NTNU
Norwegian University of
Science and Technology

Computation-in-Materio: evolving computation with light

Kristian Fladstad Normann

Master of Science in Computer Science
Submission date: February 2017
Supervisor: Stefano Nichele, IDI

Norwegian University of Science and Technology
Department of Computer Science

Summary

Evolution-in-Materio is a research field wherein evolutionary algorithms are used to seek out configurations that allows one to make use of a physical material as a computational device. The reasons for this are to seek out new viable materials for computation, as well as to explore the range of usefulness of evolutionary algorithms in computation research.

The project looks at the possibility of developing a frequency discriminator in an amorphous silicon solar panel exclusively by applying light onto the solar panel.

This thesis presents successful, albeit unstable 'blinking led' frequency discriminators that are evolved in-materio.

Sammendrag

I-materio-evolusjon er et forskningsfelt hvor evolusjonære algoritmer brukes til å lete etter materialkonfigurasjoner som kan brukes til å behandle det fysiske materialet som en beregningsenhet. Grunnene til dette er en søken etter nye fysiske brukbare beregningsmaterialer, så vel som det å oppdage nytteomfanget av evolusjonære algoritmer i datamaskinforskning.

Dette prosjektet ser på mulighetene for å utvikle en frekvens-diskriminator i 'amorphous' silisium solcellepanel ved å anvende lys på solcellepanelet.

Denne oppgaven presenterer vellykkede, dog ustabile 'blinkende lys' frekvens-diskriminatorer som er utviklet i-materio.

Acknowledgements

I would like to thank my supervisor Dr. Stefano Nichele for his guidance and support over the course of this thesis.

Contents

Summary i

Sammendrag iii

Acknowledgements v

Table of Contents viii

List of Tables ix

List of Figures xiii

1 Introduction 1
- 1.1 Assignment Text . . . 1
- 1.2 Thesis Overview . . . 2

2 Background 3
- 2.1 Novel avenues of computing . . . 3
- 2.2 Evolutionary computation . . . 3
  - 2.2.1 Evolutionary algorithms as tools . . . 5
- 2.3 Evolution-in-Materio . . . 5
  - 2.3.1 Eim experiment approaches . . . 5
  - 2.3.2 History . . . 7
  - 2.3.3 Nascence . . . 7
  - 2.3.4 Material substrates . . . 8
- 2.4 Frequency discriminator . . . 9

3 Methodology 11
- 3.1 Mecobo . . . 12
  - 3.1.1 Material under test . . . 13
- 3.2 Individual . . . 13
  - 3.2.1 Pins . . . 13

3.3 Evolutionary algorithm . . . 14
3.3.1 Fitness function . . . 15
3.3.2 mutation . . . 15
3.3.3 Generation selection . . . 16
3.4 Exploration of strategies . . . 17
3.4.1 Logic gates . . . 17
3.4.2 Crossing rate experiment . . . 23

4 Experiments **27**
4.1 Experiment setup . . . 27
4.1.1 Evolutionary algorithm overview . . . 27
4.1.2 Timing . . . 28
4.2 Experiment goals . . . 28
4.2.1 Experiment layout . . . 28
4.2.2 Representation of results . . . 29
4.3 An approximately smooth fitness function . . . 29
4.3.1 Results . . . 29
4.4 A discrete fitness function . . . 38
4.4.1 Results . . . 38
4.5 Frequency discriminator: reduced scope . . . 47
4.5.1 ”10” evolution target . . . 47
4.5.2 ”1010” evolution target . . . 48

5 Analysis **51**
5.1 Viability of approach . . . 51
5.1.1 The fitness functions . . . 51
5.1.2 Apparent Ceiling of computations . . . 52
5.1.3 Computation space . . . 52
5.2 Stability of results . . . 53
5.2.1 Dependencies between tests . . . 54
5.2.2 Environmental effects . . . 54
5.3 Using the Mecobo system . . . 54
5.4 Design constraints . . . 54

6 Conclusion **57**
6.1 Future work . . . 57

Bibliography **59**

Appendix **63**
6.2 Fitness function smoothness curve . . . 63
6.3 Code repository and experiment results . . . 64

List of Tables

2.1 electrical parameters of the amorphous silicon solar panel 9

3.1 fitness value constituents. Each sub-test of a full fitness function run yields an increment in one of the buckets. TP means the assessed logic value is ”1” and so is the expected value; the corresponding part of the fitness test is ”1”. TN means the assessed value is ”0” and expected ”0”. FP means assessed ”1”, but expected ”0” and FN means assessed ”0”, but expected ”1”. 11
3.2 Experimentally found parameters required for the EA 17
3.3 logic gates inputs and outputs 19
3.4 individual with high fitness score 19
3.5 3 good tests and 1 bad 20
3.6 2 great tests and 2 bad 20
3.7 Evolutionary algorithm parameters 21
3.8 Fitness evaluation scheme, used to calculate the constituent parts for the final single fitness function value for an individual 24
3.9 Experiment parameters 24
3.10 Fitness evaluation most common in the experiment 25

4.1 General experiment parameters 28
4.2 General experiment parameters 48

List of Figures

2.1 General behaviour of an evolutionary algorithm 4

2.2 Overview of the Evolution-in-Materio process. Artificial evolution is carried out in simulation in the computer domain with the fitness function being executed in the physical domain. 6

2.3 Visualization of varied silicon structures 9

2.4 Picture of an amorphous silicon solar panel of same type as the one used in the project. 10

3.1 Picture of the Mecobo system in action. On the right is the Mecobo hardware system(the green lights are leds for system heartbeat and are not connected to experiments). Pins on the platform are connected by wire to a led grid on the left on top of the cardboard box. The solar panel is inside, in an effort to isolate it. Its blue ground and recording wires can be seen at the bottom and are also connected to the Mecobo. 12

3.2 Overview of the experiment system 13

3.3 Graphic example representation of an abstracted individual in the evolutionary algorithm 14

3.4 ER is short for Expected Results and represents the fitness test as a list of tests, expecting step by step either logical ”1” or ”0”. AR is short for Assessed Results and represents the list of assessed logic values 16

3.5 average output value in logical ”1” domain 19

3.6 fraction of the ”1” domain covered by the average value 20

3.7 OR gate generated over 130 generations with a population of 40 individuals 21

3.8 AND gate generated over 130 generations with a population of 40 individuals 22

3.9 XOR gate generated over 130 generations with a population of 40 individuals 22

3.10 100 generation evolutionary run with population size 20, attempting to evolve a discriminator for frequency differences for blinking leds 25

4.1 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz. 30
4.2 Comparison between best individual score from evolution and same individual run through the fitness function again 10 times. 31
4.3 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz. Best individual from original evolution seeded in and cloned as starting population 32
4.4 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz. 33
4.5 Comparison between best individual score from evolution and same individual run through the fitness function again 10 times. 34
4.6 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz. Best individual from original evolution seeded in and cloned as starting population 35
4.7 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz. 36
4.8 Comparison between best individual score from evolution and same individual run through the fitness function again 10 times. 37
4.9 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz. Best individual from original evolution seeded in and cloned as starting population 38
4.10 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz. 39
4.11 Comparison between best individual score from evolution and same individual run through the fitness function again 10 times. 40
4.12 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz. Best individual from original evolution seeded in and cloned as starting population 41
4.13 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz. 42
4.14 Comparison between best individual score from evolution and same individual run through the fitness function again 10 times. 43
4.15 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz. Best individual from original evolution seeded in and cloned as starting population 44

4.16 168 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz. Be advised the y-axis differs from the other graphs in only showing range 6-8 on the fitness function. 45
4.17 Comparison between best individual score from evolution and same individual run through the fitness function again 10 times. 46
4.18 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz. Best individual from original evolution seeded in and cloned as starting population 47
4.19 Final generation number for successful evolutions that evolved a "10" blinking led frequency discriminator 49
4.20 Final generation number for successful evolutions that evolved a "1010" blinking led frequency discriminator 50

6.1 Distribution of theoretically obtainable fitness scores given a fitness test with binary representation "1010101010" and the fitness function F where $F$is $F = \frac{tp*tn}{(tp+fp+1)*(tn+fn+1)}$. The beginning left side of the plot shows all the instances where the numerator is zero and then the plot climbs all the way up to a maximum score where the correctness buckets look thusly: $tp : 5, tn : 5, fp : 0, fn : 0$ making the fraction $\frac{5*5}{(5+0+1)*(5+0+1)} \rightarrow \frac{25}{36} \approx 0.7$ as can be seen on the right side of the curve. The curve is not smooth, but possibly smooth enough to allow for somewhat fluid fitness score movement. 63

Chapter 1

Introduction

This project is an exploration of the viability of using evolutionary algorithms to develop computation in an unorthodox physial material. The traditional engineering approach would entail studying the materials to be used, learn how to carefully manipulate them and then, use a top-down design approach to develop the system. An evolutionary design process on the other hand, takes inspiration from natural evolution and endeavours to find functional designs with a ”blind-yet-guided” search for properties to exploit in the system it is working on.

This thesis explores the possibility of using Evolution-in-Materio to generate computation in an amorphous silicon solar panel by using controlled exposure of light. This is in an effort to learn if it is actually possible to develop a genuine system through merely applying light in a controlled manner.

1.1 Assignment Text

Keywords: Evolution in Materio, Cellular Automata, Evolutionary Computation, Complexity.

In this research project we try to exploit computational properties of unconventional materials (materials usually not considered as a computational substrate). Such materials may offer computation at extreme low cost and may also enable us to do computation that is hard (or impossible) on a von Neumann stored program machine. Previously investigated materials are carbon nanotubes/polymer composites, liquid crystals, gold nanoparticles. Currently we explore possible computational properties of solar panels (thin film amorphous silicon). In 2010 a first version of a platform was made. This system consists of a PCB, including an Atmel microcontroller and a Xilinx FPGA that acts as an interface between a PC and a material. The goal of the project is: Investigate the possibility of doing computation with light (LEDS / lasers) in amorphous silicon solar panels.

1.2 Thesis Overview

The experiments presented in this thesis are all variations of blinking led frequency discriminator. The differences are between fitness function differences and differing [high frequency, low frequency] pairs. The working hypothesis is that Evolution-in-Materio is feasible for the material-under-test, going by the experimental process laid out in Chapter 3 and Chapter 4.

The thesis is laid out ass follows: This chapter presents the central point of the project and the assignment text. Then follows the background chapter. It lays out the themes of evolutionary computation, Evolution-in-Materio state of the art and modern approaches to material substrates for research in Evolution-in-Materio. Then follows the Methodology chapter, detailing the developed evolutionary algorithm, how individuals are represented and lastly some of the experimental strategies that led up to the final experiments. After that comes the Experiments chapter, showing all the experimental results. Then follows the Analysis chapter, with a critical attempt at understanding the results. Finally a conclusion is drawn with some suggestions for future work.

Chapter 2

Background

2.1 Novel avenues of computing

The process of computing can occur in all sorts of forms, but the most common one is in the form of transistor based von Neumann[17] silicon microchips. That field has gone through decades of optimizations in speed of computation, throughput, cache-miss handling and energy efficiency, but the core model[9] is quite the same.

There are generally speaking, two approaches to unconventional computing. There is the effort of implementing well known modes of computation in new material substrates, which might allow for improved physical computers, yet still be compatible with the wealth of existing software. An example of such an event is the invention and subsequent transition to transistor-based solid state logic from vacuum-tube based computers[20].
Then there is the avenue of alternative models of computation. Developing systems like cellular automata[10], [28] and [29]. Extremely parallel systems that might possibly be appropriate for developing interesting new software, but unlikely to be very compatibe with pre-existing Von Neumann based systems.

2.2 Evolutionary computation

Evolutionary computation is the general term for the type of computation where an optimizing/searching/adapting process takes place that is guided by some meta-heuristic function.

One such algorithm is the genetic algorithm. Conceptually reminiscent of Darwinian evolution by natural selection, it is an algorithm type where the underlying problem is attempted solved by creating a population of different abstracted individuals and simulating a cycle of life and death for the population; generating a new generation on the basis of a fitness function. But where the fitness function in natural selection could be seen as a function of successful survival and mating, the fitness function in the genetic algorithm will instead relate to the underlying problem attempted to solve. With the differing

individuals representing solution attempt for the problem, receiving a fitness score that seeks to represents how good a solution the individual is. A general example evolutionary algorithm can be seen in figure 2.1.

