Adaptability Targeted Elimination, an Evolvability Inspired Approach

September 2016

MSc Artificial Intelligence
Adaptability targeted elimination, an evolvability inspired approach

Abstract

Evolutionary systems in nature are conformed from different species with varying adaptabilities, in some cases, species with higher adaptability can become a hinderance for the environment in which they develop, while there are several examples of this, one that caught the attention of the author is that of the so called "superbugs", which are pathogen carrying bacteria that could end up affecting a wide range of the human population in the planet, the proposition is that by knowing about the mechanisms for the propitiation of evolvability, adaptability can be manipulated in order to have a certain target to be in disadvantage against another. A procedure to explore this proposition was developed for this project, its workings and results are described in this document.
**Abbreviations**
GANN Genetic Algorithm Neural Network
NN Neural Network
GA Genetic Algorithm
EA Evolutionary Algorithm
HGT Horizontal Gene Transfer
DNA deoxyribonucleic acid
Pc crossover probability
Pm mutation probability
HC Hierarchical Crossover
SH Semantic Hierarchy
GVLG General Variable Length Genome
ISCBB inter-species communication between bacteria
# Table of contents

Chapter 1 Introduction........................................................................................................6
  1.1 Introduction................................................................................................................6
  1.2 Structure.....................................................................................................................6
  1.3 Reviewing core concepts............................................................................................6
    1.3.1 A brief revision of GA’s........................................................................................6
    1.3.2 Hierarchical crossover.........................................................................................8
    1.3.3 General Variable Length Genome......................................................................9
    1.3.4 Neural Networks ..............................................................................................10
    1.3.5 GANN................................................................................................................11
    1.3.6 Evolvavility.........................................................................................................12
      1.3.6.1 Two important methods for adaptability in bacteria.................................13
    1.3.7 Cell to cell communication in bacteria..............................................................13
  Chapter 2 Defining the problem.......................................................................................14
    2.1 Defining the problem..............................................................................................14
    2.2 Modularity...............................................................................................................15
    2.3 Crossover................................................................................................................15
    2.4 HGT .......................................................................................................................18
  2.5 Encoding..................................................................................................................19
  Chapter 3 Experiments.....................................................................................................20
    3.1 Experiments............................................................................................................20
      3.1.2 Preliminary steps..............................................................................................20
      3.1.3 Competition......................................................................................................22
    3.1.4 Problems.............................................................................................................24
  Chapter 4.........................................................................................................................24
    4.1 Conclusions.............................................................................................................24
    4.2 Further research.....................................................................................................25
  Bibliography....................................................................................................................26
  Appendix.........................................................................................................................27
CHAPTER 1 INTRODUCTION

1.1 introduction

The main interest of the author in the combination of GA with NN is the intuition that by the implementation of an EA in the design and training of a NN there is the possibility, applying certain state of the art techniques for the analysis of the evolutionary system, to allow complex systems to emerge, and in short, conveying the possibility of the appearance of desired behaviours and properties in evolutionary systems that previously needed to be specified by the users, this interest has led to the research of these methods, in this study, a GANN will be implemented to solve a pattern recognition task and it will serve to model the growth of two species of bacteria.

1.2 structure

In chapter 1 some core concepts pertaining this particular problem will be revised, along with different approaches that were researched in the development of this project, in chapter 2 the problem of this study is defined, in chapter 3 the experiments undertaken are outlined, chapter 4 consists of the conclusions, a brief discussion about the real world relation with the subject and further research and thoughts that came about during the development of this project.

1.3 Reviewing core concepts

1.3.1 A brief revision of GA’s
A Genetic Algorithm is a metaheuristic that falls within the category of Evolutionary Algorithms, they rely on the genetic operators of mutation and crossover to solve optimization problems, they were originally proposed by John Holland in 1975, the basic working of a GA is explained briefly below.

A group of strings is created representing a solution to a problem, a test for fitness is declared, subsequently, the group of strings, called a population, is tested according to the defined test, then they undergo a process of recombination and mutation, generating a new population which is to be tested again, the process repeats itself until convergence to a solution or until a certain number of repetitions (generations) is achieved.
1. Initialize population
2. Evaluate population
3. While (!stop condition) do
   1. Select individuals for reproduction (based on fitness)
   2. Apply crossover and mutation operators (breed new population)
   3. Evaluate fitness of new population

fig(1) pseudo-code for a Genetic Algorithm

As stated above, the main operators of a GA are recombination and mutation.

**Selection**

Although selection is not regarded as an operator, it has a deep impact on the effectiveness of the GA, it focuses in finding the best solutions within the existing population, and when it occurs after the operators it can be used to select for elitism. It ensures that the least fit solutions will have less probability to appear in the next generation.

**Roulette wheel selection**

Also known as fitness proportional selection, it selects an individual with a probability

$$p_i = \frac{f_i}{\sum_j f_j}$$

where M is the number of the individuals, $f_i$ is the fitness of the individual and $f_j$ the fitness of the other j individuals. The comparison to a roulette wheel becomes clear, individuals with higher fitness will have a greater portion of the wheel allocated in their favour, but this does not in any way mean that they will surely pass to the next generation, this can provide a certain degree of variability.

fig(2) an illustration of roulette wheel selection

**Ranking selection**

The population is sorted from high to low fitness value, subsequently, the top $\gamma$ ranked individuals are selected with a probability $p(\gamma)$ which is a ranking function.

**Truncation selection**
The possible solutions are ordered by fitness and then a proportion $d$ of them is selected and reproduced $\frac{1}{d}$ times.

**Tournament selection**

$k =$ tournament size

choose k individuals at random

choose best individual from tournament with probability $p$

choose second best with $p(1-p)$

choose third best with $p((1-p)^2)$

...

**Mutation**

In a GA, mutation is based in biological mutation, it is a mechanism that ensures variability in the population, there are different mutation types:

- Flip bit: takes the genome and flips its bit’s values
- Bit string: a random bit is flipped
- Uniform: a gene is replaced with a uniform random value that lies between an upper and lower bound
- Non uniform: the value of a gene is replaced with a non uniform value
- Boundary: a gene is randomly replaced with the lower or upper bound
- Gaussian: mutation adds a random value that is Gaussian distributed to the gene to undergo mutation.

**Recombination or crossover**

Two strings are paired, which are to be called parents, and undergo a process of recombination of its elements, this could take place in different ways; single point, two point, cut and splice and uniform crossover, which are explained in the following images.

**Single point crossover**

![Figure (3) one point crossover](image-url)
Two point crossover

![Two point crossover diagram]

Cut and splice

![Cut and splice diagram]

This form of crossover varies the lengths of the recombining genomes and does so in a disorderly fashion.

Uniform crossover

![Uniform crossover diagram]

One issue regarding crossover is that sometimes the lengths of both parents are different, for this problem some alternatives were proposed by different authors, in the following sections they will be explained.

1.3.2 Hierarchical crossover

One variant of crossover that has the potential to prove useful in scenarios when it is necessary to account for genomes with varying length is the HC proposed by Bentley et all [1]. In this paper, the authors introduce the concept of a Semantic Hierarchy, which they define as a "tree of meaning", in short, they parted from the idea of pairing two chromosomes of differing lengths by means of a syntax based approach, but found this premise potentially wasteful for extended genomes, in the sense that too many repetitions of labels might occur, what is new about this tree of meaning is that the
hierarchy of the genome generated is independent of the manner in which the genes are stored in memory. In a sample problem with \( N \) genes and \( n \) bits in each gene, a semantic hierarchy will present three levels, starting at the individual (population member), going down to the gene level to finish at the alleles level.

