Artificial Immune Systems for Solving the Traveling Salesman Problem

A use of CLONALG in Optimization

Bachelor Thesis

University of Applied Sciences FH Campus Wien
Department of Information Technology and Telecommunication

Submitted by:
Sebastian Ukleja

Supervisor:
Priv.-Doz. Mag. DI. DI. Dr.techn. Karl Michael Göschka

Student-ID
1610475014

Area of Specialization
IT Security

Submitted on:
17.01.2019

Abstract

Artificial immune systems are, along with neural nets and genetic algorithms, an important part of the bio-inspired approaches in machine learning. This thesis examines artificial immune systems and their performance in solving the traveling salesman problem (TSP). The TSP has many applications in industry and in scientific research like drilling of printed circuit boards, computer wiring, order picking in warehouses, vehicle routing and DNA sequencing, therefore new and more efficient ways for finding TSP solutions are always relevant. The CLONALG algorithm is compared to a more conventional greedy algorithm. To achieve this, the Optimization Algorithm Toolkit will be used and the CLONALG, a Greedy algorithm, a tuned CLONALG and a modified CLONALG with parameter control will be compared in solving 17 different TSP. Comparison shows that the clonal algorithms can achieve better results under certain circumstances but are, in their general form, not as efficient as the greedy algorithm. The CLONALG variants show better performance when applied to smaller TSP in the range from 22 to 100 nodes and can also find the optimal route for one TSP contrary to the Greedy algorithm. The CLONALG and most clonal selection algorithms work with static parameters. This thesis examines if parameter control is beneficial for solving the traveling salesman problem and shows that dynamic adaptation of certain parameters during runtime can enhance performance of the algorithm. A variant of the CLONALG with dynamic selection size (how many members of the initial population will be cloned) is implemented, tested and shows better performance than the original algorithm and the tuned one. The modified CLONALG is able to find shorter routes for many of the presented TSP compared to the original algorithm.

Abbrevations

AIS	Artificial Immune System
Ab	Antibody
Ag	Antigen
BIS	Biological Immune System
CPU	Central Processing Unit
DNA	Deoxyribonucleic acid
MAE	Mean Average Error
OAT	Optimization Algorithm Toolkit
TSP	Traveling Salesman Problem
RAM	Random Access Memory
XOR	Exclusive Or

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 efficient methods for learning and adaptation, therefore bio-inspired approaches in machine learning were successful over the past years. The immune system got more attention in the computational intelligence field over the last 20 years [Tan16].

The biological immune system has a number of desireable traits for learning in computational science as discussed 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 the same fields as neural nets or genetic algorithms, but are also efficient in combination with these algorithms [PC06]. One of the application fields is the traveling salesman problem. The artificial immune system has been successfully applied to the TSP by [CZ02] and [Sun+04]. This thesis focuses on the overall performance of the AIS in solving the TSP, especially compared to a more conventional method. The artificial immune system works with different approaches taken from the biological immune system. One of these approaches is clonal selection which was formalized into an algorithm by [CZ02]. In chapter 5 this algorithm termed 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] has shown that by 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 thesis aims to examine if the AIS is able to achieve good results in solving the TSP and especially which circumstances and 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 useful for machine learning. It can be described with the following terms [Tan16]:

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

Distribution means there is no need for a central control instance, the BIS acts where it needs to and it does so immediately without supervision. It is able to operate on multiple parts of the body simultaneously, thus it is parallel. The BIS works on different levels: first there is the physical barrier, the skin and body fluids. If this level fails the innate immune system responds, a pre-programmed immune reaction able to respond to known and unchanging threats. The last level consists of the adaptive immune system, which takes over if all of the previous levels fail. This system holds the most interest for computational science because of its ability to learn on its own. An important ability of the BIS is the distinction between any self-element of the body and the non-self or potential threat. The BIS 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, the associative memory is used to react to similar and variable 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 BIS uses many principles to be effective, in terms of computational science the principles of negative selection and clonal selection are especially interesting.

2.1 Negative Selection

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

After the negative selection the lymphocytes are be released into the tissue and blood system. In this context B cells are called antibodies (Ab), and any non-self cell is called an antigen (Ag). B cells have the ability to recognize and dock to a specific Ag. The ability to recognize an Ag is called affinity. Ab with a high affinity to an Ag are especially good at recognizing and countering this Ag. Because the production of lymphocytes and therefore the affinity to different Ag is random, additional measures are necessary to improve effectiveness.

2.2 Clonal Selection

If a B-Cell encounters an Ab it is able to proliferate (divide) into multiple terminal cells which are clones of the cell. In addition the different clones are mutated to improve affinity to the Ag. The process of cloning and mutating is repeated until a population and affinity threshold is reached, ensuring the most efficient response to the Ag. Only Ab with the highest affinity scores are cloned and mutated. If affinity is high enough, the B cell can proliferate into memory cells which will stay after the initial immune response and ensure that a secondary response to a similar Ag will be much faster than the initial one [CZ02]. These 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 AIS.The clonal selection aspect of the BIS is basically a learning system. The cloning process is a form of reinforcement learning and leads to a continued 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 reflected in 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 be maintained)

The mentioned mechanisms of the BIS 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.
The next chapter will show how an Artificial Immune System can be build using the mentioned principles.

