# Artificial Immune Systems for Solving the Traveling Salesman Problem

A use of CLONALG in Pathfinding and Optimization

**Bachelor Thesis**

FH Campus Wien

**Vorgelegt von:**
Sebastian Ukleja

**Personenkennzeichen**
XXXXXXXXXXX

**Abgabe am:**
XX.YY.20ZZ

# Abstract

Artificial immune systems are together with neural nets and genetic algorithms an important part of the bio-inspired approaches in machine learning. This thesis will examine artificial immune systems and their performance in solving the traveling salesman problem. The CLONALG algorithm will be compared to a more conventional heuristic greedy algorithm. The results will show that the clonal algorithms can achieve better results under certain circumstances but are in their general form not as efficient as the heuristic algorithm. The CLONALG and most clonal selection algorithm work with static parameters. This thesis examines also if parameter control is beneficial in solving the traveling salesman problem and shows that dynamic adaptation of certain parameters during runtime, can enhance the performance of the algorithm.

# Abbrevations

| | |
|---|---|
| BIS | Biological Immune System |
| AIS | Artificial Immune System |
| Ab | Antibody |
| Ag | Antigen |
| TSP | Traveling Salesman Problem |

## Keywords

Artificial Immune System
Clonal Selection Algorithm
Negative Selection
Traveling Salesman Problem
Bio-inspired approach
AIS
Machine learning
Optimization

# Contents

# Chapter 1

# Introduction

Bio-inspired approaches in machine learning mimic biological functions and concepts like the information exchange of neurons in the human brain or the evolution of genes through selection. The biological system often has very efficient methods for learning and adaptation therefore the bio-inspired approaches were very successful over the past years in the field of machine learning. The immune system got more attention in the computational intelligence field recently over the last 20 years [Tan16].

The biological immune system has a number of desireable traits for learning in computational science as mentioned in Chapter 2. These traits can be used in algorithms to build an artificial immune system for solving a wide range of problems as shown in Chapter 3. The artificial immune system can be applied to similar fields like neural nets or genetic algorithms but are also efficient in combination with those algorithms [PC06]. One of these problem fields is the traveling salesman problem. The artificial immune system was successfully applied to the TSP by [CZ02] and [Sun+04]. In this thesis the overall performance of the AIS in solving the TSP will be examined especially compared to a more conventional method in solving them. The artificial immune system works with different approaches taken from the biological immune system. One of these approaches is the clonal selection which was formalized into an algorithm by [CZ02]. In Chapter 5 this algorithm called CLONALG will be applied to a set of TSP. The CLONALG is originally a static algorithm with fixed parameters. The work of [RM09] and [Gar04] showed that in dynamically changing these parameters, the performance of the algorithm can be enhanced. In Chapter 5 a tuned CLONALG and a variant with dynamic parameter control will be applied to the same set of TSP and evaluated.

The goal is to examine if the AIS is able to achieve good results in solving the TSP and especially under which circumstances and what parameters within a clonal selection algorithm are responsible for a good performance.

# Chapter 2

# Biological Immune System

A biological immune system (BIS) has many features that are useful in machine learning. It can be described with the following terms [Tan16]:

- Distributed
- Parallel
- Multi-level
- Distinguishes between self and non-self
- Noise resistant
- Self-organized
- Associative memory

It is distributed because there is no need for a central control instance, the BIS acts where it need to and it does so immediately without supervising. It can operate on multiple parts of the body simultaneously thus, it is parallel. The BIS works in different levels. At first there is the physical barrier, the skin and the body fluids. If this level fails the innate immune system responds, which is a pre-programmed immune reaction that can respond to known and non-changing threats. Finally, there is the adaptive immune system if all of the previous levels fail. This is the system most interesting for computational science, because of its ability to learn on its own. A very important ability of the BIS is the distinguishing between any self-element of the body and the non-self or potential threat. It is noise resistant because it can react to variations of known threats. The BIS does not need any supervising through the brain or any other central system to organize its work flow, therefore its self-organized. Finally, it has an associative memory which is used to react on similar and variant of threats already encountered. As mentioned above, the adaptive immune system is the most interesting part of the BIS for modelling an artificial immune system (AIS). The immune system uses many principles to be effective, in terms of computational science the principles of negative selection and clonal selection are specifically interesting.

## 2.1 Negative Selection

The adaptive immune system uses two kind of lymphocytes to counter a threat. The T-Lymphocyte and the B-Lymphocyte. A Lymphocyte is a variation of a white blood cell [Jan01]. Both have different roles but both lymphocytes must have the ability to distinguish between the self and the non-self-cells in the body. A fault in this system can not only lead to an infection, it could trigger an autoimmune reaction because a self-cell could be identified as a non-self threat. To avoid this the Lymphocytes are generated through the process of negative selection. While the B-Cells will be developed in the bone marrow and the T-Cells in the thymus the process is exactly the same. Both cell types are presented to a wide range of self-cells. If any of them react to such a self-cell, the Lymphocyte will be killed and another will be generated. This process is repeated till there are only B- and T-Cells which react to non-self cells [Tan16]. This process will be imitated in the AIS in generating detector sets.

After that the lymphocytes will be released into the tissue and the blood system. The B-cells are called antibodies (Ab) in this context, and any non-self cell is called an antigen (Ag). The B-cell has the ability to recognize and dock to a specific Ag. The ability to recognize an Ag is called affinity. If an Ab has a high affinity to an Ag it is especially good in recognizing and countering this Ag. Because the generation of Lymphocytes and therefore the affinity to different Ag's is random, additional measures are necessary to improve effectiveness.

## 2.2 Clonal Selection

If a B-Cell encounters an antibody it is able to proliferate (divide) into multiple terminal cells which are clones of the cell. The cell is not only cloned, the different clones will be mutated to improve the affinity to the antigen. The process of cloning and mutating will be repeated till a population and affinity threshold is reached which ensures the most efficient response to the Ag. Only the Ab's with the highest affinity score will be cloned and mutated. If the affinity is high enough, the B-Cell can proliferate into memory cells which will stay after the response and ensure that a secondary response to a similar Ag will be much faster than the initial one [CZ02]. This memory cells represent the associative memory of the adaptive immune system.

The principles of negative selection and clonal selection are important concepts in designing an artificial immune system. The clonal selection aspect of the BIS is basically the learning system. The cloning process is a form of reinforcement learning and leads to a continues improvement [CZ02]. The mutation process itself is called affinity maturation. Random changes in the genes lead to changes in affinity in every single clone. The mutation process is inverse to the affinity level. A higher affinity level means a lower mutation rate [CZ02]. Clonal selection is based on the basic evolutionary theory of Charles Darwin.