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
  OD
END
```

Figure 2.1: General behaviour of an evolutionary algorithm

Initialize

The initialization of the population will typically mean either beginning at some start point chosen by the caller of the algorithm, or simply randomizing the population.

Evaluate

Now each individual in the population is evaluated by the fitness function. If the goal is to derive a circuit that can perform the logical operation XOR on two 1-bit logical inputs and each individual is a circuit, then the evaluation function might run all the possible inputs on an individual and assign scores based on correctness/errors.

Termination condition

The algorithm will ideally run until at least some individual has a high enough fitness score that corresponds to a successful design from the view of the caller of the algorithm. However, there is inherently no guarantee that any individual will achieve a high enough fitness score and so one might add some maximum number of allowed iterations before terminating.

Select

In each iteration, a number of current individuals are selected to function as parents for the next generation or merely be promoted to the next generation. This could mean picking only one individual, the one with the best fitness score, some n number of top individuals or possibly probabilistic ones.

Recombine

Recombination or crossover is the part of the algorithm where offspring individuals are created from parents. Its use may be stochastic by e.g. pairwise iterating over the parents and have the offspring probabilistically inherit from one or the other. Alternatively Recombine might not be used at all.

Mutate

The offspring is then typically subjected stochastically to one or more changes.

2.2.1 Evolutionary algorithms as tools

Evolution as an engineering and research tool is novel because it provides an orthogonal insight into development of new designs. That is, it differs from normal top-down research/engineering thinking. Where normal, human design strategies tend to try to strategize and lay overarching plans for how to reach a goal, an evolutionary approach is one that is blind, yet guided by a fitness function that all individuals(be they actual individuals or abstractions over the actual problem) are selected by.

Referring back to section 2.1, evolutionary algorithms present something of a third approach to computing research. By tapping into the workings of an evolutionary algorithm while seeking to design a computational system, then the scope of possible solutions might be increased. This could well be the case whether a brand new type of system is attempted developed, as in the case of this project, but can just as well be the case in the development of traditional systems. An example of the latter is the development of an X-band antenna[16] by NASA for use on NASA's Space Technology 5 (ST5) spacecraft. An evolutionary algorithm was set to search for an antenna design and managed to come up with a more performant design than the ones designed by humans.

2.3 Evolution-in-Materio

Evolution-in-Materio (EiM)[22] is a computer research area that consists of research into interesting and unorthodox physical on the one hand, and substrates to act as computing materials on the other, using evolutionary algorithms as a means of implementing the computations in the materials, see figure 2.2.

2.3.1 Eim experiment approaches

The paper "Evolution-in-materio: Evolving computation in materials"[22] describes how there are different types of approaches to an evolutionary development of a system.

Entirely in software

Entirely digital systems. The EA works on some software problem, e.g. difficult types of search and optimization problems.

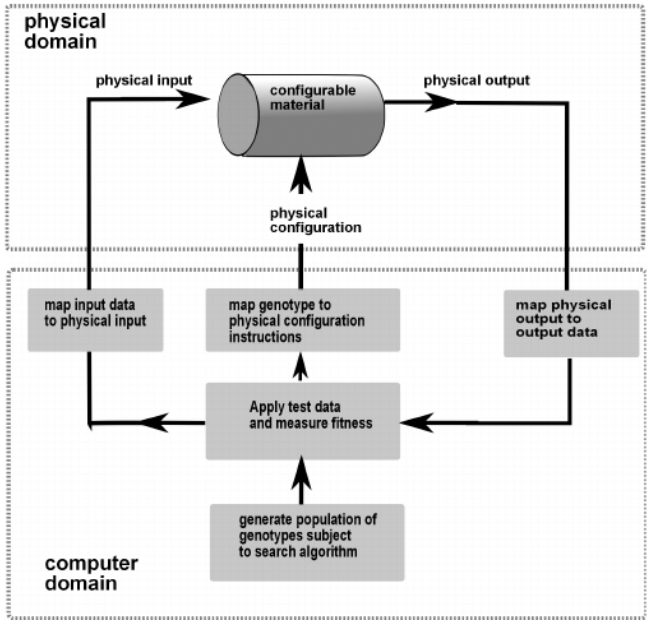

Figure 2.2: Overview of the Evolution-in-Materio process. Artificial evolution is carried out in simulation in the computer domain with the fitness function being executed in the physical domain.

Evolving the blueprint

The software evolution of a physical product. This can refer to a blueprint, as in the case with the NASA antenna [16]. What becomes notable here is the necessity of having a clear understanding of the environment the finished product will be in. Failure to account for stress, heat, pressure etc. that real world physical objects are subjected to, can render a design useless. This means that there is some amount of pressure to account for such in the selection process.

Evolving the physical object in intervals

Evolution entirely in physical material would be getting close to biological evolution by natural selection. Or as with Pask, a case of evolution guided by an overmind. However, an intermediate method of approach(called embodiment level 2.5 in [22]) between the ”Evolving the blueprint” approach and ”entirely in the physical objects” is one where the current state of the design is run as a physical object on e.g. a generation by generation basis, or individual by individual basis. The major two-fold reason for this(and likely the reason why much of the research in the field of EiM is done in this way) is that it allows for powerful software abstractions that maintain a model of the developing system as it develops. This way, the system can be iteratively tested, giving valuable feedback to the model, possibly allowing it to implicitly take in to account the various physical complexities and stresses that the physical evolved object is subjected to(likely much better than the

”Evolving the blueprint” approach). This gives a real world viability to the system much more immediately.

This project makes use of the latter approach. Each individual in the population is tested physically during the run of the fitness function, in an effort to incorporate the presumed varied capacitance and resistance in the amorphous silicon.
This work will make use of mecobo[19]. It is a hardware experiment platform (see figures 3.1 and figure 3.2, chapter 3.1) that makes it easier to connect to some physical material so that the evolutionary algorithm may be run on it to create computation. Specifically, it allows one to connect electrodes to the material that act as input, output and configuration wires.

2.3.2 History

Evolution-in-Materio can roughly be said to have started with the work of Gordon Pask in 1958 when he was able to develop a set of dendritic wires to behave as frequency discriminators[24]. Pask’s endeavor was to develop a device that could distinguish between various types of sound or magnetic fields. The manner by which he achieved this, was through training his dendritic wires by what today is most recognizable as something resembling a hill-climbing algorithm. Changing some resistor values at a time and evaluating the system on a trial and error approach. A development method that was quite new at the time. In the end the device could be discriminate between tones of 50Hz and 100Hz.

The field does have some more modern beginnings as well though. The term itself was coined in 2002[21], but the most modern beginning was likely in 1996 with Thompson. Thompson was able to successfully use artificial evolution on a field programmable gate array (FPGA) to make a square waves discriminator that could discriminate between 1kHz and 10kHz waves[27]. An interesting aspect of that evolutionary run is that Thompson discovered that the evolved design relied on more than just the logical domain. It depended on physical properties of the FPGA. While in isolation, such an unexpected development might be a detriment for a design, it does showcase part of the reason for conducting evolution-in-materio experiments in the first place. Recognizing that the evolution might surprise.

2.3.3 Nascence

NAnoSCale Engineering for Novel Computation using Evolution(NASCENCE) was an Evolution-in-Materio research project with the stated goal: ”The aim of this project is to model, understand and exploit the behaviour of evolving nanosystems (e.g. networks of nanoparticles, carbon nanotubes or films of graphene) with the long term goal to build information processing devices exploiting these architectures without reproducing individual components”[4].

Mecobo open source experimental platform

The Mecobo platform is a hardware and software platform for experiments in Evolution-in-Materio, developed by Lykkebø et al[19]. It is designed so that it can be used with all

sorts of physical substrates, see figure3.1, chapter 3.1. The system allows for applying controlled, finely specified electrical inputs onto whatever material-under-test.

2.3.4 Material substrates

The research into the physical materials that underlie computation is interesting first and foremost because there is no comprehensive knowledge on the computability of matter in general. This could be very important, as understanding what physical properties can be exploited in computation could result in great advantages over current computers, not to mention potential improvements in current computers themselves from a greater understanding of silicon based computers and their properties. By finding other mediums of performing computation we might also be able to find ways of building computers that do not have to rely on highly exhaustible resources like rare metals.

The NASCENCE project has a website listing the attributes of the ”ideal material”[7]

- Has a complex, configurable, semi-conducting structure
- Responds instantly and consistently to a wide range of voltages
- Relaxes into an un-configured or random state when voltage is removed
- Robust to changes in the environment (light, heat, electromagnetic signals etc)
- Consistent between devices

Some examples of such mediums are liquid crystal, water[11] and carbon nanotubes-based computers[23].

Liquid crystal computers

Liquid crystal has been shown repeatedly to be a very interesting substrate for computation. They have been used to evolve logic gates[13], a robot controller[15] and a tone discriminator[14].
The special thing about liquid crystal as a computation medium is that it can exist in a mesomorphic state(having properties of liquids and of solid crystal). The material is somewhat stable, but can be altered by induced current. In terms of ”The ideal material”, this makes for a slight challenge, as the material may relax into a different state when the input current stops being applied. This may warrant developing some manner of a ”reset” functionality should complex development be undertaken. On the other hand this behavior may be seen as an advantage if the state transitions can be found exploitable.

Carbon nanotube robot controller

In [23], Mohid and Miller were able to evolve a sophisticated robot controller for a swarm bot that was able to move quite well in actual physical settings with obstacles.

Max power	0.5W
Working voltage	1.5V
Working current	250mA

Table 2.1: electrical parameters of the amorphous silicon solar panel

Specifically, single-walled carbon nanotubes are interesting because they may exhibit varied electrical conductivity and semiconducting capabilities. This makes them very interesting as a computation medium research subject. For example by arranging many of them in a random network and experimenting on them to see if they can act as some specific type of computing device[26]

Solar panels

This project uses solar panels(purchased on Alibaba[5]) as the substrate for computation. An interesting aspect of this is that they can be viewed as already having a skeletal structure in place when it comes to computation. That is, they can receive photons as input and subsequently output a current. Amorphous silicon forms a continuous random network, see figure 2.3. On account of this randomness, some of the atoms have a dangling bond, which yields anomalous electrical behaviour. What remains then is to investigate the capabilities of amorphous silicon solar panels and whether it is possible to exploit the potential varied capacitive and resistive properties in the material for computational use.

The solar panel is subject to the constraints in table 2.1 and can be seen in figure 2.4.

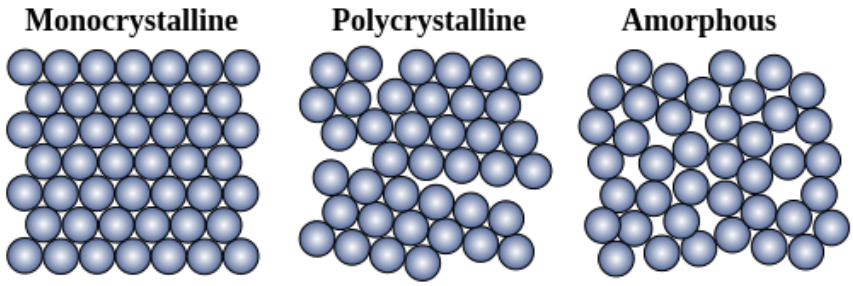

Figure 2.3: Visualization of varied silicon structures

2.4 Frequency discriminator

In its simplest form, a frequency discriminator is a device that, when presented with one of two signals, returns a different response for each signal. Specifically in this project, the amorphous silicon solar panel is evolved to output different signals based on a given pair of frequencies of blinking leds.

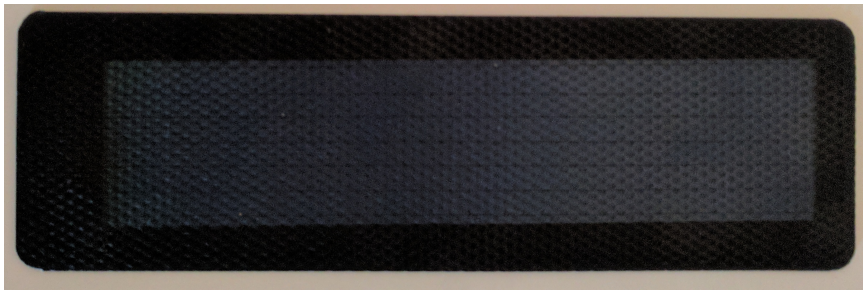

Figure 2.4: Picture of an amorphous silicon solar panel of same type as the one used in the project.

Chapter 3

Methodology

The approach when conducting the experiments has been to construct a base evolutionary algorithm and alter parameters for differing experiments.

The experiments all followed the same procedure of creating a starting population of individuals that each represent an electrical configuration. Then they were run through the evolutionary algorithm, that would either end if any individual reached maximum fitness score or a given maximum number of generations. At this point the best performing individual would be logged.

The fitness function is what is specifically run on the Mecobo. The abstract pin values of an individual are issued as particular electrical signals on the corresponding physical pins. The configuration pins run with their values and the input pin is configured with the fitness test. Over the course of such a run the recording pin records the voltage output from the solar panel for analysis.

That analysis consists of first transforming the buffer of recorded voltage points from the solar panel into something more analyzable, and the second part consists of taking the ”treated” buffer and counting up four counters used to determine the fitness score of an individual, see figure 3.1. In a given run of the fitness function, these counters are incremented in accordance with a comparison between the assessed logic values of the treated buffer and expected logic result. The counters are then used to calculate the fitness score of the relating individual.

True Positive(TP)
True Negative(TN)
False Positive(FP)
False Negative(FN)

Table 3.1: fitness value constituents. Each sub-test of a full fitness function run yields an increment in one of the buckets. TP means the assessed logic value is ”1” and so is the expected value; the corresponding part of the fitness test is ”1”. TN means the assessed value is ”0” and expected ”0”. FP means assessed ”1”, but expected ”0” and FN means assessed ”0”, but expected ”1”.

3.1 Mecobo

The Mecobo platform[19] is a hardware/software system for conducting EiM experiments. The hardware system allows for issuing and reading arbitrary electrical patterns on exposed pins. By connecting pins to the leds and the solar panel by wire, see figure 3.1, experiments can be conducted where the leds emit controlled light patterns onto the solar panel, and the solar panel voltage can be recorded for analysis and be used in the experiments.

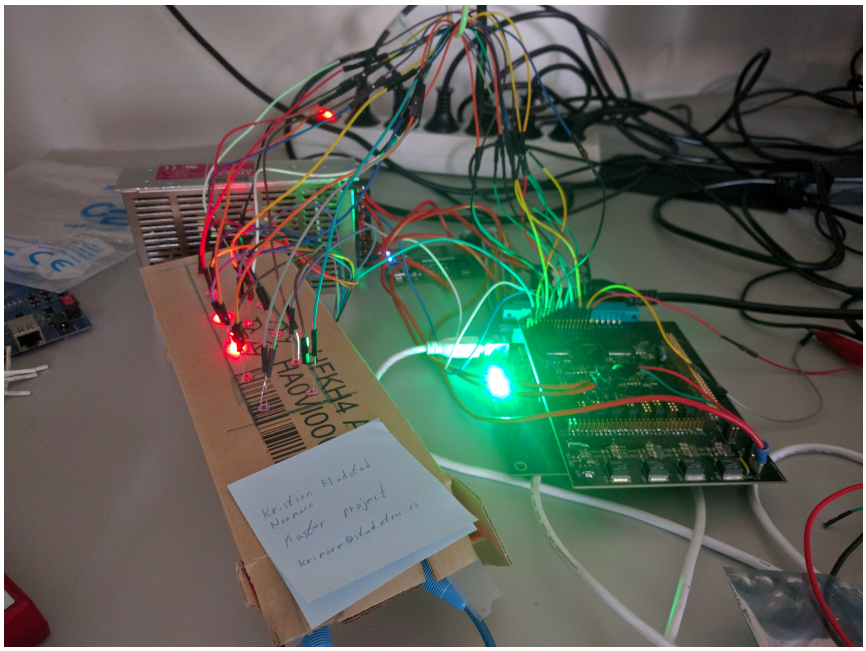

Figure 3.1: Picture of the Mecobo system in action. On the right is the Mecobo hardware system(the green lights are leds for system heartbeat and are not connected to experiments). Pins on the platform are connected by wire to a led grid on the left on top of the cardboard box. The solar panel is inside, in an effort to isolate it. Its blue ground and recording wires can be seen at the bottom and are also connected to the Mecobo.

The total system, see figure 3.2, is a client/server pair that uses thrift[1] for communication, allowing the user to write the evolution experiment as a python[2] script, and by running it, have thrift send it to the server. Server and client both used the Ubuntu[3] operating system. Server side, the Mecobo[19] part of the script is turned into an electrical specification for the pins to be used. Some are used for issuing electrical signals to induce the material-under-test and some can read voltage values from the material at specific points. The experiment platform was Mecobo v4.1[6].

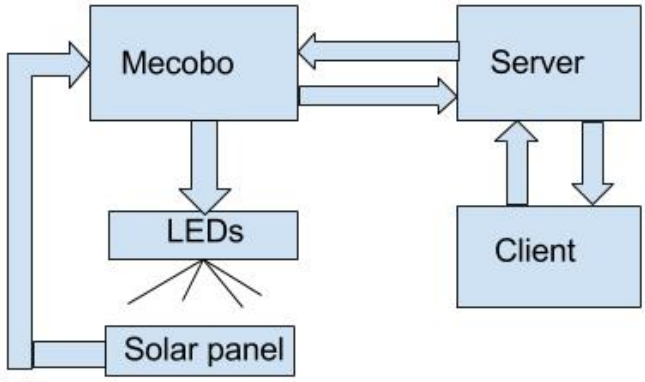

Figure 3.2: Overview of the experiment system

3.1.1 Material under test

The material under test is an amorphous silicon solar panel, see chapter 2.3.4: Material substrates. The field of Evolution-in-Materio[22] entails less focus, if not none on studying the capabilities of the solar panel. Instead, the project works with the hypothesis that is may be possible to induce controlled varied resistance and capacitance in the amorphous silicon structure. The evolutionary algorithm then uses the fitness function to search for such a configuration that is meant to achieve the overarching computation goal.

3.2 Individual

The devised evolutionary algorithms in this project work on a population of individuals. An individual here is an abstraction over a set of electrical pins. Specifically, it can be seen as consisting of three parts:

- a collection of electrical pins on the Mecobo system
- a time range for each of those pins
- and a set of voltage patterns for each of those pins over the course of that time

The constructor function for an individual takes as argument, a list of pin numbers to use, the number of configuration pins to create and the number for input pins to be created. This function randomly assigns the different types of pins accordingly.

3.2.1 Pins

The pins belonging to the abstract individual are input and configuration. In addition there are pins for electrical grounds and a recording pin connecting the Mecobo and the solar panel; allowing for reading voltage points during runs of the fitness function. The ground and recording pins are constant set for all experiments and are thus not subjected to the evolutionary algorithm.

Input pin

The input pin, or pins, depending on the required number of input pins for the given experiment, is a pin that is run with the fitness function(FF). That is, the FF might be testing the solar panel on its ability to discriminate between two differing frequencies of a blinking led, where the current experiment works with a set pair of frequencies to test.

The FF may be represented as something like ”1010”. This means ”high frequent signal, then low frequent signal, twice” in succession. Each with equal time slice. Looking at figure 3.3(ignoring input pin 2) considering input pin 1, each quarter slice of the time between t_start and t_end will be assigned the corresponding frequency signal.

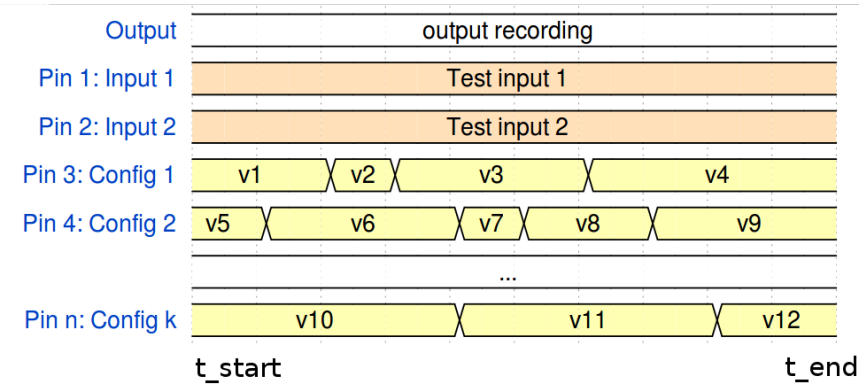

Figure 3.3: Graphic example representation of an abstracted individual in the evolutionary algorithm

Configuration pin

The configuration pins act as the genotype of the evolutionary algorithm. Upon initialisation, after being assigned to some physical pins, each configuration pin is assigned a list of randomly constructed ”voltage commands”. That is, a time-wise sequential list of varied electrical commands throughout the given fitness function run; a visual example of which can be seen in figure 3.3, ”Config 1,2,k”. It is primarily through manipulation of the configuration pins, that the evolutionary algorithm works towards improved fitness scores.

3.3 Evolutionary algorithm

The evolutionary algorithm developed in this project is a simple genetic algorithm in which a starting population of individuals is constructed either by randomly generating each individual, or by seeding in a pre-existing individual and cloning it. Then that population enters an evolution loop for the duration of the experiment. The main experiments typically had a ceiling of 150 generations or if the maximum fitness score was achieved. At either point the loop would break and final logging would be conducted before finishing.

3.3.1 Fitness function

The fitness function is where the hardware meets the individual-under-test. It entails running the given input pin with the fitness test. At the same time, the configuration pins are run with their current state of frequencies and the recording pin is run at its given recording frequency. Then, during analysis, the buffer of recorded values is split up in slices corresponding to the parts of the fitness test and each part gets analyzed and evaluated accordingly.

Splitting the output buffer

When the physical part of the experiment is over, the output buffer contains sampled voltage point data from the solar panel. This is then passed to an analysis function that splits up the buffer in slices corresponding to the fitness test. If the experiment is about evolving a frequency discriminator and the fitness test was ”10” where ”1” denotes a high frequent signal and ”0” denotes a low frequent signal than the output buffer is accordingly split in two. The buffer slices that correspond to each fitness test part are then sent along for analysis.

Frequency domination assessment

Each solar panel voltage buffer slice that corresponds to a specific frequency sub-test is then used to calculate the fourier transform of that slice via the Fast Fourier Transform[12]. That fourier transform is inspected to find the Dominating frequency, which in turn is compared to the average value of the high frequency, low frequency pair used for that particular experiment. The transformed buffer slice is then assigned an assessed logic value based on the function in equation 3.1.

$$\text{Assessed logic value} = \begin{cases} 0 & \text{if Dominating frequency } < \frac{\text{logical high}+\text{logical low}}{2} \\ 1 & \text{else} \end{cases} \tag{3.1}$$

Full individual assessment

In order to be able to calculate the final fitness value for the individual, one must iterate through all the slices of the fft treated solar panel voltage buffer data, calculate the assessed logic value, see equation 3.1, thereby creating a list of assessed logic values.

Then what remains is to count up the four fitness constituent ”buckets”, see table 3.1. This is done by step-wise iterating over the list of assessed logic values in tandem with the actual fitness test and incrementing the appropriate counter depending on match, see the algorithm in figure 3.4. When this is complete the counters are passed to the final fitness score calculator.

3.3.2 mutation

Mutation of an individual occurs in the following manner. First, a random number of configuration pins are selected. Then, each of the selected pins are reassigned a randomized

Algorithm 1 ER is short for Expected Results and represents the fitness test as a list of tests, expecting step by step either logical "1" or "0". AR is short for Actual Results and represents the list of assessed logic values