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). In computational science there is usually no distinction between a T cell and a B cell, most artificial immune system generalize both into an Ab. The Ag could be malicious code, an IP address and port combination not allowed in the network or a pattern to be classified. What represents an Ag in the algorithm 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 AIS is also able to improve the efficiency of genetic algorithms by avoiding premature convergence and aiding in finding the global optimum in problems with multiple criteria [Tan16, p. 21].

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

1. Initialize/present antigen
2. Initialize antibody population
3. Calculate affinity to the antigen for each antibody
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 4 are where negative selection and clonal selection is relevant in most forms of AIS algorithms.

At the time of writing 5 main concepts 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 in 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, the affinity is therefore high enough, it is destroyed. In this case the AIS follows the process that is found in the BIS.

It is important to decide in which way affinity is measured and which conditions trigger an immune reaction. If the used data consists of numerical values, or is easily converted into a vector thereof, the Euclidean distance [AR94] is commonly used as a mean to measure affinity [TAN16]:

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The Euclidean distance measures the straight-line distance between two points in Euclidean space. In the formula above x_i and y_i are values of two n-dimensional vectors x and y [AR94].

Another way of measuring the affinity is the use of different variants of the Hammington distance. The Hammington distance is the number of bits which 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-continuous bits. In this case a detector recognizes an element if r-continuous bits are identical with the element.