The three basic principles are [CZ02]:

- repertoire diversity (high population of Ab's)
- genetic variation (random changes to the population (blind variation))
- natural selection (high affinity Ab's will reproduce and maintained)

These are high level abstract concepts and are only used for a very brief overview of the immune system. The BIS is far more complex but the details are out of scope of this thesis.

# Chapter 3

# Artificial Immune System

## 3.1 Basics of an AIS

A definition of an AIS is given by [Tan16, p. 5]: "Artificial Immune System (AIS) is a computational intelligence system inspired by the working mechanism and principles of the biological immune system."
An AIS can be used in machine learning and is comparable to an artificial neural net in terms of different application fields. It shares some similarities with genetic algorithms due to the cloning and mutation process in clonal selection.

Generally speaking: a set of detectors (Ab) react to a specific anomaly (Ag). This anomaly could be a malicious code, an IP address and port combination that is not allowed in the network or a pattern that has to be classified. What represents an antigen in the algorithm completely depends on the context and the use of the algorithm. An AIS can be used in an Intrusion Detection System [Pam17], learning and pattern recognition [CZ02], for recommender systems, data mining and clustering [BK13] and optimization [Nan+08]. Control engineering, fault diagnosis and robotics are also application fields of an AIS [Tan16]

The basic steps of an AIS can be summarized as shown in [Tan16]:

1. Initialize/present antigen
2. Initialize antibody population
3. Calculate affinity for each antibody to the antigen
4. Check life cycle of each antibody and update it
5. If stopping condition met go to 6 else go to 3
6. Output antibody population

Step 2 and Step 4 are the steps where negative selection and clonal selection will be relevant in most forms of AIS algorithms.

At the time of writing there are mainly 5 concepts that are used in an AIS. The aforementioned negative selection, clonal selection and more recently the immune network theory, the danger theory and the dendritic cells theory.

## 3.2 Negative Selection in AIS

A typical AIS has a set of detectors which react to any non-self data of the system. The most common way to generate these detectors is to create them randomly and let them undergo a negative selection in which they are presented to self data sets. If one of the detectors recognizes a self data, therefore the affinity is high enough, it will be destroyed. In this case the AIS follows the process that is found in the BIS.

It is important to decide in which way affinity will be measured and which conditions will trigger an immune reaction. If the data that is used consists of numerical values, or is easily converted into a vector of those, the Euclidean distance [4] is commonly used as a mean to measure affinity [TAN16]:

$$\sqrt{\sum_{i=1}^{n}(xi - yi)^2}$$

The Euclidean distance gives us the straight-line distance between two points in an Euclidean space. In the formula above xi and yi are values from two n-dimensional vector x and y [AR94].

Another possibility is the use of different variants of the Hammington distance. The Hammington distance is the number of bits that must be changed in two bit strings of the same length to make both strings identical [Rob08]. In an binary string the distance can be measured by the number of ones after applying a XOR operation on both strings. A common variant of this is the use of the so called r-continues bits. In this case a detector recognizes an element if r continues bits are identical with the element.

| | |
|---|---|
| Non-self element | 0010110 |
| Detector | 0011101 |

In the example above the first 3 continues bit match in both strings. If the threshold of the detector is r = 3, it would react to this bit string if the number of continues bits is 3 or higher. In case of recommender systems another way to measure the affinity is to use the Pearson product-moment correlation [PG95]. But this correlation is not suited for optimization tasks or specifically the use in the travelling salesman problem.

## 3.3 Clonal Selection in an AIS

Negative Selection is good for generating a population of detectors. To mimic the learning abilities of the biological immune system, the clonal selection principle can be used. The CLONALG algorithm was proposed by [CZ02] in 2002 and is the foundation of many AIS algorithms that use the clonal selection principle. This thesis will focus on the CLONALG algorithm and some of its variations, as these algorithms will be used to generate the data in Chapter 5.

### 3.3.1 CLONALG

The CLONALG algorithm mimics the clonal selection of the BIS on an abstract level. It will be applied mainly at step 4 in the basic AIS sequence shown in Chapter 3. The algorithm starts with generating an initial population C. It then calculates the affinity of the population to an antigen which is called the fitness. The n best antibodies of the population C will be chosen, determined by a fixed fitness value and form the antibody set S. Every antibody in S will be cloned and represents our clone set P. Now the clone set will be mutated, every clone get randomly changed relatively to its affinity value. After the mutation the n best clones out of P are chosen. Then a new population will be generated and the process begins anew till a stopping condition is met.
An example of a pseudocode based on RM09 can be illustrated as followed: [RM09]:

```
Generate initial population C of A antibodies
Calculate Fitness(C)
while stopping condition not met do
    S= Select the n best antibodies from C
    P= Generate clones of the antibodies in S
    Mutate(P)
    C= Select the n best antibodies out of P
    C= C + New population A-n
end
```

**Algorithm 1:** Simple CLONALG pseudo code

Usually the stopping condition is a given amount of evaluation where no increase in fitness is achieved or felt below a given threshold. The initial CLONALG algorithm as proposed by [CZ02] operates with static parameters and does not adjust anything besides the mutation. Different parameters are needed for different applications fields as also stated in [CZ02]. This parameters must be changed beforehand and can't adapt dynamically during the runtime of the algorithm.

Although the algorithm is very simple and efficient in solving different tasks like multimodal problems or pattern recognition, drawbacks do exist [Gar04]. Choosing how many member should be cloned is difficult. The lack of adaptive parameters can lead to inefficiency because of bad scalability and too many evaluations which could have been avoided [Gar04].

### 3.3.2 Adaptive CLONALG variants

To make the algorithm more efficient and adaptive, some variants where proposed. There are different parameter control strategies for the CLONALG algorithm. One of these variants is proposed by [RM09]. It is based on the idea of reinforcement learning. The antibody sets will be either rewarded for a high affinity or penalized for a low one. This will be achieved through population control. The reward is the increase in antibody population for a given set and the penalty is the decrease in population [RM09]. The mutation factor is governed by the population size. This adaptive technique allows the algorithm to adjust the core parameters during runtime and makes it more efficient in problem solving and hardware usage as shown in [RM09].
Another method is proposed by [Gar04]. Some techniques from the evolutionary algorithms will be applied to the CLONALG in this approach. An algorithm based on this idea will be evaluated in chapter 5.

