seminar work, it is adequate to mention it to name its relevance in todays machine learning sector.

6 Summary

The purpose of this chapter is to give a concluding view on EAs as a technique in machine learning.

Looking at examples of EA implementations can lead to being very amazed of how well they perform. Watching EAs operate over time can look astonishing and mystifying. As well, the term “evolutionary“ might leed to the notion that this technique is very refined and perfected - after all it mimics nature which must be good. According to [6] where MIT Professor Patrick Winston told his oppinion on EAs, this view is not suitable for describing them. Instead, he suggests to be amazed of how many possible solutions in a defined domain of inputs exist that can solve a specific task. The achievment of EAs is then to find good-suited solutions quickly. Looking at it this way, a more professional evaluation of evolutionary algorithms is possible.

In the following listing the most important traits of EAs are named again.

1. Evolutionary algorithms are an efficient tool for searching approximations to optimal solutions in a given domain of input values and outputs.
2. Evolutionary algorithms are time-extensive to be configured. Otherwise, they might not produce sufficient solutions to given problems.
3. They are most relevant in the field of machine learning and used in many advanced applications today.
4. In the field of neuroevolution, evolutionary algorithms are an essential component.

Having given this summary, the seminar work concludes. For further information on evolutionary algorithms, the literature mentioned in chapter I offer great starting points.

7 Bibliography

[1] Eiben, A.E., Smith, J.E., et al.: Introduction to evolutionary computing, vol. 53, 2 edn. Springer (2003/2015)

[2] Jacobson, L.: Applying a genetic algorithm to the traveling salesman problem. URL http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5

[3] Mahjourian, R.: Neuroevolution. URL https://github.com/nnrg/opennero/wiki/NeuroEvolution

[4] Sher, G.I.: Handbook of neuroevolution through Erlang. Springer Science & Business Media (2012)

[5] Wang, M.: Optimization rastrigin function by differential evolution algorithm. URL https://de.mathworks.com/matlabcentral/fileexchange/46818-optimization-rastrigin-function-by-differential-evolution-algorithm?requestedDomain=www.mathworks.com

[6] Winston, P.: 13. learning: Genetic algorithms. URL https://www.youtube.com/watch?v=kHyNqSnzP8Y

[7] Yu, X., Gen, M.: Introduction to evolutionary algorithms. Springer Science & Business Media (2010)

Erklärung

Hiermit versichere ich, dass ich meine Seminararbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: 15.06.2017

(Unterschrift)