Non-self element	0010110
Detector	0011101

In the example above the first 3 continuous bits 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]. However 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 useful 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 using the clonal selection principle. This thesis focuses 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 are chosen, determined by a fixed fitness value and form the antibody set S. Every Ab in S is cloned and represents the clone set P. Now the clone set is mutated, every clone randomly changed relatively to its affinity value. After the mutation the n best clones out of P are chosen. Then a new population is generated and the process begins anew until a stopping condition is met.
An example of pseudocode based on [RM09] can be illustrated as follows:

```
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 has fallen 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 stated in [CZ02]. These parameters must be changed beforehand and cannot adapt dynamically during the runtime of the algorithm.

The main parameters of the CLONALG are population size, selection size, clone factor and random replacements. Population size is the initial set of Ab the algorithm starts with. Selection size is the number of Ab chosen for cloning out of the population size. The clone factor determines how many clones of the selected Ab are generated and random replacements are new Ab added to the population at the end of every iteration to keep the population partly randomized. Additionally, the mutation factor governs to what degree a clone will mutate.

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

3.3.2 Adaptive CLONALG variants

To make the algorithm more efficient and adaptive, some variants have been proposed. There are different parameter control strategies for the CLONALG algorithm. One of these variants is proposed by [RM09] and is based on the concept of reinforcement learning. Ab sets are either rewarded for high affinity or penalized for a low one. This is achieved through population control. The reward consists of an increase in Ab population for a given set and the penalty is the decrease in population [RM09]. The mutation factor is governed by population size, allowing the algorithm to adjust the core parameters during runtime and making it more efficient in problem solving and hardware usage [RM09].
Another method is proposed by [Gar04] where some techniques of evolutionary algorithms are applied to the CLONALG. An algorithm based on this concept will be evaluated in chapter 5.
In this part the basics of an AIS were discussed, the next chapter will highlight the traveling salesman problem and its definition.

Chapter 4

Traveling Salesman Problem

4.1 Background

One of the first publications on the traveling salesman problem(TSP) was written the mathematician Karl Menger in the 1920's [App+07], although the problem itself has been discussed earlier by Sir William Rowam Hamilton and Kirkman [MSM10]. It describes a graph with a certain amount of nodes and known distances between the nodes. The goal is to find a route where every node is visited exactly once and the route ends at the starting node. The route should be the shortest possible.

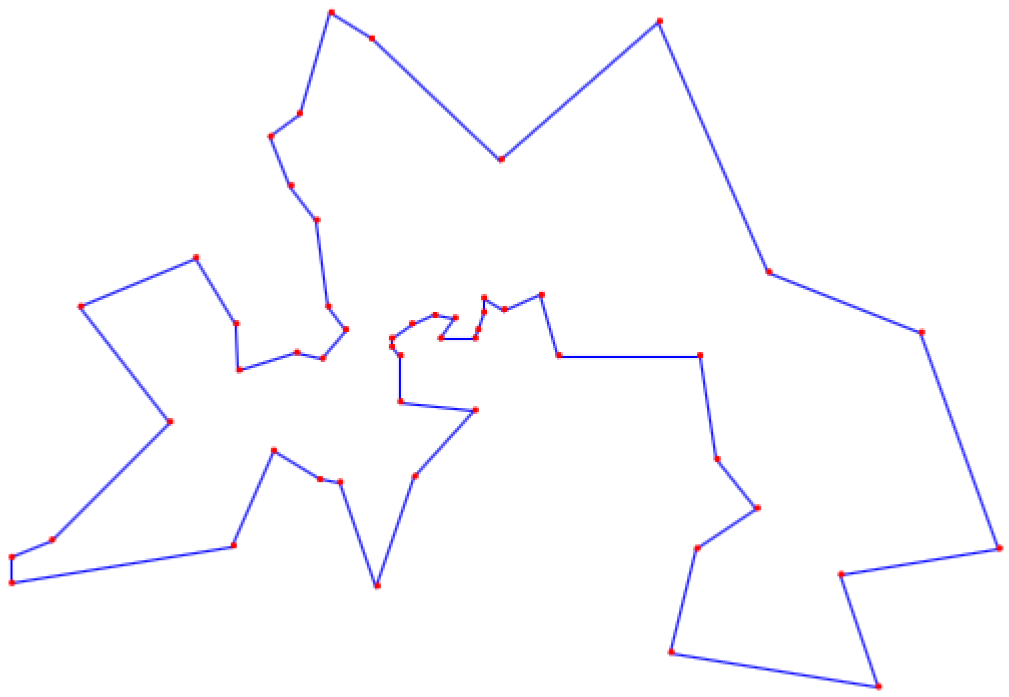

Figure 4.1: A solved TSP berlin52 (source: OAT)

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 TSP 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 and 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 examples [MSM10].