# Chapter 4

# Traveling Salesman Problem

## 4.1 Background

One of the first publications of the traveling salesman problem was by the mathematician Karl Menger in the 1920's [App+07]. The problem itself was discussed earlier by Sir William Rowam Hamilton and Kirkman [MSM10]. It describes a graph with a certain amount of nodes with known distances between the nodes. The goal is to find a route where every node is visited exactly one time and the route ends at the node where it started. The route should be the shortest possible.((example picture of an tsp))

The algorithmic complexity for a symmetric graph with n nodes is: $\frac{n(n-1)}{2}$ [App+07]. The graph is symmetric if the distance between two nodes n and m is the same as the distance between m and n. Asymetric traveling salesman problems also exist with a higher complexity of $n(n-1)$
The complexity of the problem is categorized as non-deterministic polynomial hard (NP-Hard). The runtime of an algorithm can scale exponentially with the number of nodes in the graph. No algorithm is able to solve the problem in polynomial time.
Current algorithms work with heuristics to solve the TSP.

The TSP is common in many real world applications. Drilling of printed circuit boards, computer wiring, order picking in warehouses, vehicle routing and DNA sequencing are some fields where the TSP is present [MSM10].

## 4.2 Mathematical definition

The symmetric graph is defined as $G = (V, E)$ where $V = \{1, .., n\}$ are the vertex or the nodes and $E = \{(i, j) : i, j \in V, i < j\}$ are the edges or routes. Additionally there is an arc set $A = \{i, j) : i, j \in V, i \neq j\}$ which defines all routes in the graph, no route can be used twice. A cost matrix is defined on the edge of the arc set. Usually the cost matrix is calculated using the Euclidean distance [MSM10].

A typical TSP consists of a set of cities (V), distances between the cities (E) and the cost measured in the Euclidean distance between the cities (C). All TSP used in this thesis will follow this convention.

# Chapter 5

# Evaluation

## 5.1 Setup

The CLONALG algorithm will be applied to a set of 17 different TSP problems from the TSP library TSPLIB95[1]. The number after the names of the TSP is the amount of nodes and therefore indicates the difficulty of the TSP. The implementation follows the specification in [CZ02] and is provided by the Optimization Algorithm Toolkit OAT[2]. The greedy search algorithm will be applied to the same set of problems. This algorithm is provided by the author of the OAT and uses a nearest neighbour technique with mutation.

The stopping criteria for the algorithm will be no improvements after a set amount of evaluations. No improvement means that the algorithm has not found a shorter route in it's actual iteration compared to the last one, this is often also called stagnation. The criteria for the results are:

1. Score
2. Time in ms
3. Evaluations
4. Percentage of optimal score

Percentage of optimal score is the difference to the best possible route of the TSP. The score is measured as the summarized Euclidean distance of the presented best tour. The algorithm will be run multiple times on one TSP therefore the arithmetic mean of each criteria will be the end result. To compare the results the mean average error (MAE) of the average score will be used. The MAE is composed from the difference between the score of both algorithms divided through the score of the main compared algorithm, if we want to compare the performance of algorithm A1 to A2 the MAE is calculated as $(A2-A1)/A1$. Positive MAE means better performance for A1. The significance of the

[1]Rei, https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.
[2]Bro, http://optalgtoolkit.sourceforge.net/.

difference is visible at the digit. The thousands digit shows no significant difference. This was tested by running the complete test run twice on the same algorithm. The single test runs only had a difference visible on the thousands digit. Differences on the hundreds and tens digit are significant. The CLONALG algorithm will be applied to the problem set twice with different parameters. The adaptive algorithms will be applied only once because of the dynamic parameters. The parameters which will be altered are:

1. Population size
2. Clone factor
3. Selection size
4. Random replacements

The first set of parameters will be the default parameters provided by the OAT as shown in table 5.1. The second one are the parameters that are proposed by DeCastro [CZ02] for solving TSP problems about the size of 30 nodes. The adaptive variant of the CLONALG algorithm will use the default vaulues at the beginning and adjust the paramaters during runtime. To measure the results, a modified version of the OAT will be used. The changes are a slightly different set of TSP Problems used in the domain and the addition of an adaptive CLONALG hybrid algorithm. The used TSP problems are listed in the appendix. The distances between the nodes in the TSP problem are measured as Euclidean distance. The implemented CLONALG algorithm is based on the specifications in [CZ02]. The adaptive variant expand the concept based on [Gar04]. Both algorithms will be applied 100 times on every single TSP problem.

The hardware is a i5-3320M dual core CPU with 2,60ghz each and 8gb of RAM, run on a Windows 10 operating system with only the necessary windows background tasks. All CPU and RAM usage shown in this thesis is after other processes, not related to the algorithm, are taken into account.

### 5.1.1 Evolution Strategy Parameter Control

The first CLONALG variation is based on [Gar04] and uses an idea from the evolution strategy [BS02]. The original managed to eliminate all parameters except population size from the CLONALG algorithm. This thesis uses a variant where the selection size n is eliminated. This is comparable to Variant C1-C4 from [Gar04]. It uses a strategy parameter which will adjust the selection size. The strategy parameter itself will be adjusted by a evolution strategy constant of 1.3. This constant was empirically tested and proofed to be the most effective [Gar04]. The adjustment is randomized, the strategy parameter will be either multiplied or divided by the constant based on a 50% chance. The strategy parameter itself will alter the selection size on each evaluation. The evolution strategy calls this evolution approximated by self evolution, however the difference to the original evolution strategy is, that a parameter is indirectly adjusted through another parameter [Gar04]. Naturally the selection size still has to be initialized and can't be zero. [Gar04] proposed to start with a small number, the test runs in this thesis start with a selection size of 1. Another difference to the orignial CLONALG is the absence of random replacements. The population will only be altered by the clones and their mutations. The changes made to the original pseudo code in chapter 3 are shown in algorithm 2. This variant will be called CLONALG ESPC, short for Evolution Strategy Parameter Control.

```
Set strategy paramater Sp
Generate initial population C of A antibodies
Oc=Calculate Fitness(C)
while stopping criteria not met do
    S= Select the n best antibodies from C
    P= Generate clones of the antibodies in S
    Mutate(P)
    C= Select the n best antibodies out of P
    C= C + New population A-n
    Nc= Calculate Fitness(C)
    If Nc has better fitness than Oc then
    n= Sp * 1.3 OR n= Sp/1.3
    else
    n=n
    end if
    Oc=Nc
end
```