The HC consists of two steps, first, as in standard one-point crossover, in the second stage, if the chromosomes vary in length, the process of looking for a viable point to undergo crossover begin following the steps below.

- Positioned at the individual level, a random branch is selected to proceed to the next node at the next level in individual 1, an equivalent node is selected in individual 2.
- The algorithm picks another branch at the current level in each individual to traverse down and selects two more nodes that correlate with each other. This is repeated until the algorithm reaches the leaves level.
- If at any point the algorithm fails to find a node in individual 2 that corresponds to individual 1 it performs a backtracking step a level up picking a different node in individual 1, and so forth until reaching the first level, if it fails again it determines that crossover cannot take part.

### 1.3.3 General Variable Length Genome

Another example of implementation intended to deal with the problem of variable length in genomes is the GVLG proposed by Lee et al. [2]. The justification is that in some cases, such as the optimization of a NN topology, having a fixed number of genes constrains the maximal number of connections, nodes and inputs limiting the search space. The developers parted from the idea of the canonical genome, briefly explained in the coming table.

<table>
<thead>
<tr>
<th>encoding</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Table(1) an example of the canonic genome

The difference is that for GVLG, the authors developed another kind of two string representation of the genome, but instead of integers, they used an identifying string based on the length of the genome.

<table>
<thead>
<tr>
<th>encoding</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>.1</td>
<td>.2</td>
<td>.3</td>
<td>.4</td>
<td>.5</td>
</tr>
</tbody>
</table>

Table(2) an example of the GVLG

This allows for mutation of the index string, thus allowing for mutations that result in reordering.
Table 3: an example of a mutation in the index string, it can be observed that the index value in bold underwent a mutation from .2 to .32, the resulting index string contains a rearrangement of said element.

Evidently, this approach also proves useful when dealing with variable lengths such as in the next crossover scenario.

Figure (7) an example of the functioning of GVLG, it can be observed that the values were arranged relatively to the value of the index string rather than only the position in the genome.

1.3.4 Neural Networks

Artificial neural networks are a connectionist model that aims to simulate, in an albeit simple manner (it has been argued that biological realism would prove rather unnecessary [20]), the functioning of the central nervous system, so far they have been successful in manifesting some of the real neural networks information processing capabilities such as generalization, learning and error tolerance, even though this approach has been found relevant for certain tasks.

In short, NNs can be envisioned as an assembly of neurons that interconnect to form a network topology, they can be classified in different types depending on the patterns of the connections in the neurons, the mode it operates and on the way the network is trained. [18]
Training
This pertains to the task of finding the correct parameters for the NN, a simple approach is an analogy of curve fitting, if we have a training set and a vector of targets we can minimize the error function, many of the training algorithms can be described as optimization and statistical optimization.

A great part of training algorithms depends on some kind of evaluation of gradient descent employing a backpropagation procedure, this can lead to improvements in terms of the speed to reach the minima of the error function.

Backpropagation
Error backpropagation is the process of passing a message alternately forwards and backwards through the network and it is an efficient technique for the evaluation of the gradient of the error function.

There are some practical considerations to make before applying backpropagation:
- Does the training data needs to be pre-processed?
- choosing the initial weights
- choosing an appropriate learning rate
- the moment in which the weights should be changed
- which activation function is the right one?
- how to avoid local minima in the error function
- how to decide when to stop the training

1.3.5 GANN
Genetic Algorithm Neural Networks consist in the design of a Neural Network solving the task of training the network and the definition of the parameters of said network, again, the inspiration comes from nature.

An important issue to consider in the design of a GANN is the training step, a GA will train the network regardless of how it is connected, whether it be feedforward or feedback, it can also train a mixture of both. All parameters to evolve are represented in a string.

It is also possible to evaluate how neurons interconnect with a GA, the genetic operators also can choose between different activation functions.

For this project the training step will be obviated in this implementation due that the network to be evolved is feed-forward only.
In this project the activation function will consist of simply a threshold, the neuron will fire if the threshold is surpassed by the network value.
1.3.6 Evolvability
Evolvability is defined as the capacity of an evolvable system to generate adaptive genetic diversity, meaning that the system is able to perform adaptive evolution. In recent years, some researchers have proposed that evolvability evolves, and furthermore, is a selectable trait, experiments have shown that a certain notion of evolvability can be observed in the balance of exploration and exploitation as demonstrated in [4]. There are mechanisms in nature that ensure genetic variation and the upkeeping of beneficial characteristics.

There are some mechanisms in place in nature that allow for an increase in the effectiveness and intensity of the acting selection on the variation on phenotypes in a population, some of these are:

- Mating rituals that play the role of a specified sexual selection, this acts as a more intense form of natural selection.
- Shorter periods of breeding
- Greater size of the effective population, which is the number of members an ideal population needs to have so that some specific amount of interest to appear in both ideal and real population.
- Recombination decreasing random linkage disequilibria, which is when two alleles residing at different loci associate in a non-random manner.

In the past, evolution for evolvability was dismissed as causation, given that evolvability can be considered as a characteristic of the future, but studies like [4] have proposed and demonstrated to a certain degree that evolvability can be subjected to selection in a Darwinian fashion.

Modularity and evolvability
It has been proposed by [13] that modularity is an advantage for evolvability, given that if all mutations affected all the traits, this could give rise to mutations that are an asset for some characteristics while hindering another, in theory it would mean that there would be almost no beneficial mutation, but if in contrast, the mutations effects are restricted to act within a certain module, they would affect less traits each time. In the example of a modular network, such as the one evolved in [21], a gene that in a way activates a set of other genes may evolve more rapidly than other gene that activates as well other genes with traits that are not under selection.

In the present work, the author is not interested in proving that evolvability can be evolved, only in having some notion of it in the comparison of the different ways a population develops over time in relation to changes in its environment, this is the reason as to why a measure of evolvability for this particular case was not developed.

1.3.6.1 Two important methods for adaptability in bacteria
There are two important operators in bacteria that ensure variability; Mutation and HGT.

**Horizontal Gene Transfer in bacteria**

HGT is the main reason for the wide presence of antibiotic resistance in bacteria, it is important for the evolution of these kind of microorganisms that can degrade man made substances like drugs, it is also responsible for the transmission, maintenance and evolution of virulence, it tends to involve plasmids and bacteriophages, it has been argued that HGT is the main form of genetic transfer in single celled organisms.

By means of HGT, a bacterium can obtain new functions, this takes place by several mechanisms, such are:

- Gene transfer agents: these are virus-like elements produced by bacteria that are encoded by the host bacteria and package random segments of this bacteria’s DNA which are to be transduced into another cell which will act as recipient.
- Bacterial Conjugation, which consists in the transfer of DNA in the form of plasmids between a donor to a recombinant cell during physical contact.
- Transformation, exogenous DNA is uptaken by the cell and assimilated.
- Transduction, a bacteriophage introduces foreign DNA into a cell.