4.2 Mathematical definition

The symmetric graph is defined as $G = (V, E)$ where $V = \{1, .., n\}$ are the vertices 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 edges of the arc set. Usually the cost matrix is calculated using Euclidean distance [MSM10].

A typical TSP consists of a set of cities (V), distances between the cities (E) and the cost measured in Euclidean distance between the cities (C). All TSP used in this thesis will follow this convention.
The AIS was successfully applied to different TSP by [CZ02] and [RM09] This thesis will further examine the performance of an AIS in solving the TSP and which parameters within the AIS are beneficial for a good solution. Furthermore, section 5 will evaluate under which circumstances AIS are suitable to solve TSP efficiently. The proposed tuning of the CLONALG algorithm by [CZ02] and a modified algorithm, described in section 5.1.1, will be tested on a wide range of different TSP and two different stopping criteria. The next chapter describes the evaluation of the different algorithms and the changes made to them to examine their performance for solving the TSP.

Chapter 5

Evaluation

5.1 Setup

The CLONALG algorithm will be applied to a set of 17 different TSP problems taken from the TSP library TSPLIB95[1]. The number after the name 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]. All algorithms in the OAT are implemented in Java. The greedy search algorithm will be applied to the same set of problems, 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 improvement after a set amount of evaluations. No improvement means the algorithm did not find a shorter route in it's current iteration compared to the last, this is also called stagnation. Every algorithm will run 100 times on one TSP, making it 1700 runs for all 17 TSP. 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 relative difference between the best possible route of the TSP and the result of the algorithm. 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 results the mean average error (MAE) of the average score will be used. The MAE is composed of the difference between the score of both algorithms divided by the score of the compared algorithm. To compare the performance of algorithm A1 to

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

A2 the MAE is calculated as $(A2 - A1)/A1$. Positive MAE means better performance for A1. The significance of the difference is visible at the hundreds and tens digit, the thousands digit shows no significant difference. This was tested by running the complete test run twice on the same algorithm. Since the only difference between the single test runs was in the thousands digit it was not 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 to be altered are:

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

The first set of parameters will be default parameters provided by the OAT as shown in table 5.1. The second set of parameters have been proposed by DeCastro [CZ02] for solving TSP with a size of about 30 nodes. The adaptive variant of the CLONALG algorithm will start with default values and adjust the parameters during runtime. To measure the results, a modified version of the OAT will be used. The changes to the OAT consist of a slightly different set of TSP used in the domain and the addition of an adaptive CLONALG hybrid algorithm described in 5.1.1. The used TSP are listed in table 5.1. The distances between 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 TSP problem.