**Algorithm 2:** CLONALG variant with dynamic selection size

## 5.2 Results

### 5.2.1 CLONALG un-tuned

The algorithms are run 100 times on every TSP. The stopping criteria are no improvements after 10000 iterations of the algorithm. This number is chosen to give the algorithm enough time without restricting it to a specific amount of seconds. The CPU usage spiked around 58% and was average around 30% for all algorithms. This is because the more costly operations like sorting, selection, cloning and mutation will be done in exactly the same way in all algorithms.
The parameters for the original CLONALG are shown in Table 5.1.
The parameter C is the initial population, n is the selection size for cloning, B is the cloning factor (how many clones of the chosen n will be done) and d are the random replacements to keep the new population partially randomized. [CZ02] proposed some tuning to the default parameters for more efficiency in solving TSP. The replacement value d should always be between 5-20% of the population. Higher ratios tend to randomize too much while lower ratios can still produce good results but less reliable and efficient as shown in [CZ02].

| | C | n | B | d |
|---|---|---|---|---|
| Default | 50 | 50 | 0.1 | 5 |
| Tuned | 300 | 150 | 2.0 | 60 |

Table 5.1: Tuning parameters

When comparing the results it is clearly seen that the CLONALG algorithm is worse on average than the greedy search algorithm as shown in table 5.2. However the CLONALG was able to find the best solution to the TSP ulysses22 with a MAE of 0.117 while the greedy search algorithm could not as shown in figure 5.1. The overall time for solving all problems was also shorter for the CLONALG. The number of evaluations is mostly smaller for the CLONALG which shows that stagnation started earlier in this algorithm. The average MAE for all TSP compared to the greedy algorithm is -0.524. The average MAE for all TSP calculated on the best score is in the same range with -0.546. The mean average error of -0.009 is not significant for the TSP berlin52 which highlights that CLONALG performs equal or better on small TSP under 50 nodes but is less efficient at more difficult TSP than the greedy algorithm.

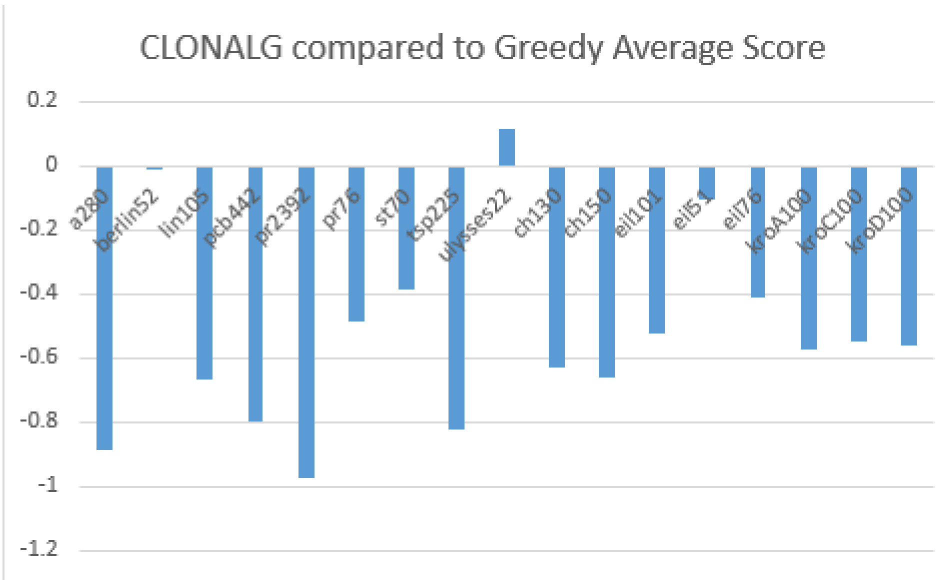

Figure 5.1: MAE on Average Score for CLONALG compared to Greedy

| TSP | Avgerage Score | Best Score | Best Percentage | MAE compared to Greedy | Average Evaluations |
|---|---|---|---|---|---|
| a280 | 26483.99 | 25581 | 891.8960838 | -0.885322038 | 38462.71 |
| berlin52 | 11077.22 | 9369 | 24.22434368 | -0.009083507 | 132557.69 |
| lin105 | 69431.08 | 62981 | 338.0068155 | -0.666910122 | 59348.99 |
| pcb442 | 649513.41 | 631174 | 1143.006814 | -0.795694149 | 38322.66 |
| pr2392 | 14231642.92 | 14122859 | 3635.889819 | -0.971810641 | 32506.49 |
| pr76 | 299815.73 | 267781 | 147.5808763 | -0.486638576 | 73496.26 |
| st70 | 1765.09 | 1510 | 123.7037037 | -0.382569727 | 82887.36 |
| tsp225 | 31309.96 | 29905 | 663.6618999 | -0.822318521 | 41252.29 |
| ulysses22 | 7273.04 | 6901 | 0 | 0.117078966 | 27588.51 |
| ch130 | 31453 | 29391 | 381.0310966 | -0.630750644 | 46159.77 |
| ch150 | 37883.24 | 36015 | 451.7003676 | -0.660824945 | 47285.45 |
| eil101 | 2126.54 | 1954 | 210.6518283 | -0.525482709 | 56058.04 |
| eil51 | 654.65 | 552 | 29.57746479 | -0.103093256 | 115597.85 |
| eil76 | 1377.39 | 1194 | 121.9330855 | -0.407509856 | 67914.53 |
| kroA100 | 96214.68 | 86675 | 307.2690537 | -0.572862998 | 60339.55 |
| kroC100 | 94471.33 | 85368 | 311.4318762 | -0.550283351 | 60015.33 |
| kroD100 | 93282 | 84305 | 295.9096459 | -0.557829485 | 57958.20 |

Table 5.2: CLONALG untuned performance

### 5.2.2 CLONALG tuned

Comparing the tuned algorithm to the original one shows, that the tuned parameter are not suited for this TSP setup. The tuned CLONALG has a worse average score in solving all 17 TSP compared to the untuned CLONALG. The unexpected outcome is, that the tuned algorithm performs better if the TSP is larger. The MAE on pr2392, which is the largest TSP in the setup, is only -0.014 but the MAE of ulysses22, the smallest TSP, is -0.305. The algorithm could not find the best solution for ulysses22. The average MAE for all TSP was -0.253 compared to the untuned algorithm. This is especially unexpected because the parameters where tuned by [CZ02] for a 30 node TSP. In their tests, the tuned algorithm behaved better on this specific TSP than the un-tuned one. This shows that the chosen parameters do not work well when the stopping criteria are 10000 evaluations without improvement.