```
function FitnessConstituentsIncrementer(ER, AR)
    tp = 0, tn = 0, fp = 0, fn = 0
    for each index i in ER.length do
        if ER[i] == 1 then
            if AR[i] == 1 then
                tp+ = 1
            else
                fn+ = 1
            end if
        else
            if AR[i] == 1 then
                fp+ = 1
            else
                tn+ = 1
            end if
        end if
    end for
    return (tn, tp, fp, fn)
end function
```

Figure 3.4: ER is short for Expected Results and represents the fitness test as a list of tests, expecting step by step either logical "1" or "0". AR is short for Assessed Results and represents the list of assessed logic values

frequency, subject to the experiment frequency constraints.

Frequency constraints

The main experiments make use of digital signals with specified voltage frequencies. Specifically, the configuration pins and the input pin. The possible assignable frequency range for a pin was set to the range 1Hz to 2500 Hz. Specifically, on the Mecobo system this means that the given pins will play square waves with the given frequency as the cycle time[6].

3.3.3 Generation selection

The generation-to-generation selection is a simple scheme where the best individual in a generation is promoted to the next generation and cloned such that the number of individuals in the population is constant. In that new generation, every individual, except one, is mutated.

Parameter	value
Min. amplitude value	40
Min. $Volt_{mecobo}$ value	-3.43V
Max. amplitude value	190
Max. $Volt_{mecobo}$ value	2.45
logical 1 max value	0.267
logical 0 min value	-0.015
logical threshold, between high and low, average	0.126

Table 3.2: Experimentally found parameters required for the EA

3.4 Exploration of strategies

Over the course of the project, several approaches were taken in an attempt to evolve computation in the solar panel. What follows are briefly, two of the attempted strategies.

3.4.1 Logic gates

The first approach was to try to evolve the logic gates OR, AND and XOR in the solar panel where the signals sent to Mecobo were of type Constant(which means analog signal) and assign the signals amplitudes.

Description

Through some experimenting with different frequencies and amplitudes(Mecobo software parameters for giving a pin voltage and time commands) it became clear that an amplitude range of $[40, 190]$ was what gave the most useful and discernible results. Specifically, by reading the "mecobo_user_manual.txt" on the mecobo development site[6] it was found that the mecobo amplitude range was $[0, 255]$, which corresponds to a voltage range of $[-5V, 5V]$. This means that the amplitude range of $[40, 190]$ corresponds to the voltage range $[-3.43V, 2.45V]$.

At this point, in order to meaningfully evolve the logic gates it became necessary to find parameters for a maximum and minimum value such that logical "1" and "0" could be assigned on a spectrum. This was achieved by running one test where the maximum number of possibly assigned pins(and thereby LEDs) were all turned on at maximum amplitude and taking the average of the recorded output. Then one where all were turned on at the minimum amplitude and taking the average(see table 3.2).

With these tests done it was possible to perform the EA using the established parameters (see table 3.2). For the sake of simplification, the output range was divided into two equal halves for the logical "0" and "1" decoding(see logical threshold, table 3.2).

With that done it became possible to represent the inputs "0" and "1" in the system. Some assigned pin(and subsequently led) would for the input "1" then be assigned a constant voltage with amplitude 190(2.45V) over the course of the entire test, and accordingly,

the ”0” input would denote amplitude 40(-3.43V).

The representation of an individual in this experiment was the same as in section3.2, with two exceptions.

1. The configuration pins could contain differing amplitude values over the course of differing time-slices, subject to the global time of the fitness test, as the example in figure 3.3 shows.
2. The mutation scheme, which worked chronologically as follows:
   (a) The individual has a 90% chance of randomly switching the place of two pins in use(e.g. from figure 3.3, switching pin 1: input 1, and pin4: config 2)
   (b) Then the individual will pick randomly a pin number it uses and see if the pin is a config pin. If it is, there is a 90% chance of calling the [time, volt] sequence mutation function of that pin. If this happens, one of the [time,volt] sequence slices is picked at random(e.g. time,volt slice ”v6” in pin 4 figure 3.3). Then there is first a 70% chance of getting a new random volt assigned. Then there is a 70% chance of the time, volt slice to expand or contract by up to one thousandth of the total time range.

Fitness function

The tests in these experiments were centered around evolving OR, AND and XOR gates. As such, the test of an individual was a set of four tests where the logic gate was tested against the expected [input,output] pairs, see table 3.3.
The fitness score for an individual is calculated by finding the average value of the output recording points assigned by summing the following parts:

1. does the average value of the output recordings place it in the correct half of the spectrum? If so, the score is assigned 2 points, see figure 3.5. If it is not, then the score is assigned -2 points.
2. Then a score is assigned based on how well, or bad the score is in terms of how close the average value is to the max or min values, see figure 3.6. If the average was correct from the previous part, then the assigned value now is a fractional value in the area $[1.0, 2.0]$ and correspondingly for a wrong average, $[-2.0, -1.0]$.
3. Since there are four input,output tests for the logic gates, see table 3.3, each of which is scored on the two previous points, the final score, all of their part sums summed together, may also receive a final bonus if all the tests succeeded. This bonus is 1.5 points.

This means that the theoretical best individual may achieve a score of 4*2 + 4*2 + 1.5 = 17.5 fitness score and the worst theoretical score is 4*-2 + 4*-2 + 0 = -16

What follows are three example scores for individuals under this fitness scheme:
high value individual: An individual with 4 correct and very high fractional score on the tests can expect something in the vicinity of the score in table 3.4:

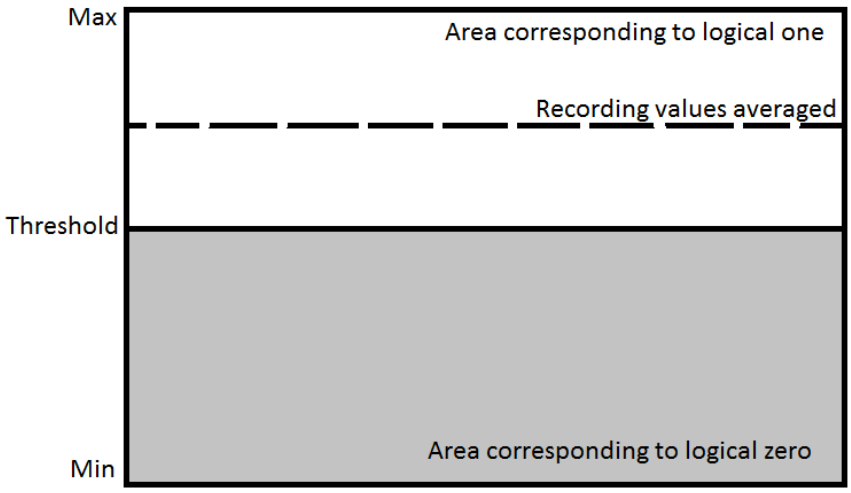

Figure 3.5: average output value in logical ”1” domain

Table 3.3: logic gates inputs and outputs

Input 1	Input 2	OR	AND	XOR
0	0	0	0	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	0

Three good scores and one bad

An individual that scores decently on three of the tests but badly incorrect on one might score similar to table 3.5

Two excellent scores and two barely incorrect

An individual that scores very well on two of the tests and just barely incorrect on two might score similar to table 3.6

Table 3.4: individual with high fitness score

test input	00	01	10	11	part sum
correctness score	2	2	2	2	8
fractional score	1.8	1.83	1.84	1.92	7.39
bonus	1.5				1.5
Total Fitness score					16.89

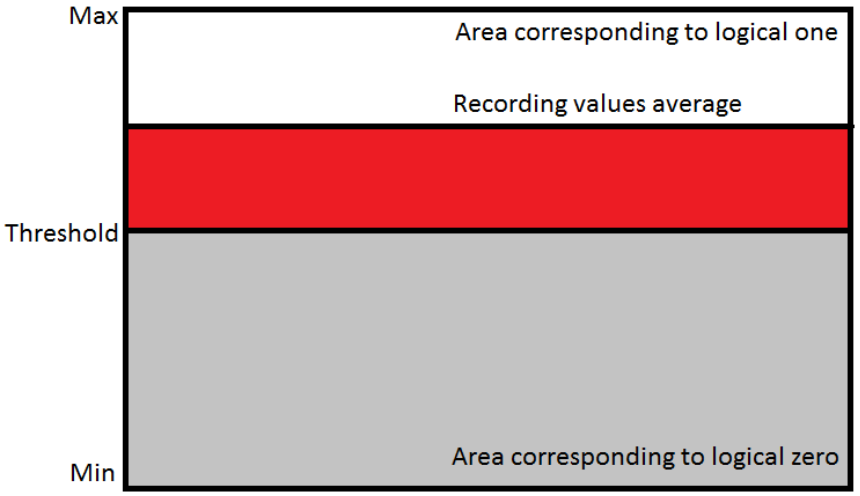

Figure 3.6: fraction of the ”1” domain covered by the average value

Table 3.5: 3 good tests and 1 bad

test input	00	01	10	11	part sum
correctness score	2	2	2	-2	4
fractional score	1.8	1.83	1.84	-1.8	3.67
bonus	0				0
Total Fitness score					7.67

Experiments

Lastly, the EAs are called using the parameters in table 3.7.

OR gate:

The OR gate experiment (see figure 3.7) found a computationally stable solution quickly, and an absolute best individual on the 77th generation with fitness score 17.09.

Table 3.6: 2 great tests and 2 bad

test input	00	01	10	11	part sum
correctness score	2	2	-2	-2	0
fractional score	1.95	1.94	-1.05	-1.02	1.82
bonus	0				0
Total Fitness score					1.82

Table 3.7: Evolutionary algorithm parameters

Parameter	configuration
population size	40
Number of maximum generations	130
Number of pins assigned to an individual	6(2 input, 4 config)

Figure 3.7: OR gate generated over 130 generations with a population of 40 individuals

AND gate:

The AND gate experiment (see figure 3.8) found a computationally stable, though incomplete solution quickly. From generation 27 to the absolute best individual on generation 128 at a fitness score of 7.96

XOR gate:

The XOR gate experiment (see figure 3.9) found a computationally stable, though incomplete solution quickly by generation 21, and an absolute best individual on the 60th generation with fitness score 7.70.

The initial randomization does much of the job. This has likely to do with a big spread from the randomization initialization function and that the population size of 40 is big enough to get a wide variety of fitness values. After that though there is little improvement on the best individual. Though as all three tests show, a lot of changes still occur in the populations and short steps of changes can lead the worst individual measurement to plunge.

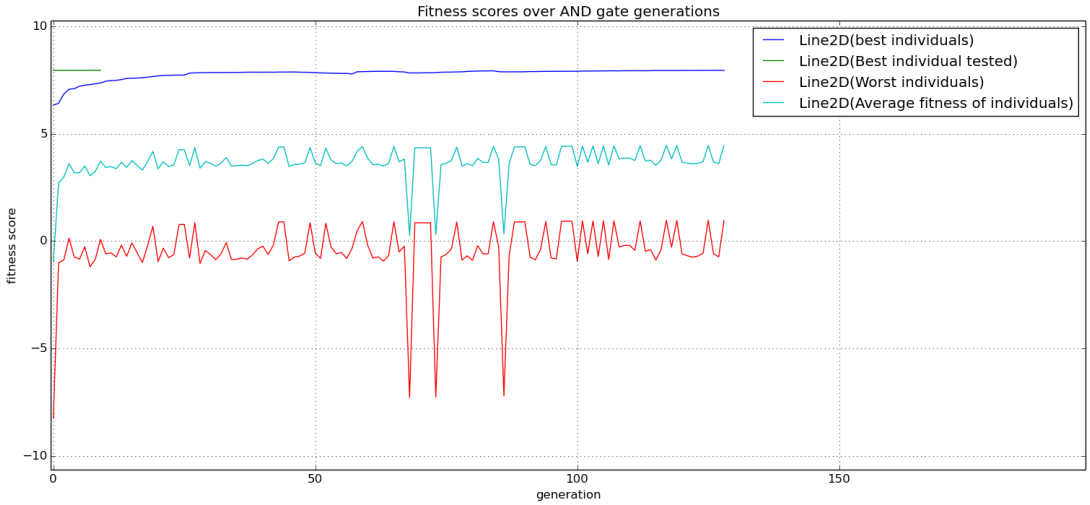

Figure 3.8: AND gate generated over 130 generations with a population of 40 individuals

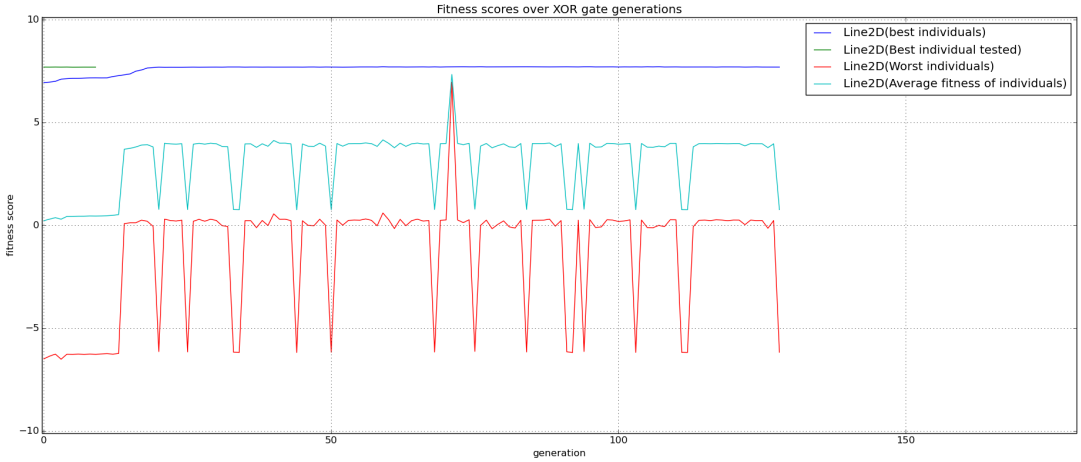

Figure 3.9: XOR gate generated over 130 generations with a population of 40 individuals

3.4.2 Crossing rate experiment

The second approach was to try to make a frequency discriminator by using a crossing rate scheme. Attempting to assess differing frequencies by finding an appropriate crossing point and counting crossings over it.

Description

The crossing rate experiment works as follows: By slicing the output buffer(of recorded voltage measurements from the solar panel) corresponding to each sub test of the input test. The next part is to iterate through each slice and calculate the average voltage(AVG)in the slice and use that as a constant for the crossing rate scheme. After that, each recording point in the slice is evaluated step-by-step and it is observed whether the transition from one point to the next crosses the AVG line. Each crossing increments a crossing counter, see equation 3.2.

A simple example would be if the average voltage value is 0.39 volt. Point 1 from the recording buffer slice has value 0.6 volt and point 2 has value 0.31 volt. That means that there is a crossing. However, if point 2 has value 0.45, there is not a crossing and the crossing counter is not incremented

$$\text{Crossing counter (CC)} = \text{CC + 1} \quad \text{if crossing} \tag{3.2}$$