**About plasmids**

A plasmid is a molecule of DNA that is separated from the chromosomal DNA and can replicate independently, they tend to carry useful genes for the host cell, such as antibiotic resistance, they are capable of reproducing autonomously in a suitable host, they can be transmitted from one bacteria to another by means of three of the mechanisms mentioned above; conjugation, transduction and transformation.

Plasmids carry at least one gene most of the time, some of said genes contain traits that will help for the survival of the cell, such as virulent factors that could enable the bacteria to colonize a host, or functions that will permit the bacteria to process a certain nutrient, including toxic molecules, nevertheless, some plasmids carry no particular function, or none that can be discerned by now.

Plasmids make themselves necessary,

**1.3.7 Cell to cell communication in bacteria**

Bacteria utilize chemical molecules as signals for communication, these are necessary for the synchronization of the activities of extensive groups of cells, in bacteria, this communication requires the production, release, detection and response generation to hormones called autoinducers, this is termed quorum sensing [15] and permits bacteria to sense its environment and be on the lookout for other bacteria and change its behaviour at the population level as response to alterations in the number of members.
and their respective species in the environment. This allows for inter-species communication between bacteria.

**Inter-species communication between bacteria**

Quorum sensing also allows for the communication between bacteria of the same and different species, this depends on different concentration of substances that are for the moment being of no concern for the problem addressed in this work, what is of interest, is the possibility of supressing this communication, action which would have the effect of preventing the contact necessary for HGT in bacteria from different species, in nature, when developing in scenarios of scarce resources, the capacity for an organism to supress this sensing in a competing species could provide a priceless advantage in the race for survival, in the same way, this provides an interesting alternative to antibiotics, given that, if communication is supressed, adaptation in bacteria would be reduced, giving way to strategies that do not rely on bactericidal substances.

**CHAPTER 2 DEFINING THE PROBLEM**

2.1 defining the problem

As mentioned above in the abstract, the objective of this project is to explore the possibility of targeting a specific population of individuals in a simple example and achieving the goal of removing them from the population in the present environment, the real world example to take in consideration is that of the superbugs because it is an increasing problem throughout the world, propitiated by the misuse of antibiotics by the general population and the different livestock industries, some doctors have warned of an antibiotics crisis in the following decades, the aim of this work is to explore the possibility of controlling a population of bacteria by means of manipulating the presence of objects in its environment in order to take them to a certain state of evolvability incompatible with a new environment.

The example system consists of a feed-forward neural network that has to solve a pattern recognition task consisting in identifying a set of objects present in a 4 by 8 grid which is divided in two, the individual then has the task of correctly identify whether it is perceiving a left, right or both objects in the grid, the goal is to perform a logic operation with both objects. These objects are supposed to represent available resources in the observable environment of the individuals, in order for the individual to replicate, it has to process these objects according to a logic operation, the objects per se are not the resources but just objects in the environment that the individuals have to process in order to take a figurative resource, which will serve as a limiting variable for the reproduction of bacteria during competition.

The way this is related to the philosophy mentioned previously is in the sense of making the goal population compete with a control population for resources.

In this problem we will obviate the antibiotic application step, as the procedure is intended to avoid the introduction of antibiotics so as to not allow either of the two
species to develop resistance, instead, the experiments are designed to force the populations to compete for resources, if the experiments succeed, the resulting environment will contain mostly if not only individuals from the control species that biologists could ensure are not resistant to a given antibiotic, so that in the moment it wins the competition it can be cleaned from the environment with the specific drug needed, thus rendering the environment free from the two populations. The target population will be trained to search for a greater variety of resources.

Furthermore, as it is possible to develop ways to quell bacterial communication [15] between individuals of different species, it is safe to a certain degree to assume the isolation between the two competing species of this particular case, thus ignoring the need for a mechanism that ensures this specific phenomenon.

The steps of the trial are the following:
1. Evolve a GANN within a constant goal, this is to be the control population.
2. Evolve a GANN within a changing environment, thus allowing for the target population to evolve for evolvability and remain in an exploitation state, searching for variability and acquiring new genetic material from neighboring individuals, creating more genetic ballast that drains its resources.
3. Evolve a mixture of both populations in an environment with plentiful resources for the control population and varying resources for the target population, the target population will need to learn to process a new set of objects with the objective of forcing the individuals in said population to exchange genetic code and store it in plasmids.

This will be explained in more detail in chapter 3.

2.2 Modularity
As proposed in [21], by changing the goal state operation as evolution takes place, the amount of time required to reach the goal is reduced and the evolved network displays a degree of modularity, allowing for ease of adaptation within a changing environment, furthermore, the evolved networks tend to have more neurons, which is important for the experiment, as it hypothesized for this experiment that a larger genome could potentially prove a liability when competing with a faster reproducing population, this will be explained in more detail in Chapter 3.

2.3 Crossover
The experiments were performed with three kinds of crossover, being one point crossover, uniform and a variation of uniform that makes use of the first component of the nominal code described above, a seed vector was created with the maximal number of genes and then a vector with the size of the actual genome filled with a translation which converted the seed vector into a vector of zeros and ones, the ones represented the full length of a gene, which is a neuron with its threshold, connections and weights. Due to the nature of the objective, the crossover variant that was chosen was the variant of uniform crossover developed for this example, as it requires complete genes to be shared between individuals, just as an exploratory measure, two other kinds of
crossover were performed, in order to assess their effectiveness in this particular problem, the comparison can be observed in the upcoming tables.

<table>
<thead>
<tr>
<th>Individual</th>
<th>fitness bottom to top</th>
<th>Fitness w/penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>47</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>87</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>54</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>57</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>58</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>66</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>123</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>257</td>
<td>1</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Table (4) fitness values for GANN with uniform crossover
<table>
<thead>
<tr>
<th>Individual</th>
<th>fitness botom to top</th>
<th>fitness w/penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>45</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>57</td>
<td>0.99</td>
<td>0.96</td>
</tr>
<tr>
<td>98</td>
<td>1</td>
<td>0.96</td>
</tr>
<tr>
<td>140</td>
<td>0.98</td>
<td>0.96</td>
</tr>
<tr>
<td>201</td>
<td>0.97</td>
<td>0.96</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>2</td>
<td>0.98</td>
<td>0.97</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>47</td>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>51</td>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>52</td>
<td>1</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table (5) fitness values for GANN with uniform crossover
The later of the two kinds of crossover was selected because it bears more similarity with the homologous transference occurring in cells.

An adaptation of the canonical genome and the GVLG index was implemented for this problem, each position in the genome is identified with a number that represents the role of each allele in the network, the value 1 represented the threshold position in a neuron, the index 2 represented an address to connect with and the value 3 represented the weight of the connection, likewise, another string identified the layer in which the neuron was located, this served for the purposes of comparing genomes of different length, which was necessary in the crossover step.

2.4 HGT
As it was stated in the introduction section, HGT plays a fundamental role in the evolvability of bacteria, as it allows for a greater variability in populations compared with systems that allow only for vertical gene transfer. In this experiment, the HGT mechanism to be emulated is bacterial conjugation, because it can be simplified in the crossover step, and does not rely on a bacteriophage, likewise, it can be easily represented as a matrix consisting of a pool of segments of different genomes, another important issue regarding this mechanism is the necessity of replicating the plasmids, which consume valuable energy, thus potentially hindering the development of the population.