| TSP | Avgerage Score | Best Score | Best Percentage | MAE compared to CLONALG | Average Evaluations |
|---|---|---|---|---|---|
| a280 | 29313.63 | 27910 | 982.202404 | -0.09652984 | 12173.27 |
| berlin52 | 22284.75 | 20861 | 176.5977194 | -0.502923748 | 13140.05 |
| lin105 | 95696.8 | 89181 | 520.2169831 | -0.274468112 | 12437.46 |
| pcb442 | 695194.14 | 680524 | 1240.194572 | -0.065709314 | 12179.71 |
| pr2392 | 14434730.31 | 14288477 | 3679.700396 | -0.014069358 | 11541.92 |
| pr76 | 445633.5 | 418899 | 287.2992539 | -0.327214561 | 12718.28 |
| st70 | 2826.17 | 2672 | 295.8518519 | -0.375448045 | 12650.36 |
| tsp225 | 35353.65 | 32603 | 732.5587334 | -0.114378289 | 12485.88 |
| ulysses22 | 10459.55 | 9389 | 36.05274598 | -0.304650774 | 13776.71 |
| ch130 | 38766.46 | 36521 | 497.7250409 | -0.188654316 | 13543.46 |
| ch150 | 45610.23 | 43708 | 569.5465686 | -0.169413529 | 12907.27 |
| eil101 | 2773.71 | 2685 | 326.8680445 | -0.233322878 | 13245.55 |
| eil51 | 1240.14 | 1170 | 174.6478873 | -0.472116051 | 13879.30 |
| eil76 | 1993 | 1806 | 235.6877323 | -0.308886101 | 13296.35 |
| kroA100 | 134148.49 | 125461 | 489.5169627 | -0.282774782 | 13492.89 |
| kroC100 | 132863.13 | 126691 | 510.5884621 | -0.288957516 | 13778.59 |
| kroD100 | 129082.57 | 124611 | 485.1930121 | -0.277346275 | 14173.76 |

Table 5.3: CLONALG tuned performance

### 5.2.3 CLONALG ESPC

The variant with an adaptive selection size shows only a insignificant worse performance with an average MAE for all TSP of -0.009. The interesting behaviour is seen in the average MAE for all TSP calculated on the best score with 0.039. This value shows that the adaptive CLONALG is better in finding the shorter route but will also produce some worse ones in the long run. The MAE on the best score was significant better on eil101, kroA100 and kroD100 which are bigger TSP. This results highlights that a adaptive selection size altered with the evolution strategy and combined with no random replacements can be beneficial for finding the shorter route. On average the CLONALG ESPC needed less evaluations to terminate, the stagnation started earlier in this variant. The overall runtime was nearly identical for the original CLONALG and the ESPC with close to 19 minutes for the complete run. Comparing the ESPC to the greedy algorithm still shows a worse performance on average and in most cases on best score for the ESPC. The ESPC found the shorter route on berlin52 with a MAE of 0.022 and the best route on ulysses22 with a MAE of 0.011. The modification enhanced the ability of the algorithm in solving the smaller TSP but did not have a measurable impact on the performance for greater TSP, compared to the greedy algorithm as seen in figure 5.4.