Once the slice of the output buffer has been evaluated, a crossing rate fraction is calculated by taking the crossing rate and dividing it by the n number of crossings in the buffer slice, see 3.3.

$$\text{Crossing Rate (CR)} = \frac{CC}{n} \tag{3.3}$$

Individual configuration

Each individual is structured with a simple time-slice system. Each of its pins, except the designated input pin, is allotted one frequency over the course of the entire time span of the fitness test.

Fitness function

The fitness function is in effect, a collection of calculations. Each calculated for each individual in a generation.

Then, another threshold is created to denote the point where a crossing rate is considered logical 'high' or 'low'. This threshold was set to 10% for the CR, or 0.1.

With each slice of the output samples going through this calculation, the individual under test gets assigned four counters for seeing how it performs in the final fitness function calculation. That is, whether the crossing rate of each slice is evaluated to logical 0 or 1, and whether that was the expected result or not(the expectation being the corresponding value in the input vector, see table 3.9), see table 3.8.

expected result	actual result	counter
1	1	true positive += 1
1	0	false negative +=1
0	1	false positive +=1
0	0	true negative += 1

Table 3.8: Fitness evaluation scheme, used to calculate the constituent parts for the final single fitness function value for an individual

Table 3.9: Experiment parameters

Parameter	value
Input vector	1010101010
Input: logical low frequency	20
Input: logical high frequency	500

Finally those four counters are used in formula 3.4 to calculate the fitness function of the individual.

$$\text{Fitness function value} = \begin{cases} \frac{tp+tn}{fp+fn} & \text{if } fp + fn \text{ is not 0} \\ 10 & \text{if } fp + fn \text{ is 0} \end{cases} \tag{3.4}$$

Generation selection scheme

The generation selection scheme is the same as the one in section 3.3.3.

Mutation

The mutation scheme is the same as the one in section 3.3.2.

Result and analysis

The test for the system used the parameters in table 3.9.

As can be seen in figure 3.10, some span in the fitness space occurred, but it never climbed beyond a fitness value of 1.5. Individuals were mostly centered around the fitness evaluation in table 3.10. The quick stagnation in the fitness function seems to likely have to do with a mismatch between expected output behaviour and the actual output. In the first part of the fitness function, a voltage threshold is set at 0.05V as a point where the number of crossings are counted. This might still be a good idea, but not with a constant value when the output values have such varied widely different values across the entire test, with large chunks being way over. Instead of a constant 0.05V value, maybe a function could be used. Creating such a function however, seems a bit like having to solve the issue in one way, in order to be able to solve the problem in this way.

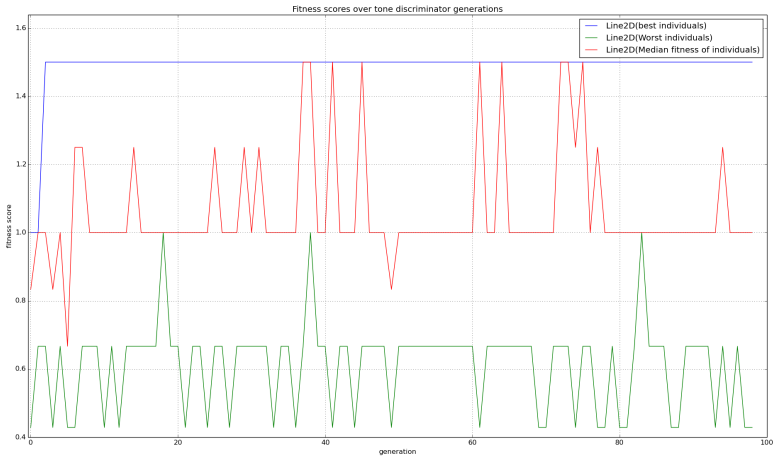

Figure 3.10: 100 generation evolutionary run with population size 20, attempting to evolve a discriminator for frequency differences for blinking leds

true positive:	0
true negative:	5
false positive:	0
false negative:	5

Table 3.10: Fitness evaluation most common in the experiment

Chapter 4

Experiments

During this thesis, a series of experiments were conducted in an effort to ascertain the feasibility of amorphous silicon solar panels as a physical medium of computation. The arrived upon manner of computation for this study was a series of attempts at evolving a frequency discriminator using an ”input” led blinking at different frequencies.

In the following sections the setup for the experiments is first laid out in detail. Then follows the main experiments. Finally follows a set of experiments devised to respond to some of the issues found in the main experiments.

4.1 Experiment setup

The conducted experiments are all variations of the approach that employs the Fast Fourier Transform, as described in chapter 3.3.1, Fast Fourier Transform. The first two rounds of experiments are about evolving a tone discriminator subject to a fitness function test where each individual is tested on how well they can discriminate on the test ”1010101010”. That is, a sequential test where the input pin runs with equal time, alternating between a given relative high frequent digital signal and a relative low frequent signal.

4.1.1 Evolutionary algorithm overview

The experiments are conducted using a custom made evolutionary algorithm made in python[2]. The algorithm is quite simple. The initial population of individuals is constructed by random generation of the starting frequency and assigned pins to use. Each individual is then tested on the Mecobo[19], thereby giving the individuals their fitness score. The best individual is then promoted to the next generation where it is cloned so that the population size is constant from generation to generation. Then, all individuals save one is mutated.

These experiments are subject to a set of constraints, see figure 4.1.

max generation	150
population size	25
frequency pairs (low, high)	(100, 1000), (200, 1500), (300, 1800)
fitness test time	5 seconds
fitness test	1010101010
Sampling frequency	100kHz
Configuration pin frequency range	[1Hz, 2.5kHz]