The hardware used consists of a i5-3320M dual core CPU with 2,60ghz each and 8gb of RAM, run on a Windows 10 operating system. 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 adaptive CLONALG variation is based on [Gar04] and uses a concept from the evolution strategy [BS02]. The algorithm was able to eliminate all parameters except population size from the CLONALG algorithm. This thesis uses a variant where the selection size n is eliminated and is comparable to variant C1-C4 from [Gar04, p. 1054]. For this purpose the CLONALG implementation of the OAT is taken and modified based on the concepts in [Gar04]. A strategy parameter adjusting selection size will be used. The strategy parameter itself will be adjusted by an evolution strategy constant of 1.3. This constant has been tested empirically and proven to be the most effective [Gar04]. The adjustment will be 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. However the difference to the original evolution strategy is that a parameter is indirectly adjusted through another parameter [Gar04]. The algorithm used in this thesis generates the strategy parameter out of a gaussian random number, the result are the last selection size + the last strategy parameter * a random gaussian number. This is specific to the parameter generation in the OAT and is not necessary, the generation of the strategy parameter can be done in any other way suitable for the given application. Naturally, the selection size still has to be initialized and can't be zero. As [Gar04] proposed to start with a small number, the test runs in this thesis will 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, standing 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: Improvement Limitation

5.2.1 CLONALG untuned

The algorithms are run 100 times on every TSP. The stopping criteria are defined as 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% with an 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 stands for the initial population, n for the selection size for cloning, B for cloning factor (how many clones of the chosen n will be made) and d for random replacements to keep the new population partially randomized. [CZ02] proposed tuning the default parameters for more efficiency in solving TSP. The replacement value d

should always be between 5-20% of the population since higher ratios tend to randomize too much while lower ratios still produce good but less reliable and efficient results [CZ02].

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

Table 5.1: Tuning parameters

Comparing the performance of the untuned CLONALG to the greedy algorithm shows that the CLONALG algorithm on average performs worse than the greedy search algorithm (figure 5.1). 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 table 5.2.In addition the overall time for solving all problems was also shorter for the CLONALG. The number of evaluations is most of the time smaller for the CLONALG showing that stagnation started earlier. The average MAE for all TSP compared to the greedy algorithm is -0.524. The average MAE for all TSP calculated on the best instead of the average score is in the same range with with a result of -0.546. The mean average error of -0.009 is not significant for the TSP berlin52, highlighting that CLONALG performs equaly good or better on small TSP under 50 nodes, but is less efficient at more difficult TSP compared to the greedy algorithm.

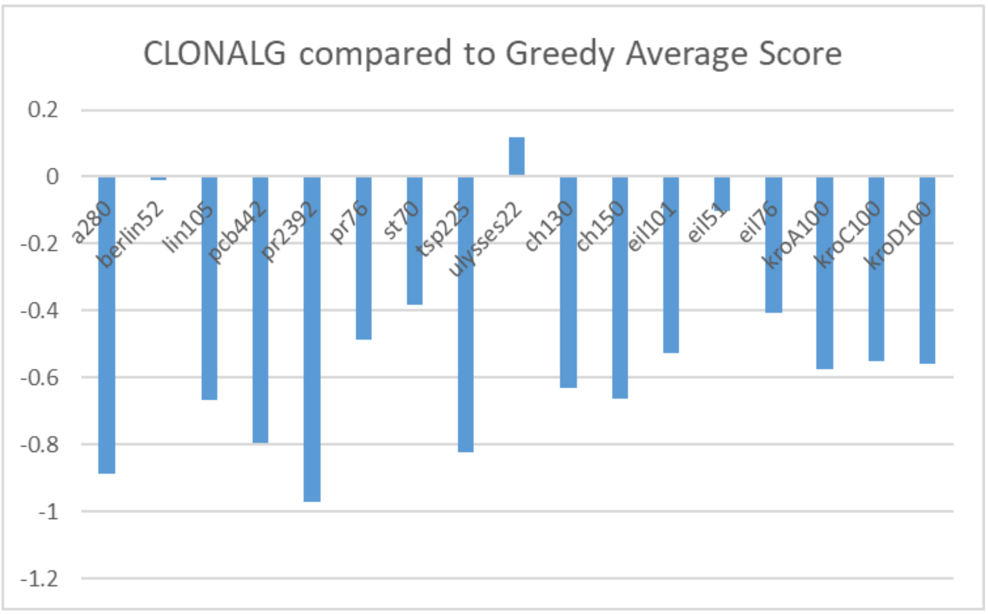

Figure 5.1: MAE on Average Score for CLONALG compared to greedy