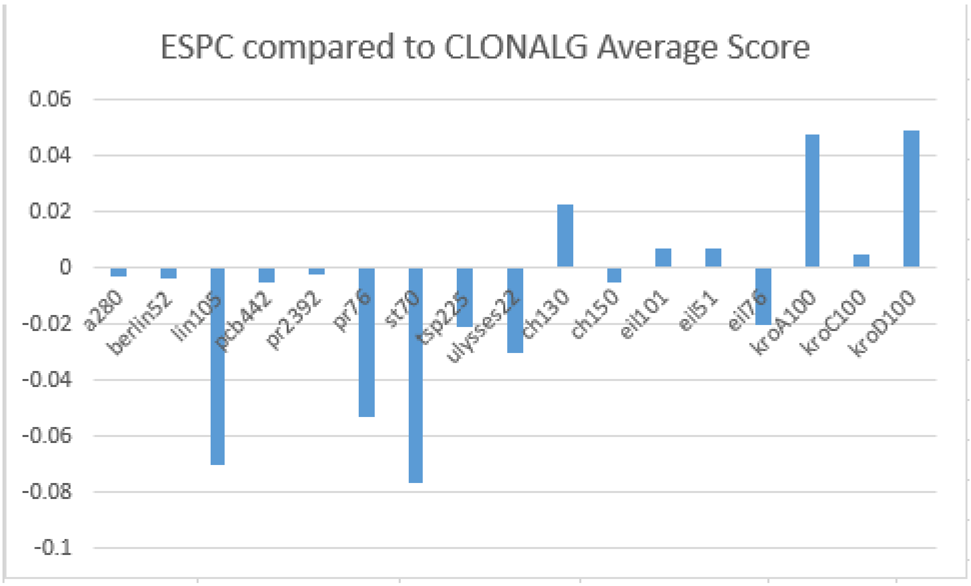

Figure 5.2: MAE on Average Score for ESPC compared to CLONALG

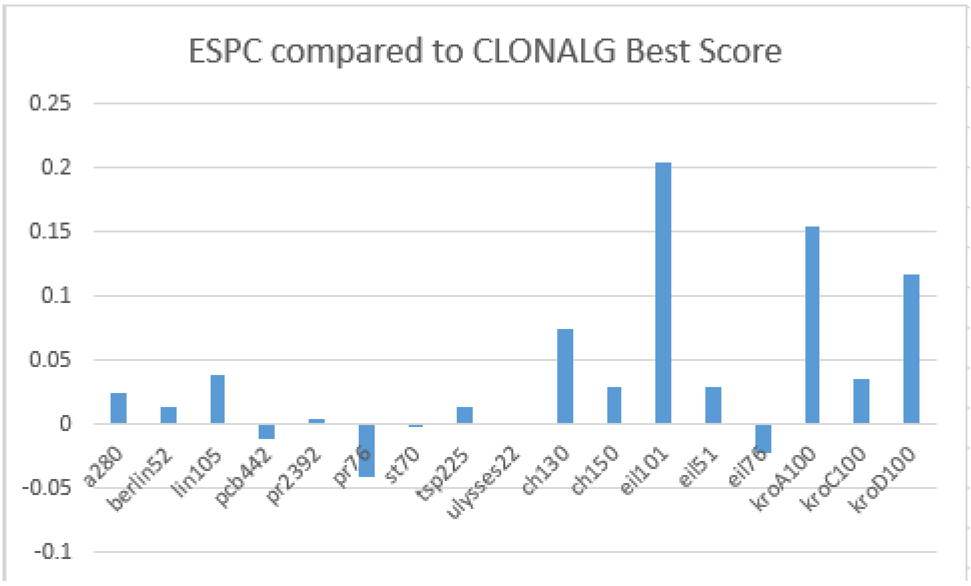

Figure 5.3: MAE on Best Score for ESPC compared to CLONALG

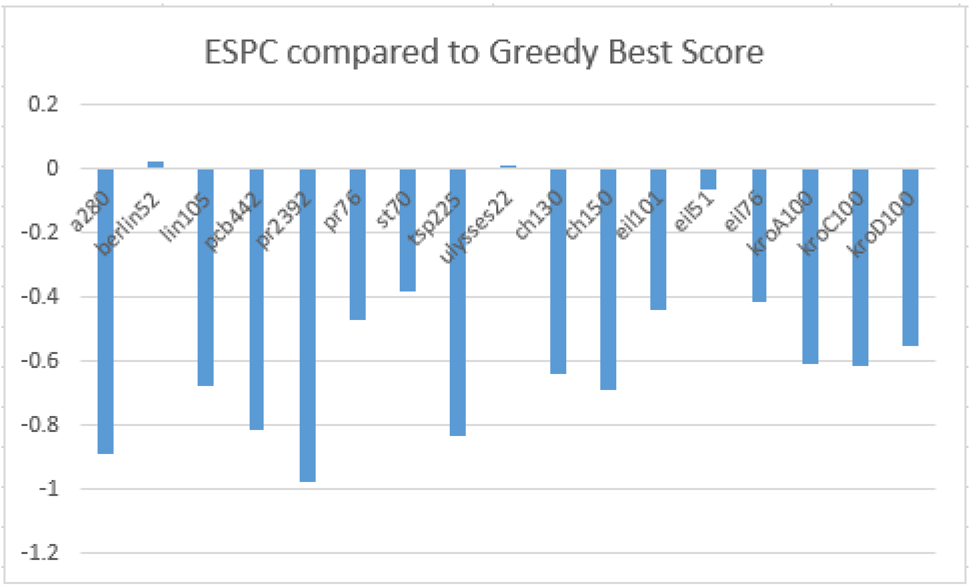

Figure 5.4: MAE on Best Score for ESPC compared to Greedy

| TSP | Average Score | Best Score | Best Percentage | MAE compared to CLONALG | MAE on Best Score | Average Evalu-tions |
|---|---|---|---|---|---|---|
| a280 | 26559.75 | 24949 | 867.3904614 | -0.002852436 | 0.025331677 | 34025.59 |
| berlin52 | 11120.65 | 9241 | 22.52718112 | -0.003905347 | 0.013851315 | 173022.70 |
| lin105 | 74683.73 | 60638 | 321.7122192 | -0.070331918 | 0.038639137 | 48963.38 |
| pcb442 | 652981.26 | 637838 | 1156.130608 | -0.005310796 | -0.010447794 | 35590.02 |
| pr2392 | 14259316.88 | 14060841 | 3619.484329 | -0.001940763 | 0.004410689 | 31152.73 |
| pr76 | 316531.58 | 278997 | 157.9507947 | -0.052809423 | -0.040201149 | 64605.41 |
| st70 | 1912.1 | 1511 | 123.8518519 | -0.076884054 | -0.000661813 | 68687.03 |
| tsp225 | 31963.99 | 29509 | 653.5495403 | -0.020461463 | 0.013419635 | 36970.86 |
| ulysses22 | 7499.74 | 6901 | 0 | -0.030227715 | 0 | 27280.05 |
| ch130 | 30748.35 | 27332 | 347.3322422 | 0.022916677 | 0.075332943 | 45024.26 |
| ch150 | 38069.35 | 34981 | 435.8609069 | -0.00488871 | 0.029558903 | 40273.29 |
| eil101 | 2111.16 | 1622 | 157.8696343 | 0.007285094 | 0.204685573 | 53395.42 |
| eil51 | 650.27 | 536 | 25.82159624 | 0.006735664 | 0.029850746 | 118097.23 |
| eil76 | 1405.74 | 1221 | 126.9516729 | -0.020167314 | -0.022113022 | 62993.43 |
| kroA100 | 91818.4 | 75100 | 252.8803684 | 0.047880163 | 0.15412783 | 62252.60 |
| kroC100 | 94006.78 | 82471 | 297.4697576 | 0.004941665 | 0.035127499 | 57812.68 |
| kroD100 | 88897.32 | 75500 | 254.5599699 | 0.049322972 | 0.116622517 | 58883.91 |

Table 5.4: ESPC performance

When looking at the average Percentage the algorithms deviate from the best solution, the Greedy algorithm outclasses the CLONALG variants. The average percentage for the Greedy is 63.7, the untuned CLONALG achieves 566.6, the tuned CLONALG 693.1 and the ESPC 570.2. These values on their own are not suited to judge the performance of the algorithm because of the wide range in difficulty of the TSP, but the difference compared to the Greedy algorithm is very high. The Greedy algorithm needed about 52 minutes to complete the whole test run, while the CLONALG variants needed about 19 minutes.

## 5.3 Time as stopping criteria

To test if a fixed amount of runtime yields different results than the no improvements criteria, the algorithms will be applied to another test run. In this run every algorithm only runs once on every TSP. The stopping criteria are now 25 seconds.
The CLONALG has an average MAE for all TSP of -0.328 compared to the Greedy algorithm. This is better than with the 10000 iterations without improvement criteria, where this MAE was -0.546. This time the CLONALG achieved a better score on the TSP berlin52, ulysses22, st70, eil51 and eil76 as shown in figure5.5. The performance for the CLONALG was better with this stopping criteria.