Number of input pins	1
Number of configuration pins	6

Table 4.1: General experiment parameters

Mutation scheme

When mutating a population of individuals all individuals are subjected to the same procedure. The pins to be mutated for each individual are random-generated, constrained by the number of configuration pins. For each of those pins there is a 40% that the pin gets a new random-generated frequency in the range of [1Hz, 2.5kHz]

4.1.2 Timing

The fitness function runs for a time of 5 seconds, see table 4.1, and a fitness test of ”1010101010”, each sub test is allotted 0.5 seconds each. The sampling frequency on the recording pin is at 100kHz, putting the sample size of the voltage output buffer at 500 000 voltage points for each test. Between each run of the fitness function there is a pause of 100ms to let the solar panel power down.

4.2 Experiment goals

The experiments revolve around making a frequency discriminator. That is, to evolve an electrical configuration such that the amorphous silicon solar panel may output voltage in recognition of whether it is seeing the relative low or relative high frequency signal.

4.2.1 Experiment layout

The experiments are structured as follows: The first and second round differ in using two different fitness functions. Each round tests the three frequency pairs in figure 4.1.

Each such test entails first evolving a frequency discriminator by randomized starting population. Then the best evolved individual from that evolutionary run is re-tested by running it through the fitness function again 10 times in order to assess reproducibility and stability of the result. Finally a new smaller evolutionary run of population 10 and maximum generation count 50 is conducted where the starting population is initialised as 10 copies of the best individual from the original evolutionary run. This is to assess whether the new evolution behaves differently, possibly reaching the old best fitness value fast, or at all.

Last and third round of experiments entail running a large amount of evolutions of relatively simpler fitness tests.

4.2.2 Representation of results

The graphs representing evolutionary runs are shown with fitness values per generation the course of the evolutionary run. Each generation has the fitness value of the best, median and worst individual.

The re-evolution graphs also contain a line denoting the best fitness value from the original evolution.

The the final evolution runs of simpler fitness tests are presented with histograms depicting final generation number for successful evolution.

4.3 An approximately smooth fitness function

This round of experiments employs a modified version of the fitness function used in[8], see equation 4.1. The values tp,tn,fp,fn are the same fitness constituent counters as in the algorithm in figure 3.4.

The domain of the fitness values is non-uniformly distributed in the range $[0, 25/36 \approx 0.699]$. A graph of this domain can be seen in appendix 6.2

$$\text{Fitness function} = \frac{tp \times tn}{(tp + fp + 1) \times (tn + fn + 1)} \tag{4.1}$$

4.3.1 Results

Frequency pair: (100Hz, 1000Hz)

The high, low frequency pair of 100Hz, 1000Hz experiment, see figure 4.1. The evolutionary run generally hits an apparent 'ceiling' on the fitness values early at 0.25, but finds briefly a higher scoring configuration between generations 87 and 102. Alternating between 0.34285714285714286 and 0.37037037037037035.

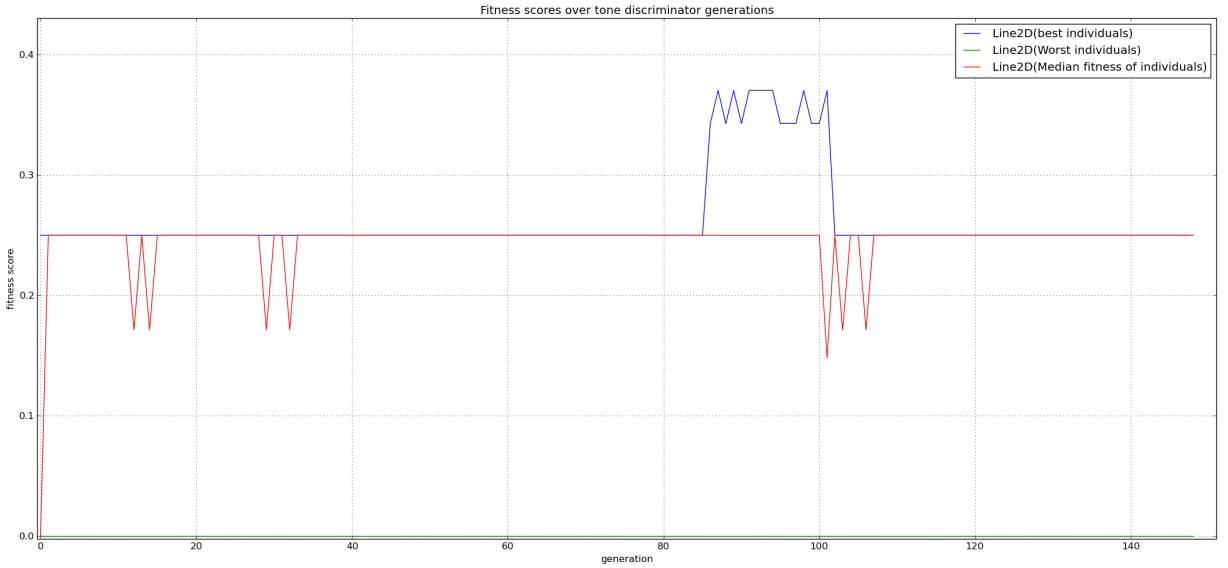

Figure 4.1: 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz.

Tested stability

The test of stability, see figure 4.2 indicates inability to achieve score as good as best logged score from evolution. In addition the fitness values are erratic, likely indicating unstable result.

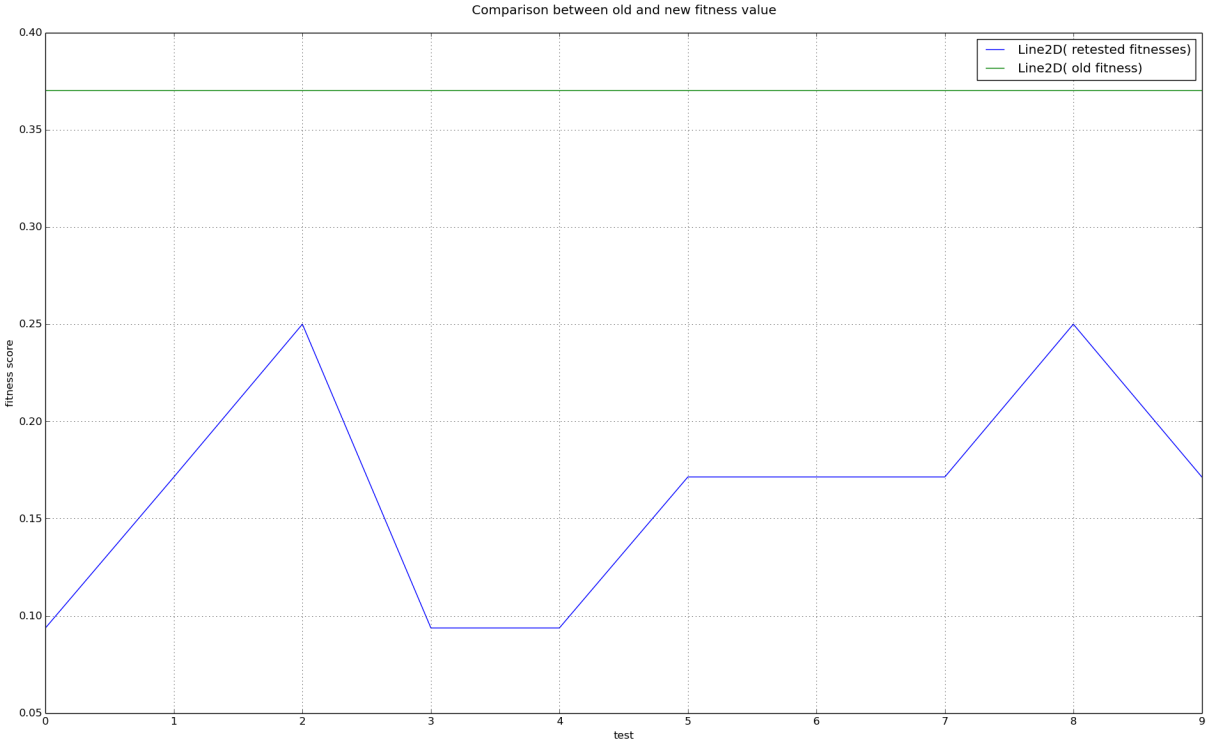

Figure 4.2: Comparison between best individual score from evolution and same individual run through the fitness function again 10 times.

Re-evolution

The re-evolution, see figure 4.3 does not improve beyond the fitness value 0.25 at fitness constituents $(3, 3, 2, 2)$. That is the same value that the original evolution largely topped out on. However, the values of the median and worst individuals throughout the evolutionary run, indicate exploration in the configuration space.

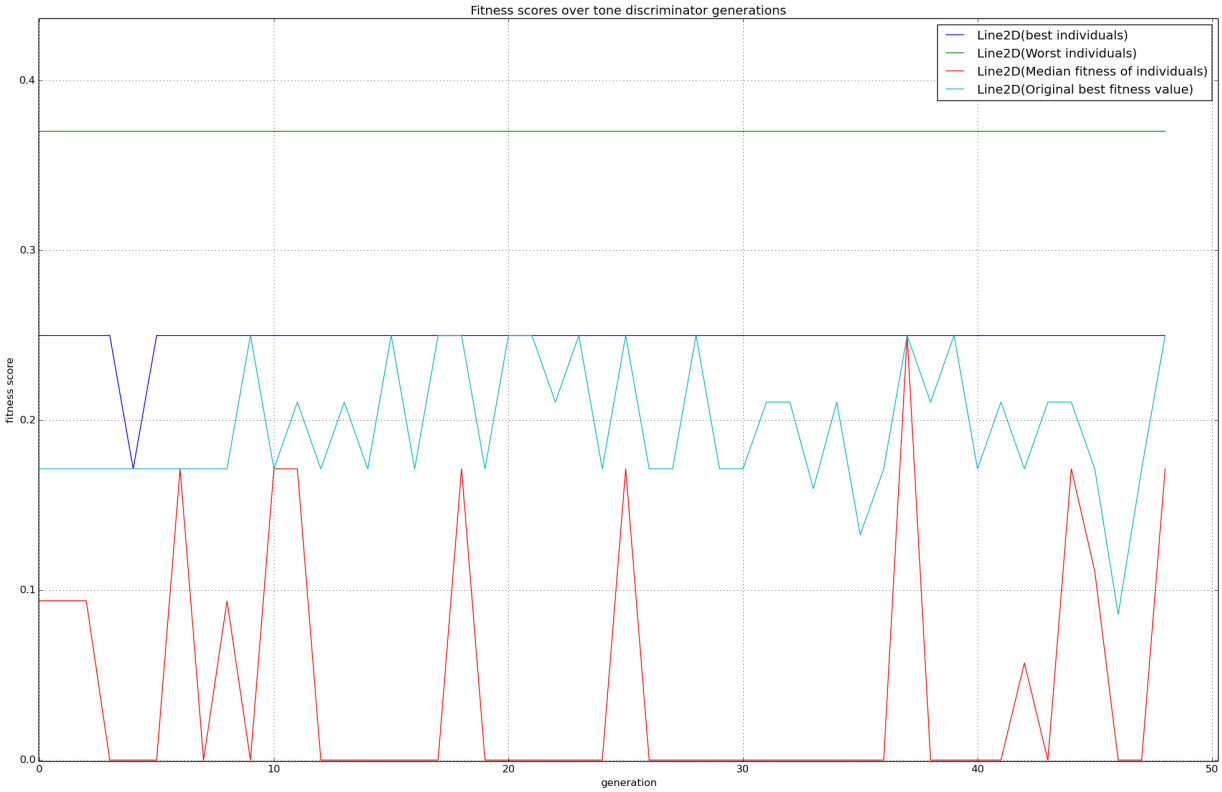

Figure 4.3: 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz. Best individual from original evolution seeded in and cloned as starting population

Frequency pair: (200Hz, 1500Hz)

The evolutionary run for the frequency pair of 200Hz, 1500Hz experiment, see figure 4.4 generally hits a 'ceiling'(the blue line) on the fitness values early at fitness value 0.46875, fitness constituents $(3, 5, 0, 2)$. However, it is generally not a lasting one and the search falls almost immediately to a general level of a lower value.

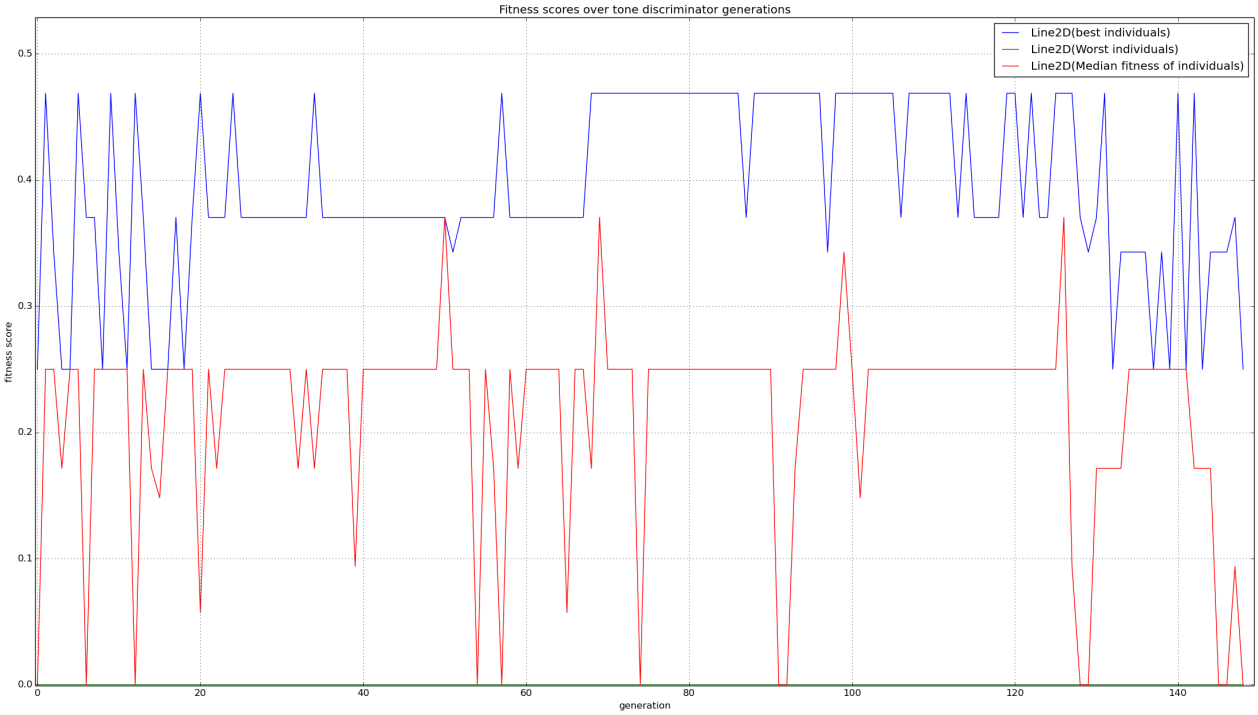

Figure 4.4: 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz.

Tested stability

The test of stability, see figure 4.5 indicates inability to achieve score as good as best logged score from evolution. In addition the fitness values are erratic, likely indicating unstable result.

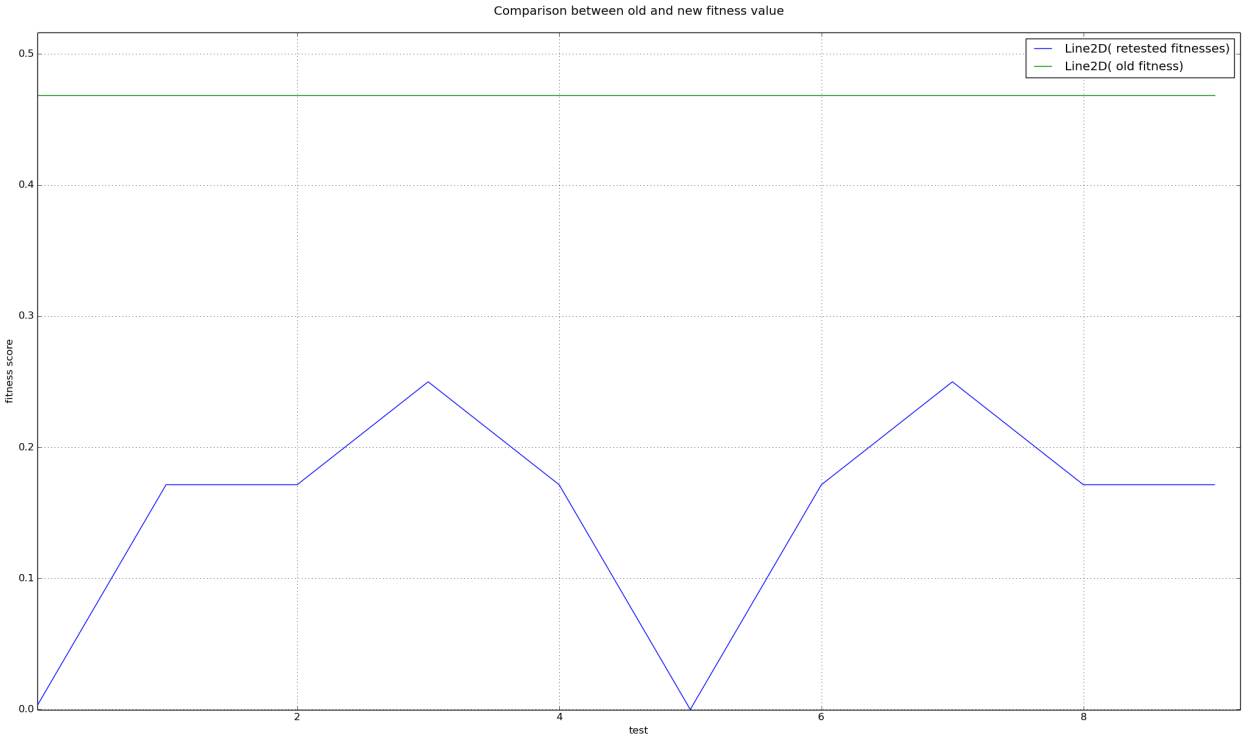