A preliminary form of HGT is implemented in each iteration so as to speed up evolution, this is done by randomly pairing the individuals in the population with another member and performing crossover, arguably, the crossover step could also represent the transduction method, because for all practical purposes, the cell is assimilating a foreign segment of DNA, only this process is sped up because all the present cells are undergoing the process, in order to make this crossover step more akin to bacterial HGT, the Pg was set to be in function of the difference in length of both parental genomes, the more similar they were, the more chances they had to undergo recombination, interestingly, time of convergence to a perfect solution is longer in this case, this reasserts the affirmations that homologous recombination serves more to repair failures than to generate variability.

Plasmids
Plasmids are present in the form of a multidimensional matrix with number of pages equal to the number of runs, in order to emulate nature to a certain degree, plasmids reproduce independently of the host (inside the host) with a high probability, plasmids are gathered from a random individual in each iteration with a probability depending on the Hamming distance of the array of successful attempts of processing the 100 values in the observable space per iteration, this in order to emulate the biological function of plasmids of giving a new functionality to the host cell.
2.5 Encoding
The parameters of the NN are encoded in the genome, which is as follows; the genome consists of 103 bits, which encode 15 neurons, the following table explains in more detail the encoding.

The nominal code
This code was created in order to identify the function of a particular locus, thus determining the upper and lower limits of the range of values in which to perform the mutation operation. This served as some form of header to the genome. This is explained in the table below.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>1</th>
<th>n(1...4)</th>
<th>layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>2</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>3</td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

Table (6) nominal index code

Encoding the neurons

<table>
<thead>
<tr>
<th>A neuron</th>
<th>Layer 1</th>
<th>Layer n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>threshold</td>
<td>address in observable space</td>
</tr>
<tr>
<td></td>
<td>layer n-1</td>
<td>connection weight</td>
</tr>
</tbody>
</table>

Table (7) encoding of the neurons

Some mechanisms were designed in order to ensure that the connection alleles referred to an existing neuron, first, a mechanism for the shifting of the unused neurons to the far right of the layer was implemented.

```
for tem=1:3
    la=find(newPop(:,3)==tem);
    temp1=find(isnan(newPop(la,elites+ii+3)))
    temp2=find(~isnan(newPop(la,elites+ii+3)))
    newPop(la,elites+ii+3)=newPop(la,temp2,elites+ii+3)'
    newPop(la(templ),elites+ii+3)='
end
```

This segment of code solves the issue of connections to inexisting neurons in an upper layer, in a semi fixed size GANN this easily solves the problem without the need of a complicated hierarchical or semantic code.

Likewise, for the tests with uniform crossover, a mechanism was set in place to ensure that
Chapter 3 Experiments

3.1 Experiments
Different approaches will be tested in order to demonstrate the following hypothesis:

For the controlling population:
- In an unchanging environment, the evolved networks will have fewer neurons
- The time to reach a perfect solution increases, but it can be addressed by reducing the set of recognizable objects.
- When the members of the sets of right and left objects on the observable space is reduced, the number of correct recognitions rises

For the targeted population
- In a variating environment convergence to a solution is faster
- Although the resulting networks are bigger
- If the set of perceivable objects is increased, the networks should get bigger and the time to convergence will grow.

For the competition step
- A simple model will be developed in order to simulate the competition between the two non communicating species:

We assume that the speed in which bacteria reproduce depends on their ability to process resources, this is to say that for example, if we assume a standard reproduction cost in time units is 1 for a 100% efficient bacteria, for a bacteria of 90% efficiency it will go up to 1.11 units, furthermore, subtracting to the efficiency, there is the extra weight in the form of connections surpassing a certain threshold, in this case 9, for greater networks, each extra gene will cost a .01 penalty in efficiency, this is to say, that for the previous example, if the less efficient individual at 90% efficiency had two more genes than the threshold, it would need a total of 1.14 time units.

\[
\text{Time units needed} = \frac{1}{f - .01ex}
\]

Where f is fitness and ex represents the extra number of neurons per individual.

3.1.2 Preliminary steps
For the first stage of the trial, the step in which a GANN is evolved in an unchanging environment, the premise was that the network would evolve with fewer neurons and would take more time to converge to a result, in most of the trials, a perfect result was achieved by around 98% of a population of 300 individuals at 100 generations. Subsequently, the GANN was evolved in a variable environment.
Table (8) evolving the first population, this one evolves in a fixed environment, the table shows the values of 9 evaluations of the algorithm, each of 100 generations, it can be noted that in average, the individuals that reach a perfect solution are more than in the next case, also, the time units needed to reproduce are fewer.

For the second stage, a GANN is evolved in a changing environment, the premise was that as the network had to look for more objects the number of correct answers would decrease and the number of neurons needed would increase.

Table (9) evolving the second population in a changing environment and with more objects to recognize, the table shows the values of the number of individuals achieving a perfect solution over the course of 100 generations, it can be observed that this number is lower on average, on the other hand, the average time needed to undergo reproduction based on the scheme proposed before increases in relation with the other population.
### 3.1.3 Competition

A theoretical competition is proposed in which both populations evolve with limited resources and time, evidently, the one that is less effective processing resources will be at disadvantage, as can be noted in the following tables.