TSP	Avgerage Score	Best Score	Best Percentage	**MAE compared to greedy**	Average Evaluations
a280	26483.99	25581	891.896	-0.885	38462.71
berlin52	11077.22	9369	24.224	-0.009	132557.69
lin105	69431.08	62981	338.007	-0.667	59348.99
pcb442	649513.41	631174	1143.007	-0.796	38322.66
pr2392	14231642.92	14122859	3635.890	-0.972	32506.49
pr76	299815.73	267781	147.581	-0.487	73496.26
st70	1765.09	1510	123.704	-0.383	82887.36
tsp225	31309.96	29905	663.662	-0.822	41252.29
ulysses22	7273.04	6901	0	0.117	27588.51
ch130	31453	29391	381.031	-0.631	46159.77
ch150	37883.24	36015	451.700	-0.661	47285.45
eil101	2126.54	1954	210.652	-0.525	56058.04
eil51	654.65	552	29.577	-0.103	115597.85
eil76	1377.39	1194	121.933	-0.408	67914.53
kroA100	96214.68	86675	307.269	-0.573	60339.55
kroC100	94471.33	85368	311.432	-0.550	60015.33
kroD100	93282	84305	295.910	-0.558	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 parameters are not suited for the tested TSP setup. The tuned CLONALG has a worse average score in solving all 17 TSP compared to the untuned CLONALG. An unexpected outcome is that the tuned algorithm performs better on larger TSP. The MAE on pr2392, the largest TSP in the setup, is only -0.014 whereas the MAE of ulysses22, the smallest TSP, is -0.305. The algorithm could not find the best solution for ulysses22. Compared to the untuned algorithm the average MAE for all TSP was -0.253. This is especially unexpected because the parameters where tuned by [CZ02] for a 30 nodes TSP. In their tests, the tuned algorithm behaved better on this TSP than the untuned one. The results show that the chosen parameters do not work well if 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	982.202	-0.097	12173.27
berlin52	22284.75	20861	176.598	-0.501	13140.05
lin105	95696.8	89181	520.217	-0.274	12437.46
pcb442	695194.14	680524	1240.195	-0.066	12179.71
pr2392	14434730.31	14288477	3679.700	-0.014	11541.92
pr76	445633.5	418899	287.299	-0.327	12718.28
st70	2826.17	2672	295.852	-0.375	12650.36
tsp225	35353.65	32603	732.559	-0.114	12485.88
ulysses22	10459.55	9389	36.0527	-0.305	13776.71
ch130	38766.46	36521	497.725	-0.189	13543.46
ch150	45610.23	43708	569.547	-0.169	12907.27
eil101	2773.71	2685	326.868	-0.233	13245.55
eil51	1240.14	1170	174.648	-0.472	13879.30
eil76	1993	1806	235.688	-0.309	13296.35
kroA100	134148.49	125461	489.517	-0.283	13492.89
kroC100	132863.13	126691	510.588	-0.289	13778.59
kroD100	129082.57	124611	485.193	-0.277	14173.76

Table 5.3: CLONALG tuned performance

5.2.3 CLONALG ESPC

The variant with an adaptive selection size shows only an insignificantly worse performance with an average MAE of -0.009 for all TSP compared to the tuned CLONALG. An 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 suited to finding the shorter route, but will also produce some worse solutions in the long run. The MAE on the best score was significantly better on eil101, kroA100 and kroD100, the bigger TSP in this setup. These results highlight that an adaptive selection size altered with the evolution strategy and combined with no random replacements can be beneficial to finding the shorter route. On average the CLONALG ESPC needed less evaluations to terminate, the stagnation started earlier. The overall runtime was nearly identical for the original CLONALG and the ESPC, with close to 19 minutes for a complete test run on all 17 TSP. Comparing the ESPC to the greedy algorithm still shows a worse performance of the ESPC, both on average score and in the majority of the best scores. 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 smaller TSP compared to the greedy algorithm, but did not have a measurable impact on its performance on greater TSP (figure 5.4).