Figure 4.5: Comparison between best individual score from evolution and same individual run through the fitness function again 10 times.

Re-evolution

The re-evolution, see figure 4.6 goes through a somewhat steady climb up to the same fitness value ceiling as the original evolution when measuring the best individual. The median fitness score of individuals also tends slowly upwards in score.

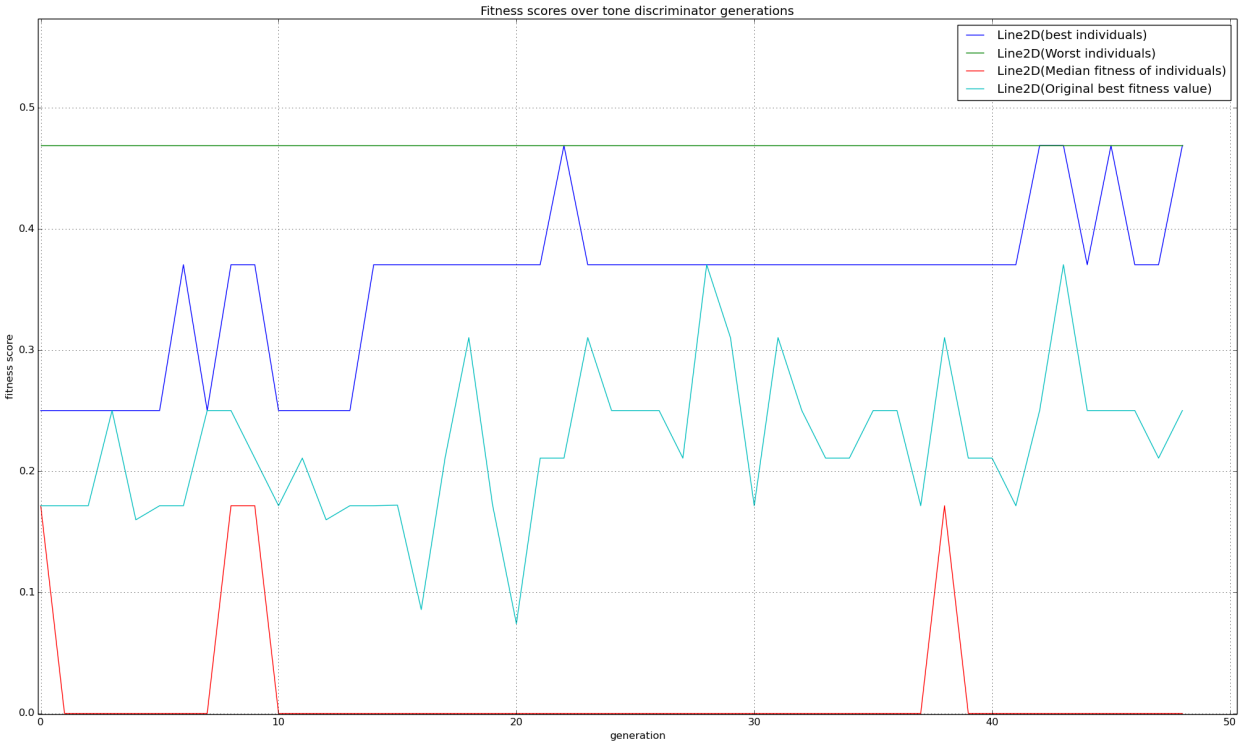

Figure 4.6: 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz. Best individual from original evolution seeded in and cloned as starting population

Frequency pair: (300Hz, 1800Hz)

The evolutionary run for the frequency pair of 300Hz, 1800Hz experiment, see figure 4.7 generally hits a 'ceiling'(the blue line) on the fitness values early, at 0.46875, with fitness constituents $(3, 5, 0, 2)$, see appendix

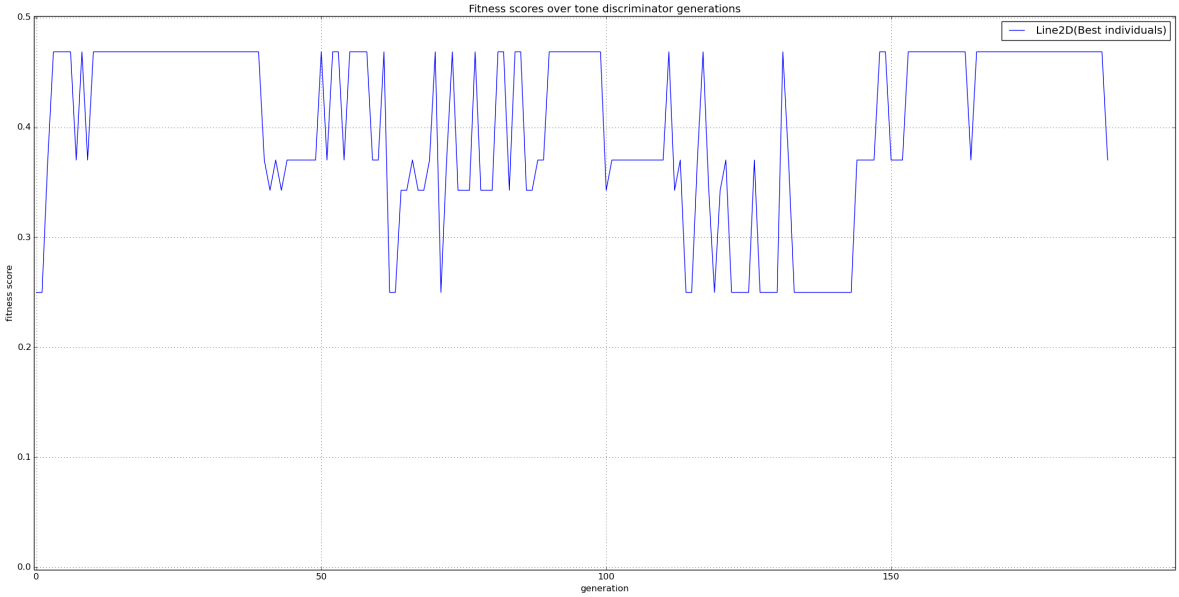

Figure 4.7: 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz.

Tested stability

The test of stability, see figure 4.8 indicates inability to achieve score as good as best logged score from evolution.

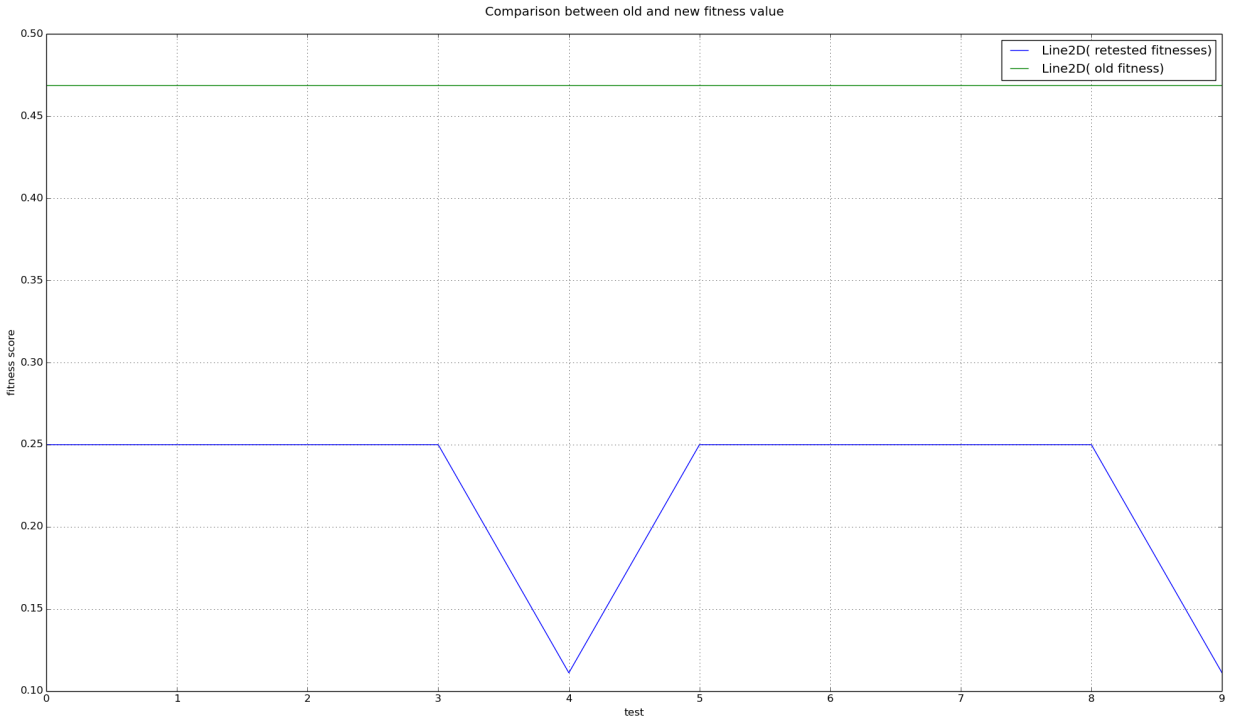

Figure 4.8: Comparison between best individual score from evolution and same individual run through the fitness function again 10 times.

Re-evolution

The re-evolution, see figure 4.9, goes through a somewhat steady climb upwards in the fitness value as measured by looking at the best individual and the median individual throughout the evolution, though the best value is still significantly lower than the best fitness score of the original evolution despite the original evolution hitting that score almost immediately.

Figure 4.9: 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz. Best individual from original evolution seeded in and cloned as starting population

4.4 A discrete fitness function

This round of experiments uses a simple fitness function, see equation 4.2. The values tp,tn,fp,fn are the same fitness constituents as in the algorithm in figure 3.4.

The domain of the fitness values is simply [0,10] on account of the fitness test being ”1010101010”, meaning at most 5 true positive and 5 true negative values.

$$\text{Fitness function} = tp + tn \tag{4.2}$$

4.4.1 Results

First pair: (100Hz, 1000Hz)

The evolutionary run for the frequency pair (100Hz, 1000Hz), see figure 4.10 climbs up to a ceiling of 8 at fitness constituents $(3, 5, 0, 2)$.

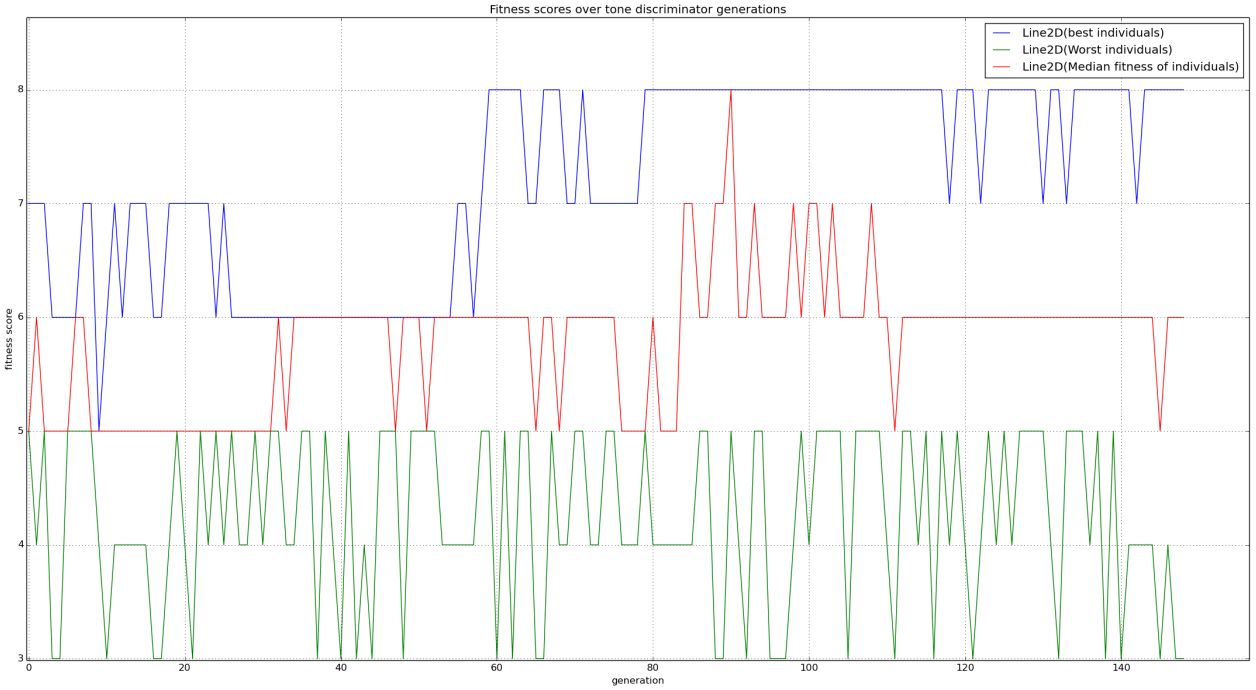

Figure 4.10: 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz.

Tested stability

The test of stability, see figure 4.11 indicates inability to achieve score as good as best logged score from evolution. In addition the fitness values are erratic, likely indicating unstable result.

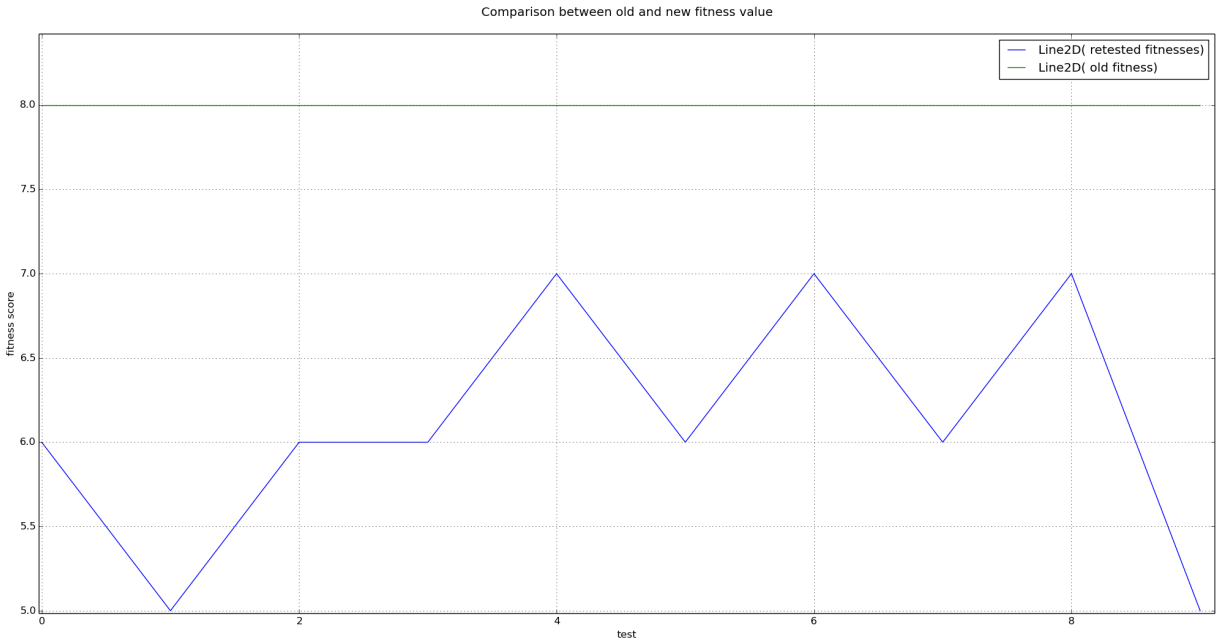

Figure 4.11: Comparison between best individual score from evolution and same individual run through the fitness function again 10 times.

Re-evolution

The re-evolution, see figure 4.12 finds largely a ceiling on the fitness score 7, though manages twice to achieve the same fitness score as the original evolution.

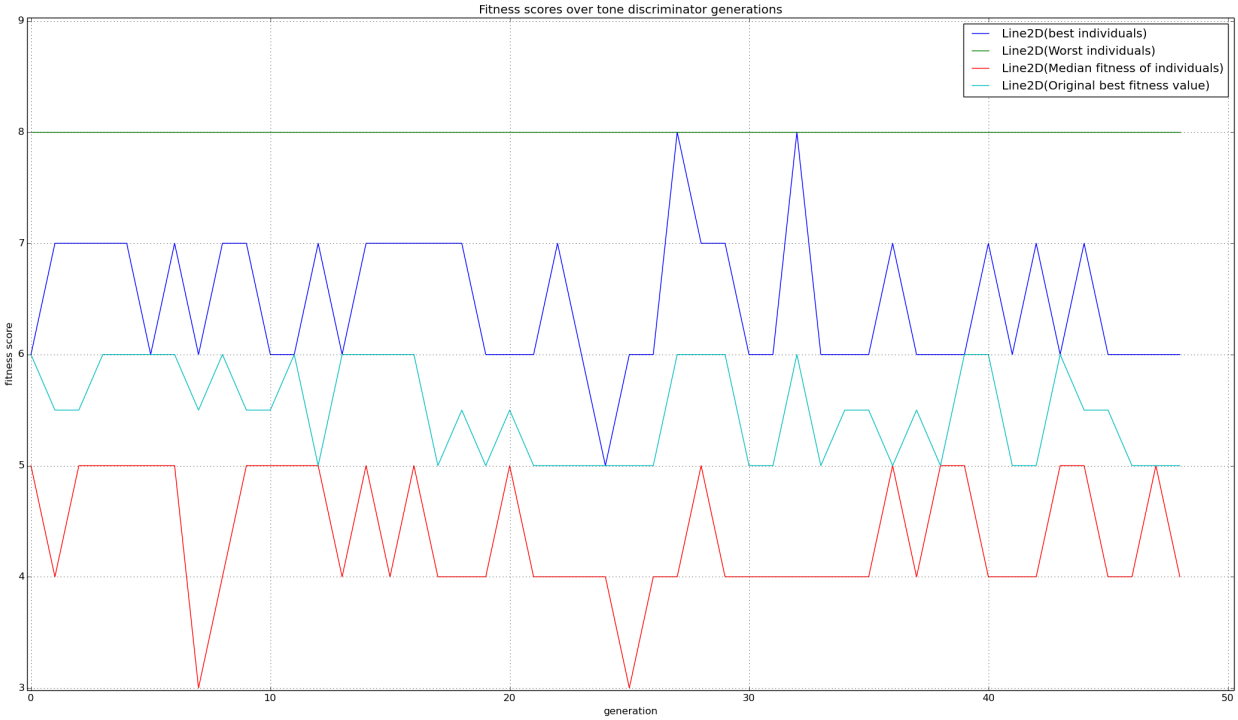

Figure 4.12: 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 100Hz and 1000Hz. Best individual from original evolution seeded in and cloned as starting population

Second pair: (200Hz, 1500Hz)

The evolution, see figure 4.13 spends the first 29 generations unable to find a configuration space that yields a better fitness score than 6. From then until generation 62, the fitness score erratically climbs up to 8, with the fitness constituents being $(3, 5, 0, 2)$. Though the 8 fitness score is not consistent and regularly drops.

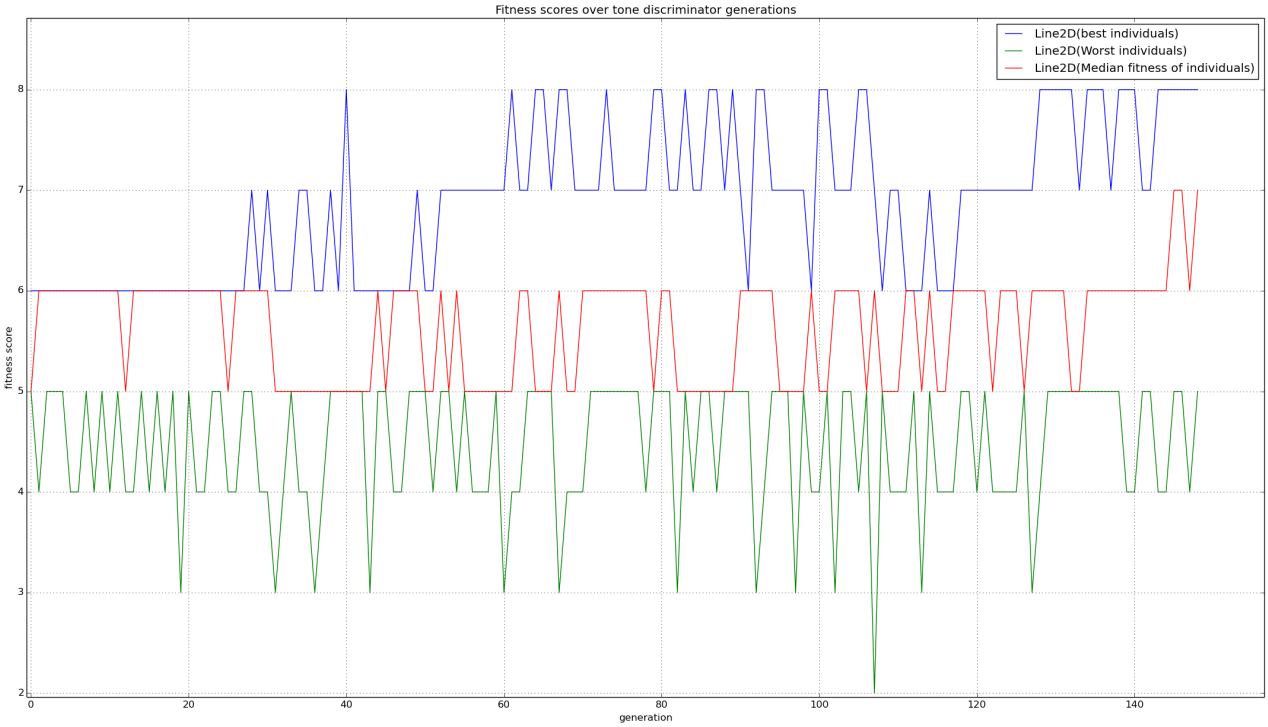

Figure 4.13: 150 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz.

Tested stability

The test of stability, see figure 4.14 indicates inability to achieve score as good as best logged score from evolution. In addition the fitness values are erratic, likely indicating unstable results.

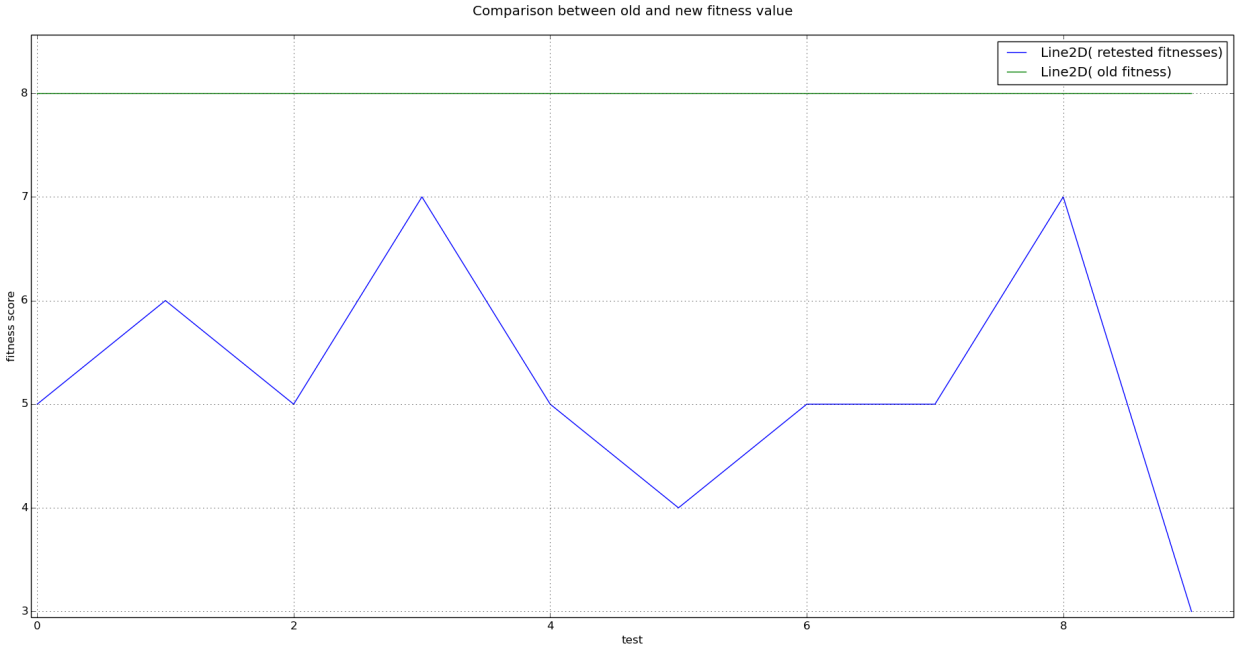

Figure 4.14: Comparison between best individual score from evolution and same individual run through the fitness function again 10 times.

Re-evolution

The re-evolution, see figure 4.15 erratically hits the highest fitness score as the original evolution. This occurs somewhat faster though similarly erratic in dropping to a fitness score of 7.

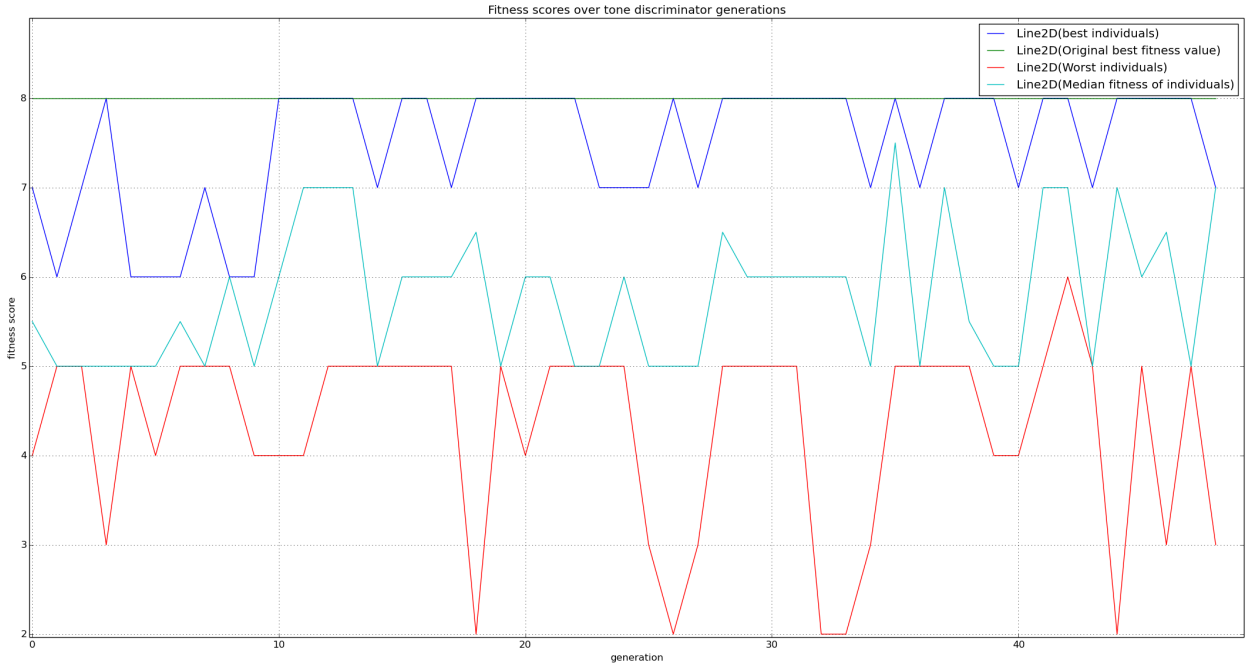

Figure 4.15: 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 200Hz and 1500Hz. Best individual from original evolution seeded in and cloned as starting population

Third pair: (300Hz, 1800Hz)

The evolutionary run for the frequency pair of 300Hz, 1800Hz experiment, see figure 4.16. The fitness score undergoes a 'climb' in the first 63 generations, and then generally hits a 'ceiling'(the blue line) on the fitness values early, at 8, with fitness constituents $(3, 5, 0, 2)$. Though These scores are erratic and drop quickly to 7.

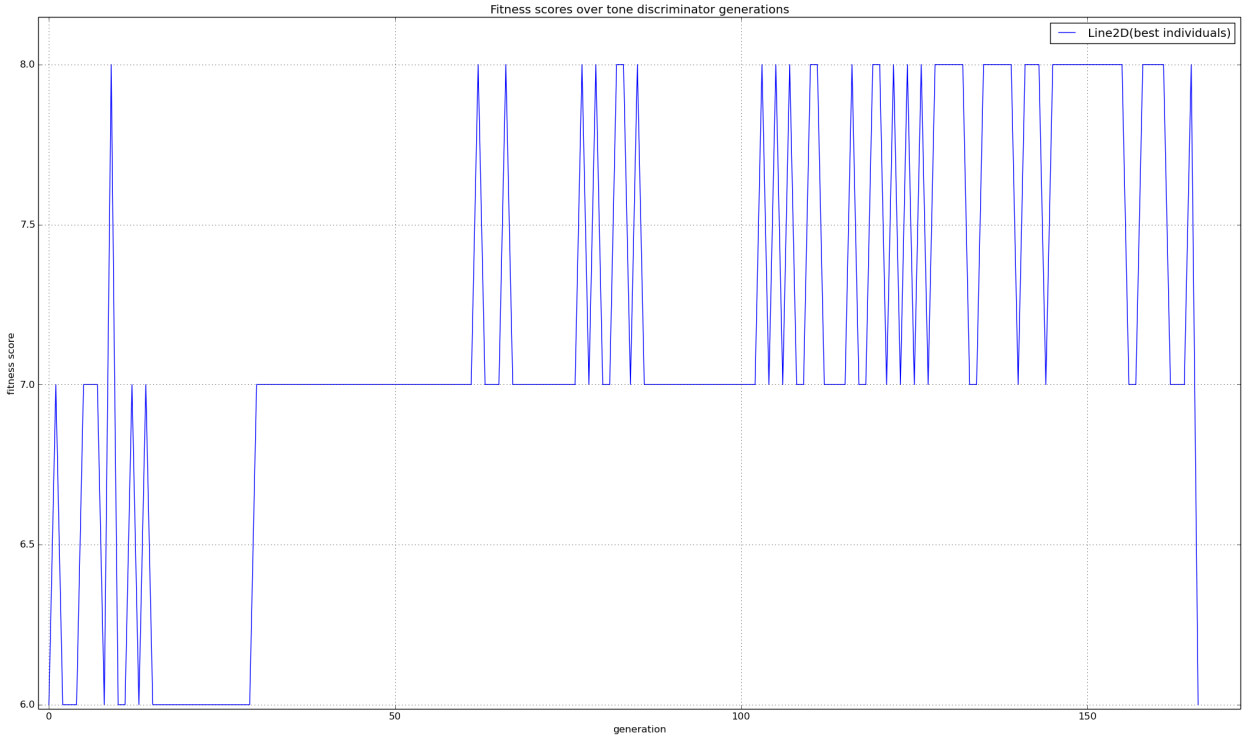

Figure 4.16: 168 generation evolutionary run with population size 25, attempting to evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz. Be advised the y-axis differs from the other graphs in only showing range 6-8 on the fitness function.

Tested stability

The test of stability, see figure 4.17 indicates inability to achieve score as good as best logged score from evolution.