<table>
<thead>
<tr>
<th>Lowest effectiveness</th>
<th>2.941176</th>
<th>2.857143</th>
<th>2.857143</th>
<th>2.702703</th>
<th>3.030303</th>
<th>...</th>
<th>...</th>
<th>1.020408</th>
<th>1.020408</th>
<th>1.020408</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.777778</td>
<td>1.098901</td>
<td>1.075269</td>
<td>1.111111</td>
<td>1.075269</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.020408</td>
<td>1.020408</td>
</tr>
<tr>
<td>1.941176</td>
<td>1.162791</td>
<td>1.176471</td>
<td>1.162791</td>
<td>1.149425</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.030928</td>
<td>1.030928</td>
<td>1.030928</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.631579</td>
<td>2.631579</td>
<td>2.564103</td>
<td>2.564103</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.075269</td>
<td>1.075269</td>
<td>1.075269</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.702703</td>
<td>2.702703</td>
<td>2.777778</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.010101</td>
<td>1.020408</td>
</tr>
<tr>
<td>2.857143</td>
<td>2.857143</td>
<td>2.941176</td>
<td>2.941176</td>
<td>2.702703</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.020408</td>
<td>1.010101</td>
</tr>
<tr>
<td>3.030303</td>
<td>2.702703</td>
<td>2.777778</td>
<td>2.777778</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.030928</td>
<td>1.030928</td>
</tr>
<tr>
<td>2.702703</td>
<td>1.851852</td>
<td>1.851852</td>
<td>1.851852</td>
<td>1.818182</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
<tr>
<td>3.030303</td>
<td>2.702703</td>
<td>2.777778</td>
<td>2.702703</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.702703</td>
<td>2.941176</td>
<td>2.777778</td>
<td>2.702703</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
<tr>
<td>3.030303</td>
<td>2.857143</td>
<td>2.857143</td>
<td>2.857143</td>
<td>2.941176</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.010101</td>
<td>1.020408</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.631579</td>
<td>2.777778</td>
<td>2.857143</td>
<td>2.702703</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.020408</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.777778</td>
<td>2.941176</td>
<td>2.777778</td>
<td>2.631579</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.030928</td>
<td>1.020408</td>
<td>1.020408</td>
</tr>
<tr>
<td>2.857143</td>
<td>3.030303</td>
<td>2.702703</td>
<td>2.857143</td>
<td>2.702703</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.777778</td>
<td>2.777778</td>
<td>2.777778</td>
<td>2.857143</td>
<td>2.702703</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.030928</td>
<td>1.030928</td>
<td>1.041667</td>
</tr>
<tr>
<td>1.123596</td>
<td>1.123596</td>
<td>1.123596</td>
<td>1.123596</td>
<td>1.123596</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.020408</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.857143</td>
<td>2.941176</td>
<td>2.857143</td>
<td>2.325581</td>
<td>2.272727</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.020408</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.777778</td>
<td>2.941176</td>
<td>2.857143</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.030928</td>
<td>1.020408</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.631579</td>
<td>2.702703</td>
<td>2.631579</td>
<td>2.857143</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.098901</td>
<td>1.098901</td>
<td>1.075269</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.857143</td>
<td>2.857143</td>
<td>2.857143</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.380952</td>
<td>1.388889</td>
<td>1.388889</td>
<td>1.408451</td>
<td>1.388889</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.020408</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.702703</td>
<td>2.857143</td>
<td>3.030303</td>
<td>2.702703</td>
<td>2.777778</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.777778</td>
<td>2.857143</td>
<td>2.777778</td>
<td>2.777778</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.030928</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.777778</td>
<td>1.162791</td>
<td>1.162791</td>
<td>1.162791</td>
<td>1.149425</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
<tr>
<td>2.777778</td>
<td>2.941176</td>
<td>2.857143</td>
<td>2.857143</td>
<td>2.777778</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.020408</td>
<td>1.010101</td>
</tr>
<tr>
<td>2</td>
<td>2.040816</td>
<td>2.040816</td>
<td>1.960784</td>
<td>2</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>1.020408</td>
<td>1</td>
</tr>
<tr>
<td>2.777778</td>
<td>2.777778</td>
<td>2.777778</td>
<td>2.777778</td>
<td>2.777778</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.030928</td>
<td>1.041667</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.941176</td>
<td>2.777778</td>
<td>2.777778</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.020408</td>
<td>1.020408</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.777778</td>
<td>1.694915</td>
<td>1.724138</td>
<td>1.724138</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.041667</td>
<td>1.010101</td>
<td>1.020408</td>
</tr>
<tr>
<td>2.777778</td>
<td>2.857143</td>
<td>2.777778</td>
<td>2.857143</td>
<td>2.777778</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.030928</td>
<td>1.041667</td>
<td>1.052632</td>
</tr>
<tr>
<td>2.857143</td>
<td>2.702703</td>
<td>2.941176</td>
<td>2.777778</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.030928</td>
<td>1.030928</td>
<td>1.020408</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.857143</td>
<td>2.857143</td>
<td>2.857143</td>
<td>2.777778</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.020408</td>
<td>1.020408</td>
<td>1.030928</td>
</tr>
<tr>
<td>2.941176</td>
<td>2.857143</td>
<td>2.777778</td>
<td>2.777778</td>
<td>2.857143</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1.010101</td>
<td>1.010101</td>
<td>1.010101</td>
</tr>
</tbody>
</table>

Table (10) effectiveness of target population over generations
In this document the author explored a scenario where a species is forced to adapt to a changing environment while a competing species experiments the same scenario without being affected by the changes, thus creating a perfect situation for the disadvantage of the first population, this model requires for the cancelation of interspecies communication between the two species, despite that the penalties for extra chromosomal code by the side of the target population, the need to adapt to search for
more patterns than the controlling population proved enough to hinder the effectiveness of the population, thus rendering it vulnerable to losing resources to the other population.

The development of this project has served to open the eyes of the author to complications in the design of GANNs, and also the complexity of trying to faithfully model biological processes based on assumptions and looking for similarities, additionally, the author has also noted a requirement of programming knowledge needed to attain for future developments.

During the research stage of this project the author came in contact with different interesting subjects about AI and Biology.

Further research in biology is needed in order to make a more efficient model of plasmids in terms of functionality inheritance and the generalization of their workings.

**4.2 Further research**

Although for this case it was assumed that a mechanism was in place in order to quench the interspecies communications coming from the targeted population, it would be of interest to evolve a way of disallowing said communication rather than have it designed manually, but this kind of knowledge greatly surpasses that of the author of this document.

Additionally, further expansion of this work could certainly be the creation of an evolving environment as a manner of coevolution in order to determine the best series of steps that keep the target population interested in searching for new strategies while efficiently managing the resources, the encoded variables over which genetic operators would act upon could be the number of members of both left and right sets, the cycle of environmental changes for the target population and the rate by which the individuals acquire plasmids.

During the coding of the project, the nominal indexes researched and introduced arose curiosity as to an encoding in order to allow the very code to evolve, this, in conjunction with decomposable goals and some form of comparison between different codes performing similar tasks or decomposed tasks could prove an interesting way of generating error checking mechanisms or the understanding of how a particular answer or set of answers evolved in time in function of the specifications of the task at hand or the very way the designer made changes on the go, if this is attainable the author believes it would have potential uses in automatic programming and network and evolving systems analysis.

**Bibliography**


[5] Deem M et al “Life has evolved to evolve”


Apendix

Generic code for any population

```matlab
clear all
runs=100;
pop=300;
elites=0;
Pplasint=.5;
Pm=.5;
initialNeurons=[8 4 2 1];
lengthOfRetinaSet = 100;
maxConnections=[3 3 3 2];
assemblyOfInitialRetinas = zeros(lengthOfRetinaSet,11);
newPop = zeros(106 , pop+3);
%assemblyOfInitialValues
	tic
PlasMatr=zeros(106,pop,runs-1);
pruebas=10;
resultados=zeros(pruebas,pop);

SetOfLeftObjects= [1 1 1; 0 1 1; 0 1 0 1; 1 1 0 1; 0 0 0 1; 1 0 1 1; 0 1 0 0; 1 1 1 0];
SetOfRightObjects= [0 1 1 1; 1 1 1 1; 1 1 0 1; 1 0 1 0; 1 0 1 1; 0 0 1 0; 1 1 0 0; 1 0 0 0];

% SetOfLeftObjects= [1 1 1 1; 0 1 0 1; 1 0 1 0; 1 0 1 0];
% SetOfRightObjects= [0 1 1 1; 1 1 1 1; 1 1 0 1];
IO=[0 0 0; 1 1 0; 2 1 0; 3 1 1];

% prueba =1;
% while (1)
%     if prueba>pruebas
%         break
%     end
%     if prueba>pruebas
%         break
%     end
for qqq=1:7:100
    genotype(qqq,2)=1;
    for k=1:2:5
        ppp=qqq;
        ppp=ppp+k;
        genotype(ppp,2)=2;
    end
    for h=2:2:6
        ppp=qqq;
        ppp=ppp+h;
        genotype(ppp,2)=3;
    end
end
for hhh=1:103
    if (hhh>0) && (hhh<57)
        genotype(hhh,3)=1;
    end
    if (hhh>56) && (hhh<85)
        genotype(hhh,3)=2;
    end
    if (hhh>84) && (hhh<99)
        genotype(hhh,3)=3;
    end
    if (hhh>98) && (hhh<104)
```

genotype(hhh,3)=4;
end
end

for n=1:4
den=(find(genotype(:,3)==n));
len=length(den);
start=den(1);
cod=linspace(1/len,1,initialNeurons(n));
lenCod=length(cod);
for wowo=1:length(cod)
nominalCode(floor(wowo*lenCod-lencod+1):floor(wowo*(lenCod)-lencod)+(lencod))=cod(wowo);
genotype(start:start+length(nominalCode)-1,1)=nominalCode';
end
end

genotype=genotype(1:103,:);
lengthofgen=size(genotype);
lengthofgen=lengthofgen(1);
thresholdsInd = [8 8 4 2];
weigths = [-1 1 ];
TTT = [4 3 2 1 0 -1 -2 -3; 4 3 2 1 0 -1 -2 -3; 2 1 0 -1 0 0 0 0; 1 0 0 0 0 0 0];