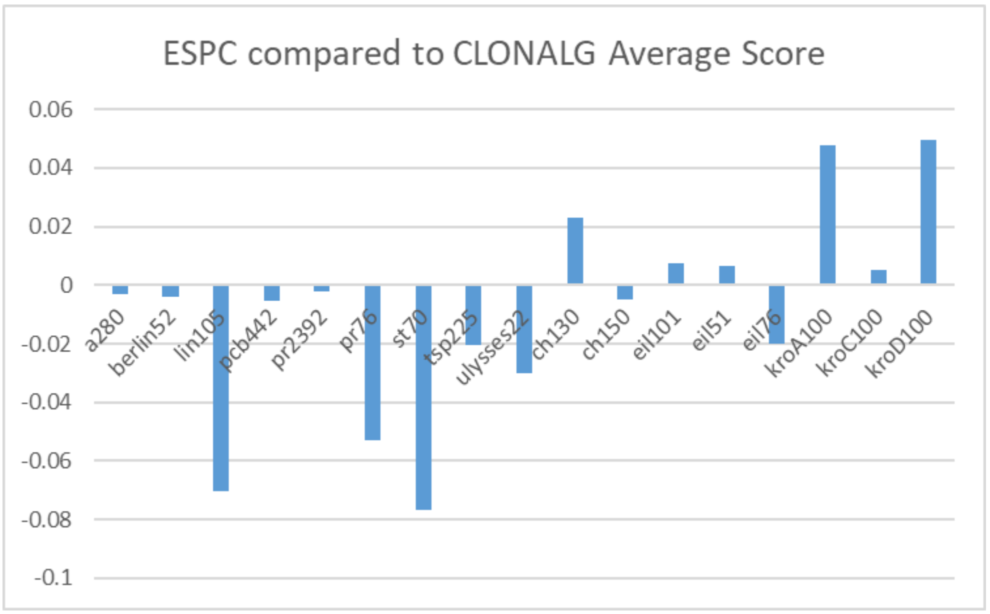

Figure 5.2: MAE on Average Score for ESPC compared to CLONALG

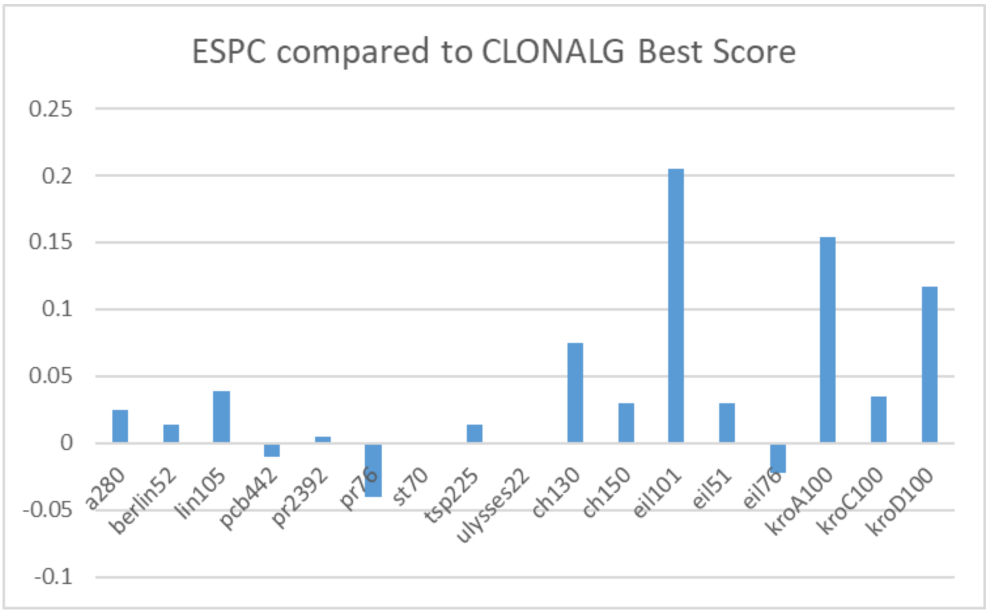

Figure 5.3: MAE on Best Score for ESPC compared to CLONALG

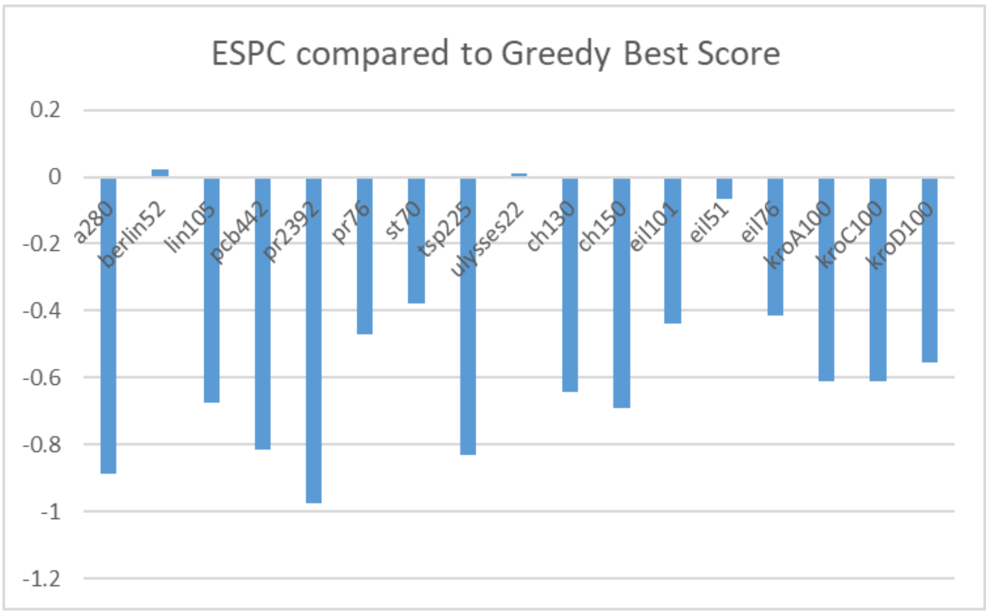

Figure 5.4: MAE on Best Score for ESPC compared to greedy

TSP	Average Score	Best Score	Best Percentage	**MAE compared to CLONALG**	MAE on Best Score compared to CLONALG	Average Evalutions
a280	26559.75	24949	867.390	-0.003	0.025	34025.59
berlin52	11120.65	9241	22.527	-0.004	0.014	173022.70
lin105	74683.73	60638	321.712	-0.070	0.039	48963.38
pcb442	652981.26	637838	1156.131	-0.005	-0.010	35590.02
pr2392	14259316.88	14060841	3619.484	-0.002	0.004	31152.73
pr76	316531.58	278997	157.951	-0.052	-0.040	64605.41
st70	1912.1	1511	123.852	-0.077	-0.001	68687.03
tsp225	31963.99	29509	653.550	-0.020	0.013	36970.86
ulysses22	7499.74	6901	0	-0.030	0	27280.05
ch130	30748.35	27332	347.332	0.023	0.075	45024.26
ch150	38069.35	34981	435.861	-0.005	0.030	40273.29
eil101	2111.16	1622	157.870	0.007	0.205	53395.42
eil51	650.27	536	25.822	0.007	0.030	118097.23
eil76	1405.74	1221	126.952	-0.020	-0.022	62993.43
kroA100	91818.4	75100	252.880	0.048	0.154	62252.60
kroC100	94006.78	82471	297.470	0.005	0.035	57812.68
kroD100	88897.32	75500	254.560	0.049	0.117	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 Results: Time Limitation

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 runs once on every TSP. The stopping criteria is 25 seconds.
The CLONALG has an average MAE of -0.328 for all TSP compared to the greedy algorithm. This is better than with the criteria set to 10000 iterations without improvement, where this MAE was -0.546. This time the CLONALG achieved a better score on the TSP berlin52, ulysses22, st70, eil51 and eil76 (figure5.5). The performance of the CLONALG was better with this stopping criteria.

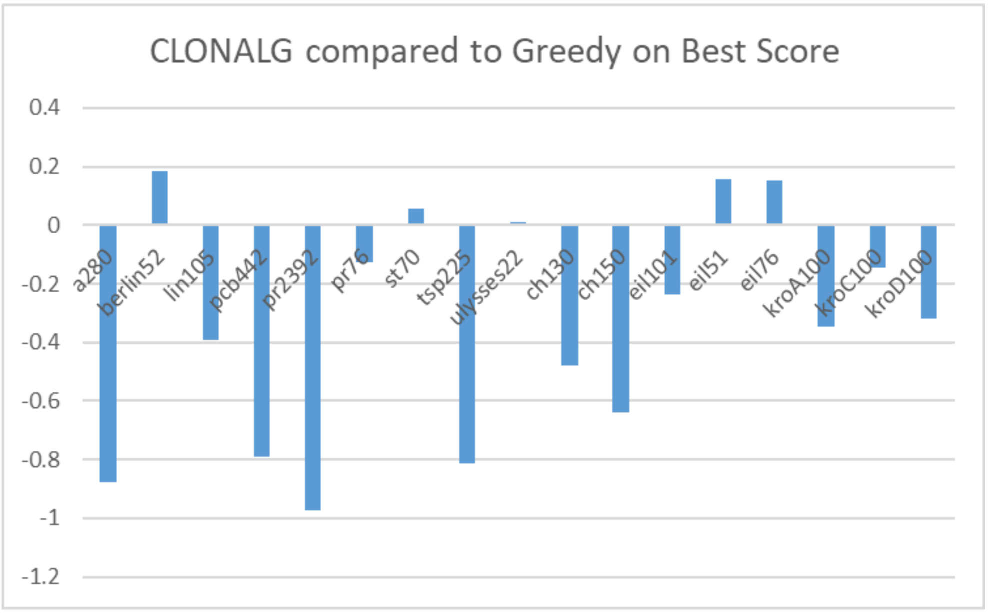

Figure 5.5: MAE on Best Score for CLONALG compared to greedy with the 25 seconds stopping criteria

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 of 0.110 for all TSP compared to the CLONALG and achieves a better score on all TSP except berlin52, st70, eil51, and eil76 (table 5.6)). The ESPC also shows a better performance when running for 25 seconds compared to the no improvements criteria.

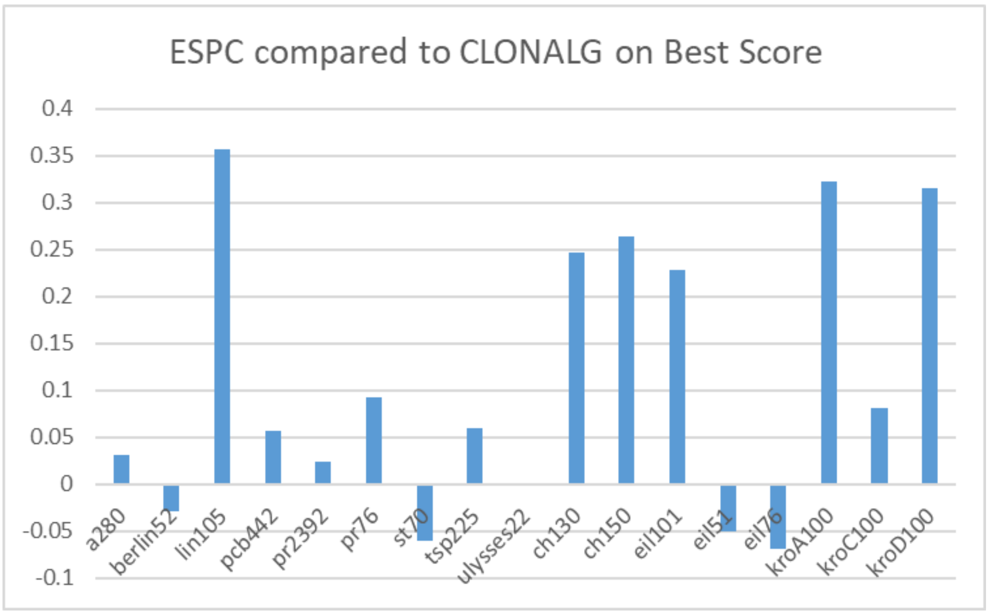

Figure 5.6: MAE on Best Score for ESPC compared to CLONALG with the 25 seconds stopping criteria

The average MAE compared to the greedy algorithm is -0.270, which is better than 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 than the CLONALG.

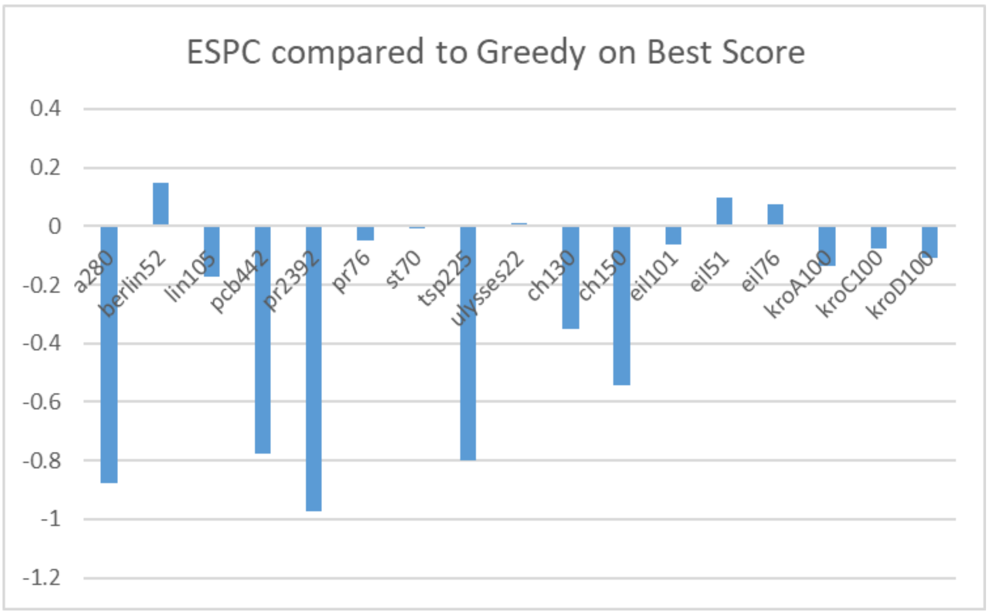

Figure 5.7: MAE on Best Score for ESPC compared to greedy with the 25 seconds stopping criteria

The average deviation percentage from the best score became 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. These values highlight that the CLONALG variants profit more from a longer runtime in solving the TSP than the greedy algorithm. The deviation for the CLONALG variants improved more than 100 percent, but still remained very high compared to the greedy algorithm (table 5.5). This is best seen on the largest TSP pr2392, where all clonal algorithms had a deviation from optimal score of more than 3000 percent. The greedy algorithm shows an exceptional performance on pr2392, with only over 1 percent deviation from optimal score, however, this TSP is highly structured and organized (figure 5.8). The nearest-neighbour technique of this algorithm is suited for solving this TSP, but is less efficient at less structured and more random TSP. The clonal algorithms do not benefit from any kind of order, symmetry or structure in the TSP, but they are also not negatively affected by the lack thereof.