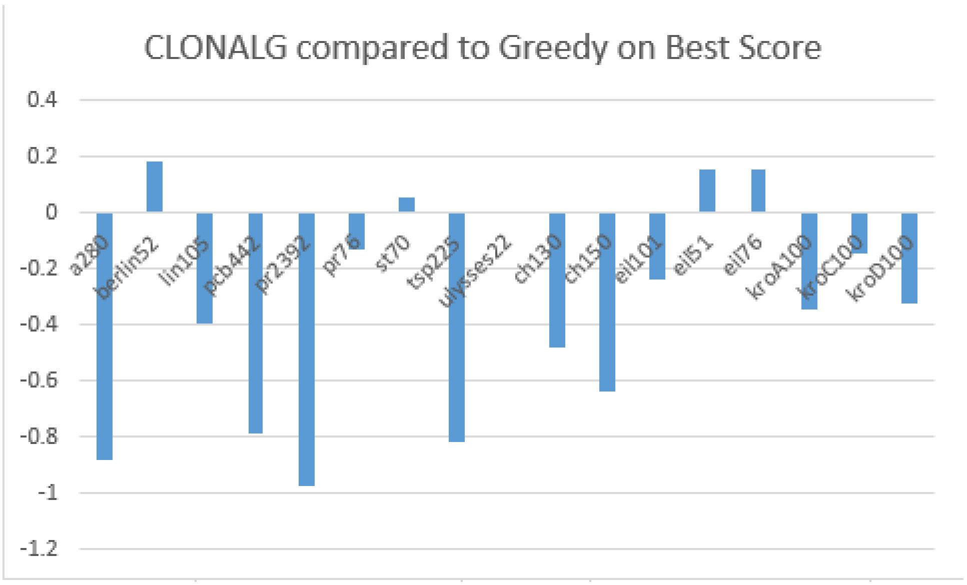

Figure 5.5: MAE on Best Score for CLONALG compared to Greedy with the 25 seconds stopping criteria

The tuned CLONALG algorithm still performs worse than the untuned, but can achieve a better score for ulysses22, berlin52 and eil51 than the greedy algorithm. It was also able to find a better route for ulysses22 compared to the tuned and the ESPC variant which was not possible in the previous evaluation.
The ESPC has an average MAE for all TSP of 0.110 compared to the CLONALG and achieves a better score on all TSP except berlin52, st70, eil51, and eil76 shown in table 5.6. The ESPC has also a better performance when running fo 25 seconds compared to the no improvements criteria.

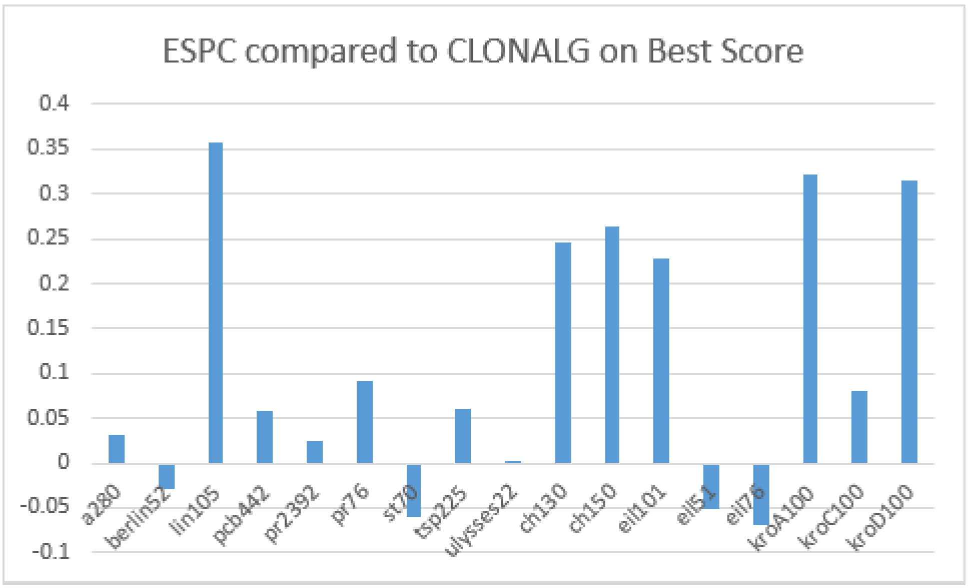

Figure 5.6: MAE on Best Score for ESPC compared to CLONALG with the 25 seconds stopping criteria

The average MAE compared to the Greedy algorithm is -0.270 which is better compared to the previous evaluation of -0.530. The ESPC scored better on the TSP berlin52, ulysses22, eil51 and eil76. These are the same TSP where the ESPC had a worse score compared to the CLONALG.

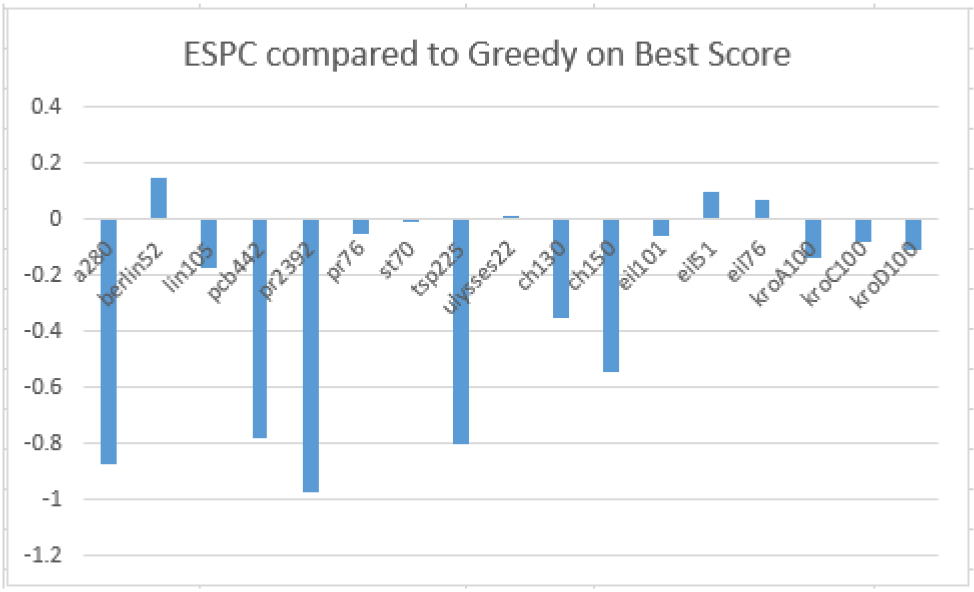

Figure 5.7: MAE on Best Score for ESPC compared to Greedy with the 25 seconds stopping criteria

The average percentage deviation from the best score got better for all algorithms but to a different degree. The Greedy algorithm now deviates 54.2, the untuned CLONALG 447.2, the tuned CLONALG 529.6 and the ESPC 410.9 percent. The deviation for the CLONALG variants became more than 100 percent better but remains still very high compared to the Greedy algorithm. This values highlight that the CLONALG variants profit more from a longer runtime in solving the TSP than the Greedy algorithm.