%this creates the reference of the number of elements to be connected in
%each layer in lookfor, the first is fixed to 8 because it is the number of
%cells in the retinas grid
LF1=find(genotype(:,3)==1);
LF1=sum(genotype(LF1,2)==1);
LF2=find(genotype(:,3)==2);
LF2=sum(genotype(LF2,2)==1);
LF3=find(genotype(:,3)==3);
LF3=sum(genotype(LF3,2)==1);
lookfor=[8 LF1 LF2 LF3];

for ja=4:pop+3
%ja=1:103
    for je=1:103
        ge=genotype(je,2);
        ref=genotype(je,3);
        %je=4:303
        if je == 1
            neuronsIn1=randi([4 8]);
            neuronsIn2=randi([2 4]);
            neuronsIn3=randi([1 2]);
            conInd=[neuronsIn1 neuronsIn2 neuronsIn3];
        end
        if ge==3
            genotype(je,ja)=randi([1 2]);
        end
        if ref == 1
            if ge==1
                genotype(je,ja)=randi([1 thresholdsInd(ref)]);
            end
            m=randi([10 8],1,2);
            n = randi([1 8]);
            if m(1)==m(2)
                n = randi([1 8]);
            end
            genotype(je+1,ja)=m(1);
            genotype(je+3,ja)=m(2);
            genotype(je+5,ja)=n;
        end
        if ref ~= 1
            if ge==1
                genotype(je,ja)=randi([1 thresholdsInd(ref)]);
            end
            if ref == 4
                m=randi([0 conInd(ref-1)],1,2);
                n = randi([1 conInd(ref-1)]);
                if m(1)==m(2)
                    n = randi([1 conInd(ref-1)]);
                end
            end
        end
    end
end
genotype(je+1,ja)=m(1);
genotype(je+3,ja)=m(2);
genotype(je+5,ja)=n;
end
if ref ==4
m=randi([1 conInd(ref-1)],1,2);
genotype(je+1,ja)=m(1);
genotype(je+3,ja)=m(2);
end
end
end

% Erase extra neurons
if je==103
for er=1:3
if conInd(er)==initialNeurons(er)  
La=find(genotype(:,3)==er);
ErIn = 7*(initialNeurons(er)-conInd(er));
genotype(max(La)-ErIn+1:max(La),ja)=nan;
end
end

genotype(104:106,ja)=conInd';
end
end

% this creates the set of lengthOfRetinaSet number of retinas to test the population with

assemblyOfInitialRetinas(:,1:8)=randi([0 1],lengthOfRetinaSet,8);
A=bi2de(assemblyOfInitialRetinas(:,1:4));
B=bi2de(assemblyOfInitialRetinas(:,4:8));
ZA=bi2de(SetOfLeftObjects);
ZB=bi2de(SetOfRightObjects);
assemblyOfInitialRetinas(:,9)=ismember(A,ZA);
assemblyOfInitialRetinas(:,10)=ismember(B,ZB);

%here the first population is evaluated
% first we create the phenotype

phen=zeros(lengthofgen,pop);
y=length(genotype);
for cons = 1:lengthofgen
    tr=genotype(cons,3);
    w=genotype(cons,4:y);
    T=find(~isnan(w));
    TTT=TTT(tr,w(1,T));
    phen(cons,:)=w;
    break
end
% now translate the weight of the connection weights
while genotype(cons,2) ==3
    w=genotype(cons,4:y);
    T=find(~isnan(w));
    w(1,T)=weights(1,1,w(1,T));
    phen(cons,:)=[w;
    break
end

results=zeros(lengthOfRetinaSet,pop);
for ret=1:lengthOfRetinaSet
    % translate the address of the retinas in the genotype into the values of the retinas taken
    for it1=1:lengthofgen
        while genotype(it1,2) == 2 &amp; amp; genotype(it1,3) == 1

end
    %else
    %break
end
    %for ret=1:lengthOfRetinaSet
    %results(ret,:)=phen(assemblyOfInitialRetinas(ret,:),:);
end

w = genotype(it1,4:y);
T = find(~isnan(w) & w > 0);
w(T) = assemblyOfInitialRetinas(ret,w(T));
phen(it1,:) = w;
break
end
end
c1 = 1;
c2 = 1;
c3 = 1;

% feed forward
for it = 1:103
tr = genotype(it,2);
% Evaluate the first layer of neurons
if tr == 1 & genotype(it,3) == 1
Lay1Values(ct1,:) = phen(it+1,:) * phen(it+2,:) + ...
phen(it+3,:) * phen(it+4,:) + ...
phen(it+5,:) * phen(it+6,:);
ct1;
G(1,:) = Lay1Values(ct1,);
I(1,:) = phen(it,:);
G(G(1,:) < I(1,:)) = 0;
G(G(1,:) > 0 & ~isnan(G(1,:))) = 1;
Lay1Values(ct1,:) = G;
c1 = c1 + 1;
end

% now we translate the addresses for connections of layer 2
if c2 == 1 & it == 56
La = find(genotype(:,3) == 2);
Ge = find(genotype(La,2) == 2);
waw = genotype(La(Ge),4:y);
for m = 1:pop
T = find(waw(:,m) > 0 & ~isnan(waw(:,m)));
waw(T,m) = Lay1Values(waw(T,m),m);
end
phen(La(Ge),:) = waw;
c2 = c2 + 1;
end

% Second layer
if tr == 1 & genotype(it,3) == 2
Lay2Values(ct2,:) = phen(it+1,:) * phen(it+2,:) + ...
phen(it+3,:) * phen(it+4,:) + ...
phen(it+5,:) * phen(it+6,:);
ct2;
G(1,:) = Lay2Values(ct2,);
I(1,:) = phen(it,:);
G(G(1,:) < I(1,:)) = 0;
G(G(1,:) > 0 & ~isnan(G(1,:))) = 1;
Lay2Values(ct2,:) = G;
c2 = c2 + 1;
end

% now we translate the addresses for connections of layer 3
if c3 == 1 & it == 84
La3 = find(genotype(:,3) == 3);
Ge3 = find(genotype(La3,2) == 2);
waw3 = genotype(La3(Ge3),4:y);
for m = 1:pop
T = find(waw3(:,m) > 0 & ~isnan(waw3(:,m)));
waw3(T,m) = Lay2Values(waw3(T,m),m);
end
phen(La3(Ge3),:) = waw3;
c3 = c3 + 1;
end