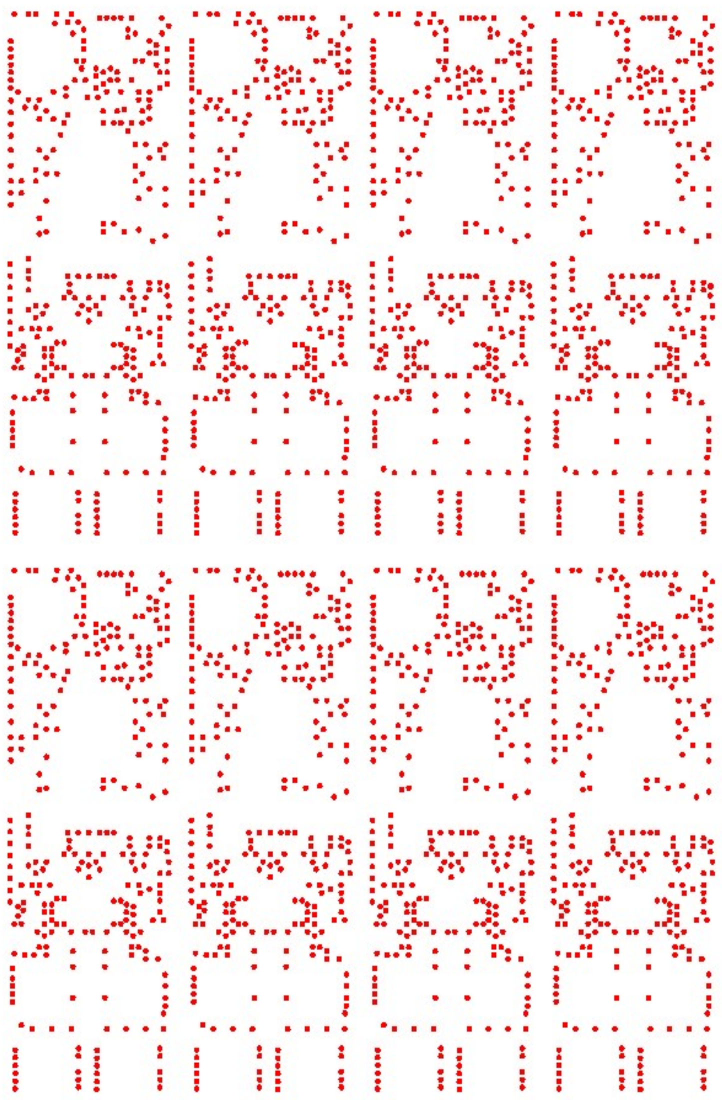

Figure 5.8: pr2392 the largest TSP in the setup (source: [Coo])

Average deviation percentage from optimal score	**10000 iterations without improvement stopping criteria**	**25 seconds stopping criteria**
Greedy	63.7	54.2
CLONALG tuned	693.1	592.6
CLONALG untuned	566.6	447.2
ESPC	570.2	410.9

Table 5.5: Average deviation from optimal score

Chapter 6

Related work

M.C.Riff and E.Montero proposed a parameter free CLONALG variant in [RM09]. They tested different strategies involving elemination of population size, selection size or both at the same time. The concept behind the modification was the use of a reinforcement learning model as mentioned in chapter 3. According to their results solely making the selecetion size dynamic 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 described 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 successfully eliminate all parameters except population size, which still has to be initialized with a value best suited to 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 eliminated, could outperform compared algorithms. The author also speculates that the ACS could be especially well suited to dynamic multimodal problems, where the optima change over time. However, this could not be tested in his work [Gar04].
The designers of the original CLONALG, L.E.de Castro and F.J.Von Zuben, proposed 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. These parameters were also applied to the evaluation in chapter 5, but yielded very different results than in de Castro's and Von Zuben's work.
L.Pasti and L.N.de Castro proposed a neuro-immune hybrid algorithm [PC06]. A single layer self organized neural network was combined with clonal selection from the AIS. The resulting hybrid algorithm 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. The tested clonal algorithms were not suited to solve TSP with a high amount of nodes. The deviation from optimal score could exceed 3000 percent in case of the most difficult TSP in this setup with over 2000 nodes.

7.1 Future work

The evaluation in this thesis did not test if and how the mutation factor contributes to overall performance in solving TSP. This parameter has been dynamically adjusted in every tested variant inverse to affinity, however, it is not clear if a different default value or adjustment rate could affect performance. Further experiments for testing variations of the mutation factor should be done. The work of [Gar04] has been applied to multimodal problems so far, therefore applying an almost parameter free CLONALG variant to a set of different TSP could yield interesting results. TSP have similarities to pathfinding problems and testing the ability of the clonal selection algorithms in solving pathfinding problems could be done in future research.

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.

[Coo] William Cook. *University of Waterloo, Traveling Salesman Problem*. `http://www.math.uwaterloo.ca/tsp/history/tspinfo/pr2392_info.html`[Accessed: 16.01.2019].

[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.

List of Figures

List of Tables