# Chapter 6

# Related work

M.C.Riff and E.Montero proposed a parameter free CLONALG variant in [RM09]. They tested different strategies involving the elemination of the population size, the selection size or both at the same time. The idea behind the modification was the use of a reinforcement learning model as mentioned in chapter 3. According to their results making the selecetion size dynamic alone yielded the best increase in performance when solving a TSP, which is comparable to the results of the ESPC in chapter 5. The ESPC achieved the same result with a different strategy.
For the ESPC variant in this thesis some techniques from S.M.Garret's work on parameter free clonal selection desribed in [Gar04] were used. Garret applied different parameter free variants of the CLONALG to different multimodal problems. The modification was based on the evolution strategy where a static parameter is used to alter the dynamic parameters. Garret was able to successful eliminate all parameters except population size which still has to be initialized with a value best suited for the specific task. The adaptive clonal selection ACS was also tested with different strategies eliminating one parameter at the time. The results showed that the ACS with all parameters eleminated could outperfom the compared algorithms. The author also speculates that the ACS could be especially well suited for dynmamic multimodal problems, where the optima are changing over time. This could not be tested in his work [Gar04].
The designer of the original CLONALG L.E.de Castro and F.J.Von Zuben also proposed some parameter control based on sensitivity analysis [CZ02]. The tuned parameters were tested on a 30 nodes TSP and achieved better results than the untuned algorithm. This parameters were also applied to the evaluation in chapter 5 but yielded much different results than in de Castro and Von Zuben's work.
L.Pasti and L.N.de Castro proposed a neuro-immune hybrid algorithm in [PC06]. A single layer self organized neural network was combined with the clonal selection from the immune system. The results showed that this hybrid was able to produce high quality results for many TSP and had good performance in solving more difficult TSP with a high amount of nodes.
Wei-Dong Sun et al tested an artificial immune system based on the immune network theory on small TSP to a range of 100 nodes. Their proposed algorithm could achieve the best score for the TSP in 63 percent of runs as stated in [Sun+04].

# Chapter 7

# Conclusion

When using the 10000 evaluations without improvement as a stopping criteria, the algorithms based on the CLONALG show a worse performance compared to the greedy algorithm on average. The results still show potential for these algorithms based on their better performance with smaller TSP espcecially in finding the shorter route or in case of one TSP in finding the best route. The clonal based algorithms also terminated faster which can be an advantage for certain applications even if the overall quality of the result is worse.
Static tuning of the parameters made the performance of the CLONALG worse. The dynamic tuning of the selection size based on evolution strategy however produced a better performance than the original CLONALG. This shows that adaptive tuning is very promising for the cloning selection algorithms as also seen in the related work of [Gar04] and [RM09].
A different stopping criteria of 25 seconds and only one run on every TSP yielded better results for the CLONALG variants. All variants achieved a better performance and could produce more shorter routes compared to the Greedy algorithm than with the first stopping criteria. All tested algorithms are very basic and not optimized for solving the TSP but still achieved a good performance on small TSP in a range under 100 nodes. Adapting the cloning parameters during runtime and giving the algorithm enough time are beneficial to the performance.

# Bibliography

[App+07] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007. ISBN: 0691129932, 9780691129938.

[AR94] Howard Anton and Chris Rorres. *Elementary linear algebra : applications version*. English. 7th ed. Includes index. New York : Wiley, 1994. ISBN: 0471587419 (acid-free).

[BK13] E.K. Burke and G. Kendall. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. SpringerLink : Bücher. Springer US, 2013. ISBN: 9781461469407.

[Bro] Jason Brownlee. *Optimization Algorithm Toolkit*. `http://optalgtoolkit.sourceforge.net/`[Accessed: 18.12.2018].

[BS02] Hans-Georg Beyer and Hans-Paul Schwefel. “Evolution Strategies &Ndash;A Comprehensive Introduction”. In: 1.1 (May 2002), pp. 3–52. ISSN: 1567-7818.

[CZ02] L. N. de Castro and F. J. Von Zuben. “Learning and optimization using the clonal selection principle”. In: *IEEE Transactions on Evolutionary Computation* 6.3 (2002), pp. 239–251. ISSN: 1089-778X. DOI: `10.1109/TEVC.2002.1011539`.

[Gar04] S. M. Garrett. “Parameter-free, adaptive clonal selection”. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. Vol. 1. 2004, 1052–1058 Vol.1. DOI: `10.1109/CEC.2004.1330978`.

[Jan01] C. Janeway. *Immunobiology Five*. Immunobiology 5 : the Immune System in Health and Disease. Garland Pub., 2001. ISBN: 9780815336426.

[MSM10] Rajesh Matai, Surya Singh, and Murari Lal Mittal. “Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches”. In: *Traveling Salesman Problem*. Ed. by Donald Davendra. Rijeka: IntechOpen, 2010. Chap. 1. DOI: `10.5772/12909`.

[Nan+08] S. J. Nanda et al. “Development of a New Optimization Algorithm Based on Artificial Immune System and Its Application”. In: *2008 International Conference on Information Technology*. 2008, pp. 45–48. DOI: `10.1109/ICIT.2008.20`.

[Pam17] M. E. Pamukov. “Application of artificial immune systems for the creation of IoT intrusion detection systems”. In: *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 1. 2017, pp. 564–568.

[PC06] R. Pasti and L. Nunes de Castro. “A Neuro-Immune Network for Solving the Traveling Salesman Problem”. In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. 2006, pp. 3760–3766. DOI: `10.1109/IJCNN.2006.247394`.

[PG95] Karl Pearson and Francis Galton. “VII. Note on regression and inheritance in the case of two parents”. In: *Proceedings of the Royal Society of London* 58.347-352 (1895), pp. 240–242.

[Rei] Gerhard Reinelt. *TSPLIB95*. `https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`[Accessed: 12.12.2018].

[RM09] M. C. Riff and E. Montero. “A Dynamic Adaptive Calibration of the CLONALG Immune Algorithm”. In: *2009 International Conference on Adaptive and Intelligent Systems*. 2009, pp. 187–193. DOI: `10.1109/ICAIS.2009.38`.

[Rob08] D.J.S. Robinson. *An Introduction to Abstract Algebra*. De Gruyter Textbook. De Gruyter, 2008. ISBN: 9783110198164.

[Sun+04] W. . Sun et al. “An immune optimization algorithm for TSP problem”. In: *SICE 2004 Annual Conference*. Vol. 1. 2004, 710–715 vol. 1.

[Tan16] Y. Tan. *Artificial Immune System: Applications in Computer Security*. Wiley, 2016. ISBN: 9781119076285.