% Third layer
if tr == 1 & genotype(it,3) == 3
Lay3Values(ct3,:) = phen(it+1,:) * phen(it+2,:) + ...
phen(it+3,:) * phen(it+4,:) + ...
phen(it+5,:) * phen(it+6,:);
ct3;
G(1,:) = Lay3Values(ct3,);
I(1,:) = phen(it,:);
G(G(1,:) < I(1,:)) = 0;
G(G(1,:) > 0 & ~isnan(G(1,:))) = 1;
Lay3Values(ct3,:) = G;
ct3=ct3+1;
end

% now we translate the addresses for connections of layer 4
if it==98
    La4=find(genotype(:,3)==4);
    Ge4=find(genotype(La4,2)==2);
    waw4=genotype(La4(Ge4),4:y);
    for m=1:pop
        T=find(waw4(:,m)~=0 & ~isnan(waw4(:,m)));
        waw4(T,m)=Lay3Values(waw4(T,m),m);
        wot4(:,m)=waw4(:,m);
    end
    phen(La4(Ge4),:)=wot4;
end

% Fourth layer
if tr == 1 && genotype(it,3) == 4
    Lay4Values(1,:) = phen(it+1,:).*phen(it+2,:)+...  
        phen(it+3,:).*phen(it+4,:);
    G(1,:)=Lay4Values(1,:);
    I(1,:)=phen(it,:);
    G(G(1,:)<I(1,:))=0;
    G(G(1,:)~=0)=1;
    Lay4Values(1,:)=G;
end

% results(ret,:)=Lay4Values;
expectedVal(ret,1)=IO(find(IO(:,1)==assemblyOfInitialRetinas(ret,11)),2);
assemblyOfInitialRetinas(ret,12)=expectedVal(ret,1);
Corrects(ret,Corr)=1;
end

C2=sum(Corrects);
fit(1,:)=linspace(1,pop,pop);
fit(2,:)=C2.*.01;
% [y,i]=sort(fit(2,:));
% b=fit(:,i);
% apply a fitness penalty for the genomes with more than 13 neurons
LenPen=sum(~isnan(phen(1:98,:)))./7;
LenPen(LenPen<9)=0;
LenPen(LenPen>9) = (LenPen(LenPen>9) - 9)*.01;
fit(3,:) = fit(2,:)-LenPen;

% Copy nominal code into new population
newPop(:,1:3)=genotype(:,1:3);
ConLims=zeros(4,pop);
ConLims(1,:)=8;

% genotype2=zeros(106,303);
run=1;

while(1)
    % while resources >
    if run>runs
        break
    end
    % elitism
    [zz,i]=sort(fit(3,:));
    b=fit(:,i);
    newPop(:,elites+4:pop+3)=genotype(:,b(1,1:pop-elites)+3);
newPop(:,4:elites+3)=genotype(:,b(1,(pop-elites+1):pop)+3);

parent2ind=randi([1 pop-elites],1,pop-elites);
pDonorIndex=randi([1 pop-elites],1,pop-elites);

for ii=1:pop-elites
    %compare both parents, this will give us a "probability" of crossover
    pointsVector=zeros(105,1);
    pointsVector2=zeros(105,1);
    cpl=length(find(~isnan(newPop(1:103,ii+elites+3))));
    cp2=length(find(~isnan(newPop(1:103,pDonorIndex(1,ii)+elites+3))));
    Pc=(103-abs(cpl-cp2))/103;
    if rand(1)<=Pc
        pointsVectorSeed=randi([0 1],15,1);
        seeds=(find(pointsVectorSeed==1))*7-6;
        for vi=1:length(seeds)
            pointsVector(seeds(vi):seeds(vi)+6,1)=1;
        end
        genotype(find(pointsVector==1),ii+elites+3)=newPop(find(pointsVector==1),ii+elites+3);
        genotype(find(pointsVector==0),ii+elites+3)=newPop(find(pointsVector==0),pDonorIndex(1,ii)+elites+3);
    else
        genotype(:,elites+ii+3)= newPop(:,elites+ii+3);
    end
end

% if rand(1)<=Pplasint || run > 1
end

if run > 1
    % plasmid inheritance
    % this is the point individual plasmids in each cell have to surpass
    % in order to be inherited
    PplasmidInh=randi([4 8])*0.1;
    % this is the individual probability to be transferred, it was set
    % from 0.5 to 1 because in nature plasmids tend to have mechanisms to
    % ensure their replication
    Pindplas=randi([5 10],1,runs-1)*0.1;
    plasIndex=find(Pindplas>PplasmidInh);
    o=PlasMatr(:,ii,plasIndex);
    PlasMatr(:,ii,plasIndex)=o;
end

% plasmid acquisition
    cpa1=length(find(~isnan(newPop(:,ii+elites+3))));
    cpa2=length(find(~isnan(newPop(:,pDonorIndex(1,ii)+elites+3))));
    Pplas=(103-abs(cpa1-cpa2))/103;
    if rand(1)<=Pplas
        pointsVectorSeed2=randi([0 1],15,1);
        seeds2=(find(pointsVectorSeed2==1))*7-6;
        for vii=1:length(seeds2)
            pointsVector2(seeds2(vii):seeds2(vii)+6,1)=1;
        end
        PlasMatr(find(pointsVector2==1),ii+elites,run)=newPop(find(pointsVector2==1),pDonorIndex(1,ii)+elites+3);
        PlasMatr(106,ii+elites,run)=length(seeds2);
    end

% switch inactive genes to the end of the layer
for tem=1:3
    la=find(genotype(:,3)==tem);
    temp1=find(isnan(genotype(la,elites+ii+3)));
    temp2=find(~isnan(genotype(la,elites+ii+3)));
    genotype(la,temp1,elites+ii+3)=[genotype(la,temp2,elites+ii+3)' ];
end
for k=1:3
    in=find(genotype(:,3)==k);
    genotype(103+k,4:pop+3)=sum(~isnan(genotype(in,4:pop+3)))/7;
    ConLims(k+1,:)=genotype(103+k,4:pop+3);
end

%mutation
Muts(1,:)=rand(1,pop);
vec = [-1 1];
for q=1:pop
    if Muts(1,q)>Pm
        x =randi([1 2]);
        x= vec(x);
        point=randi([1 103]);
        genotype(point,q+3)= genotype(point,q+3) + x;
        if genotype(point,2)==1
            maxt=max(TTT(genotype(point,3)));
            maxt=thresholdsInd(genotype(point,3));
            minC=1;
            if genotype(point,q+3) > maxt
                genotype(point,q+3)=minC;
            end
            if genotype(point,q+3) < minC
                genotype(point,q+3)=maxt;
            end
        end
        if genotype(point,2)==2
            index=genotype(point,3);
            maxC=ConLims(index,q);
            minC=1;
            if genotype(point,q+3) > maxC
                genotype(point,q+3)=minC;
            end
            if genotype(point,q+3) < minC
                genotype(point,q+3)=maxC;
            end
        end
        if genotype(point,2)==3
            if genotype(point,q+3) == 3
                genotype(point,q+3) =1;
            end
            if genotype(point,q+3) == 0
                genotype(point,q+3) = 2;
            end
        end
    end
end

%eliminate connections that refer to inexistent neurons, acquired through crossover
for e=2:4
    j=find(genotype(:,2)==2 & genotype(:,3)==e);
    %compare=repmat(genotype(103+e,4:303),length(j),1);
    inter=(genotype(j,4:pop+3));
    inter(inter>repmat(genotype(102+e,4:pop+3),length(j),1))=0;
    genotype(j,4:pop+3)=inter;
end

%translate thresholds and weights of the new population
for cons = 1:lengthofgen
    tr=genotype(cons,3);
    %first translate the threshold values
    while genotype(cons,2) ==1
        w=genotype(cons,4:y);
        T=find(~isnan(w) & w~=0);
        w(T)=TTT(tr,w(T));
        phen2(cons,1)=w;
        break
    end
end
%now translate the weight of the connection weights
while genotype(cons, 2) == 3
    w = genotype(cons, 4:y);
    T = find(~isnan(w) & w -= 0);
    w(T) = weights(1, w(1, T));
    phen2(cons, :) = w;
    break
end
end

for ret = 1:lengthOfRetinaSet
    %translate the address of the retinas in the genotype into the values
    %of the retinas taken
    for it1 = 1:lengthofgen
        while genotype(it1, 2) == 2 && genotype(it1, 3) == 1
            w = genotype(it1, 4:y);
            T = find(~isnan(w) & w ~= 0);
            w(T) = assemblyOfInitialRetinas(ret, w(1, T));
            phen2(it1, :) = w;
            break
        end
        ct1 = 1;
        ct2 = 1;
        ct3 = 1;
        %feed forward
        for it = 1:103
            tr = genotype(it, 2);
            %Evaluate the first layer of neurons
            if tr == 1 && genotype(it, 3) == 1
                Lay1Values(ct1, :) = phen2(it + 1, :) * phen2(it + 2, :) +... + phen2(it + 3, :) * phen2(it + 4, :) +... + phen2(it + 5, :) * phen2(it + 6, :);
                ct1;
                G(1, :) = Lay1Values(ct1, :);
                I(1, :) = phen2(it, :);
                T = find(~isnan(G(1, :)))
                G(G(1, :) < (1, :) < 0)
                G(G(1, :) = 0 && ~isnan(G(1, :))) = 1;
                Lay1Values(ct1, :) = G;
                ct1 = ct1 + 1;
            end
            %now we translate the adress for conections of layer 2
            if ct2 == 1 && it == 56
                La = find(genotype(:, 3) == 2);
                Ge = find(genotype(La, 2) == 2);
                waw = genotype(La(Ge), 4:y);
                for m = 1:pop
                    T = find(waw(:, m) ~= 0 && ~isnan(waw(:, m)))
                    waw(T, m) = Lay1Values(waw(T, m), m);
                    wot(:, m) = waw(:, m);
                end
                phen2(La(Ge), :) = wot;
            end
            %second layer
            if tr == 1 && genotype(it, 3) == 2
                Lay2Values(ct2, :) = phen2(it + 1, :) * phen2(it + 2, :) +... + phen2(it + 3, :) * phen2(it + 4, :) +... + phen2(it + 5, :) * phen2(it + 6, :);
                G(1, :) = Lay2Values(ct2, :);
                I(1, :) = phen2(it, :);
                G(G(1,:) < (1, :) < 0)
                G(G(1,:) = 0 && ~isnan(G(1, :))) = 1;
                Lay2Values(ct2, :) = G;
            end
    end
end
\texttt{ct2=ct2+1;}
\texttt{end}

\% now we translate the adresses for connections of layer 3
\texttt{if \ ct3==1 \&\& \ it==84
    \La3=find(genotype(:,3)==3);
    \Ge3=find(genotype(La3,2)==2);
    waw3=genotype(La3(Ge3),4:y);
    for \ m=1:pop
        T=find(waw3(:,m)==0 \& \& isnan(waw3(:,m)));
        waw3(T,m)=Lay2Values(waw3(T,m),m);
    \endfor
    phen2(La3(Ge3,:), :)=wot3;
\texttt{end}

\% Third layer
\texttt{if \ tr == 1 \&\& \ genotype(it,3) == 3
    Lay3Values(ct3,:) = phen2(it+1,:).*phen2(it+2,:)+
    phen2(it+3,:).*phen2(it+4,:)+
    phen2(it+5,:).*phen2(it+6,:);
    ct3;
    G(1,:) = Lay3Values(ct3,:);
    I(1,:) = phen2(it,:);
    G(G(1,:)<I(1,:))=0;
    G(G(1,:)==0 \& \& isnan(G(1,:)))=1;
    Lay3Values(ct3,:) = G;
    ct3=ct3+1;
\texttt{end}

\% now we translate the adresses for connections of layer 4
\texttt{if \ it==98
    \La4=find(genotype(:,3)==4);
    \Ge4=find(genotype(La4,2)==2);
    waw4=genotype(La4(Ge4),4:y);
    for \ m=1:pop
        T=find(waw4(:,m)==0 \& \& isnan(waw4(:,m)));
        waw4(T,m)=Lay3Values(waw4(T,m),m);
    \endfor
    phen2(La4(Ge4,:), :)=wot4;
\texttt{end}

\% Fourth layer
\texttt{if \ tr == 1 \&\& \ genotype(it,3) == 4
    Lay4Values(1,:) = phen2(it+1,:).*phen2(it+2,:)+
    phen2(it+3,:).*phen2(it+4,:);
    G(1,:) = Lay4Values(1,:);
    I(1,:) = phen2(it,:);
    G(G(1,:)<I(1,:))=0;
    G(G(1,:)==0)=1;
    Lay4Values(1,:) = G;
\texttt{end}
\texttt{end}

\% evaluate again against the expected values
\% the following operation changes the goal state
index=ceil(run/20);
\texttt{if \ mod(index, 2)==0
    expInd = 3;
\texttt{end}
\texttt{if \ mod(index, 2)==1
    expInd = 2;
\texttt{end}

expectedVal(ret,1)=IO(find(IO(:,1)==assemblyOfInitialRetinas(ret,11)),expInd);
\texttt{if \ expectedVal(ret,1)==assemblyOfInitialRetinas(ret,12),2;
    assemblyOfInitialRetinas(ret,12)=expectedVal(ret,1);
\texttt{end}
\texttt{Corrects(ret,Corr)=1;
\texttt{end}
\texttt{c2=sum(Corrects);}
fit(1,:) = linspace(1, pop, pop);
fit(2,:) = c2 .* .01;
[zz, i] = sort(fit(3,:));
b = fit(:,1);

LenPen = sum(~isnan(phen2(1:98,:)))/7;
LenPen(LenPen<9) = 0;
LenPen(LenPen>9) = (LenPen(LenPen>9) - 9)*.01;
fit(3,:) = fit(2,:) - LenPen;
run = run + 1;
resultados(pruebas,:) = fit(3,:);

% Plasmid replication

toc
% x = sum(fit(2,:))
% yy = sum(fit(3,:))
% x - yy

successes(prueba,1) = length(find(fit(2,:) == 1));

% prueba = prueba + 1
penalty = sum(abs(fit(3,find(fit(2,:) == 1))) - 1);
eff = 1./fit(3,:);
% end