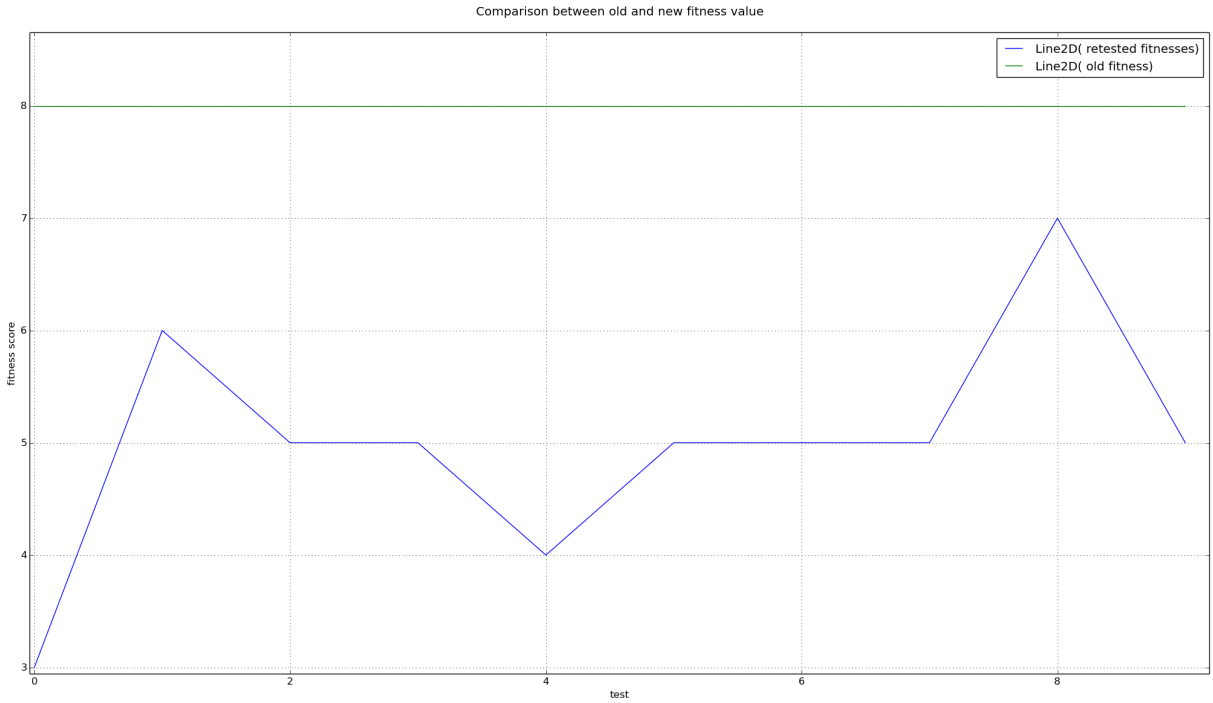

Figure 4.17: Comparison between best individual score from evolution and same individual run through the fitness function again 10 times.

Re-evolution

The re-evolution, see figure 4.18, spends the first 18 evolutions exploring the configuration space between the fitness scores 6 and 7. Then follows a few generations exploring between the fitness scores $6, 7, 8$, before the last generations where the best fitness score settles into a run of fitness score 7 with fitness constituents $(4, 3, 2, 1)$ and $(2, 5, 0, 3)$, save generation 35-39 where configurations getting a fitness score of 8 with fitness constituents $(3, 5, 0, 2)$ are achieved.

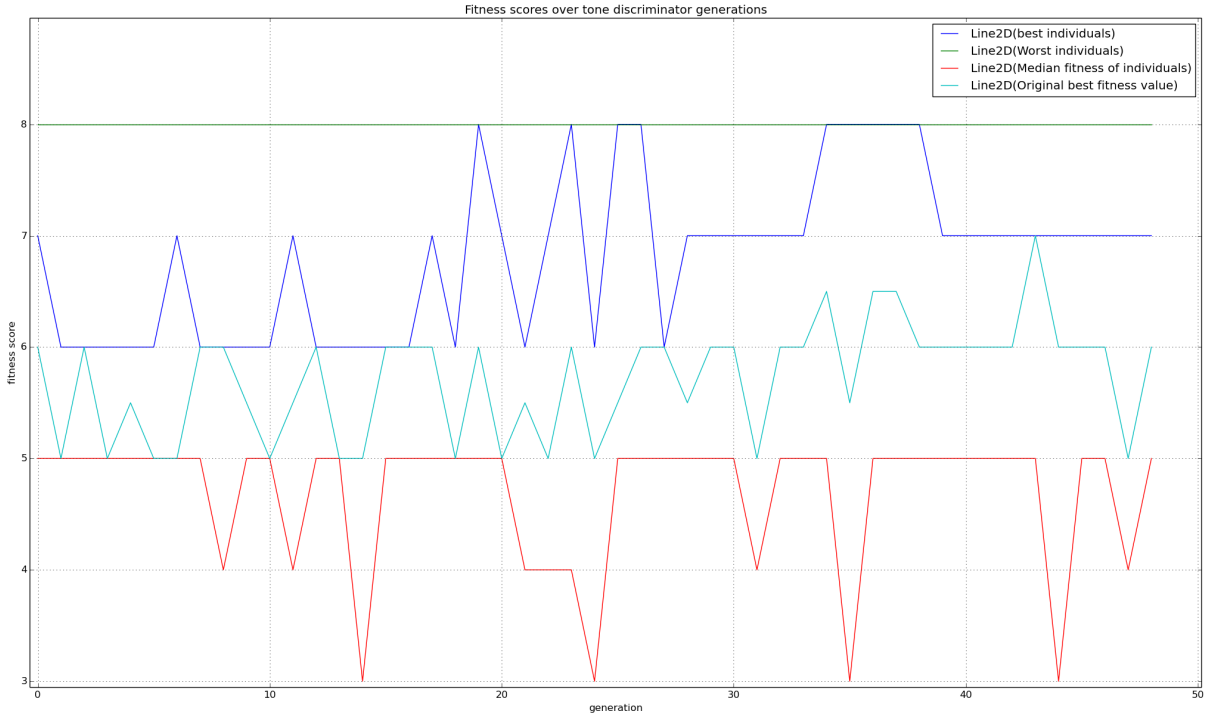

Figure 4.18: 50 generation evolutionary run with population size 10, attempting to re-evolve a discriminator for blinking leds, where the frequency difference pair is 300Hz and 1800Hz. Best individual from original evolution seeded in and cloned as starting population

4.5 Frequency discriminator: reduced scope

In response to some of the patterns that emerged from the previous experiments, a set of experiments were carried out to seek clarification on what was observed. All of the previously conducted experiments used the fitness input test ”1010101010”. The following experiments pertain to the case of evolving a blinking tone discriminator for the inputs ”10”, ”1010” The final series of experiments were subject to the parameters in figure 4.2.

4.5.1 ”10” evolution target

A series of 80 evolutionary runs were conducted, attempting to evolve a frequency discriminator on the abstract ”10” fitness test. All 80 evolutionary runs were successful. Figure 4.19 shows a histogram of the ’final generation’ distribution for the successful evolutions; where the fitness score reached possible maximum score of $tp + tn = 1 + 1 = 2$ and therefore ended the specific evolutionary run.

max generation	30
population size	10
frequency pair (low, high)	(100Hz, 1000Hz)
(fitness tests, time)	(10, 1sec), (1010, 2sec)
Sampling frequency	100kHz
Configuration pin frequency range	[1Hz, 2.5kHz]
Number of input pins	1
Number of configuration pins	6
fitness function	tp + tn, see equation 4.2

Table 4.2: General experiment parameters

4.5.2 ”1010” evolution target

A series of 80 evolutionary runs were conducted, attempting to evolve a frequency discriminator on the abstract ”1010” fitness test.

A series of 80 evolutionary runs were conducted, attempting to evolve a frequency discriminator on the abstract ”1010” fitness test. All 80 evolutionary runs were successful. Figure 4.20 shows a histogram of the ’final generation’ distribution for the successful evolutions; where the fitness score reached possible maximum score of $tp + tn = 2 + 2 = 4$ and therefore ended the specific evolutionary run.

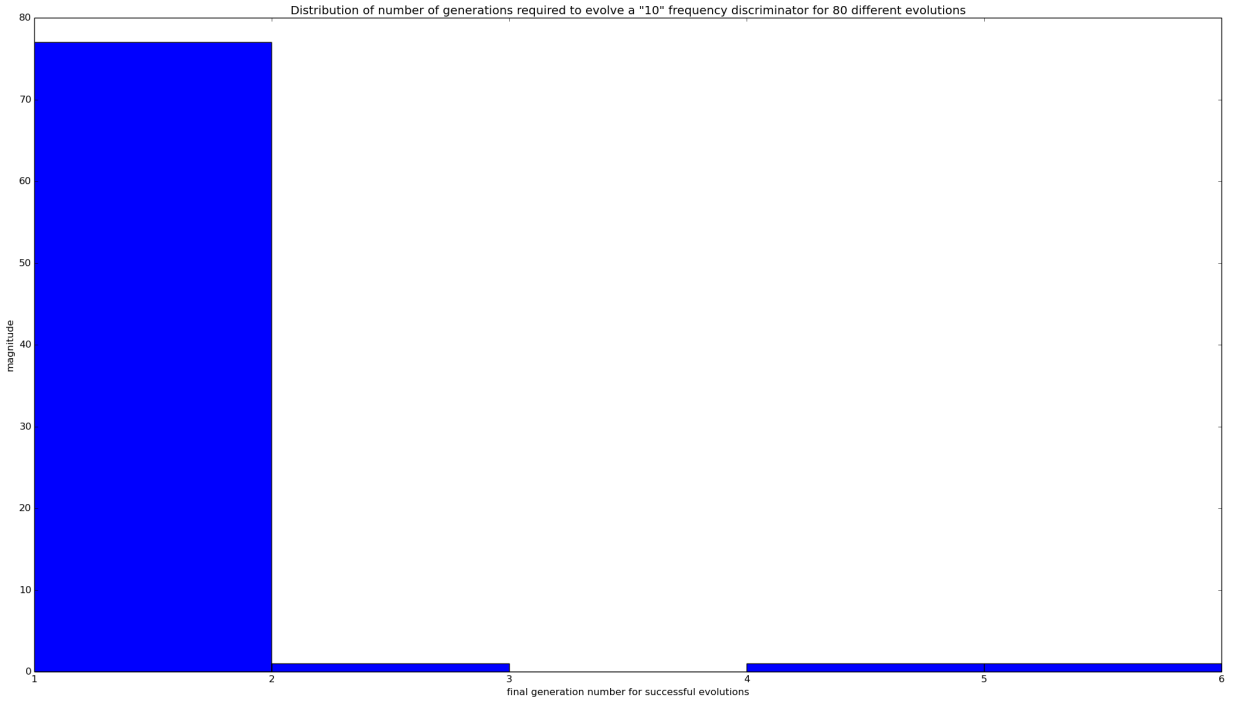

Figure 4.19: Final generation number for successful evolutions that evolved a ”10” blinking led frequency discriminator

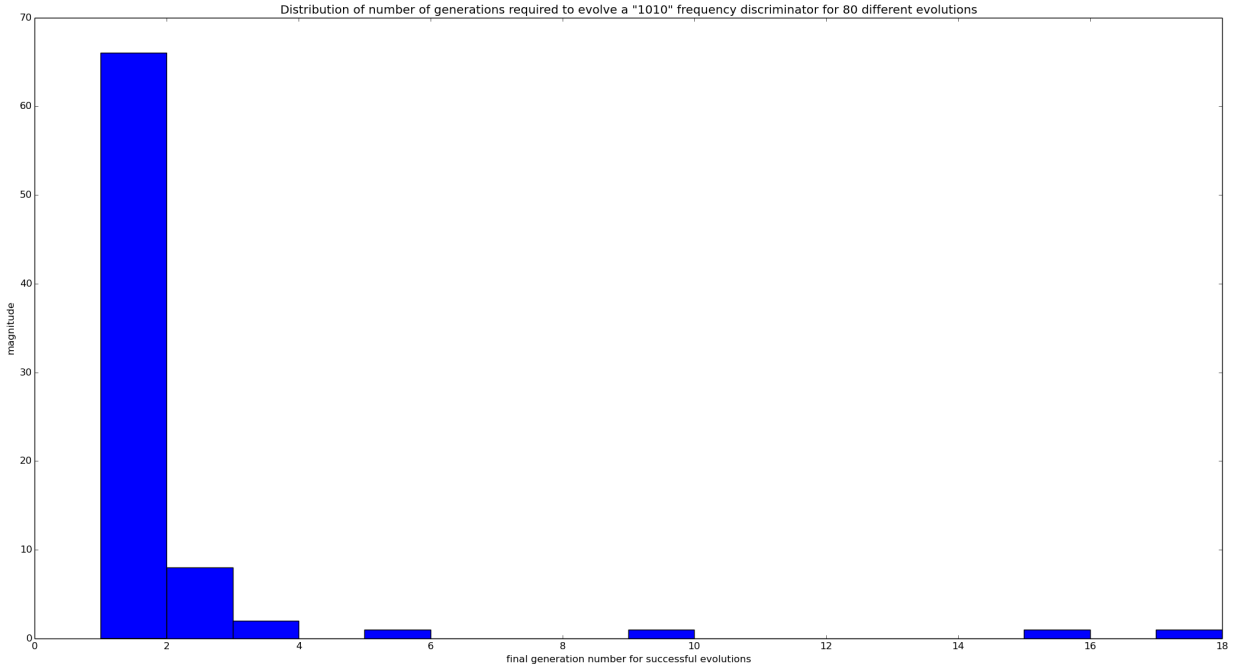

Figure 4.20: Final generation number for successful evolutions that evolved a ”1010” blinking led frequency discriminator

Chapter 5

Analysis

The experiments indicate that it is possible for Evolution-in-Materio style computation to be developed in amorphous silicon solar panels. Several attempts at evolving a 'blinking led' frequency discriminator were conducted and a pattern of partial success emerged across the experiments.

5.1 Viability of approach

The experiment results are mixed in terms of successful evolution of a frequency discriminator. Most of the experiments see some evolved success over their respective experiment running time. Some recurring patterns are discussed here.

5.1.1 The fitness functions

The motivation behind the first type of experiments with the 'Approximately smooth' fitness function, see equation 4.1, was that it might lend itself to gradual improvements in the fitness score space. It is a modified version on the fitness function used in [8], accounting for the possibility of $tp + fp = 0$ or $tn + fn = 0$, which would have yielded divide by zero in the original function.

Combined with the clone-and-mutate approach, the idea was that a hill-climbing-like process might be achieved. A peak could be found, and from there on, the configuration space could be explored by the rest of the mutated population, in a manner somewhat inspired by the general working of the Simulated annealing algorithm[18].

While the first experiments with 100Hz, 1000Hz did not yield much success. However, both the 200Hz, 1500Hz, experiment and the 300Hz, 1800Hz experiment resulted in best fitness scores of 0.46875 at fitness constituents: (true positive: 3, true negative: 5, false positive: 0, false negative: 2). They also found such a configuration almost immediately, see figures 4.4 and 4.7, hitting this 'ceiling' erratically for the rest of the evolution.

The same type of results were found with equivalent fitness constituents $(3, 5, 0, 2)$ at a fitness score of 8 with the discrete version, see equation 4.2. This shows that the idea

that the approximate smooth fitness function would yield better ability to climb the fitness scores might be wrong.

For each of the 'original' evolutions under the discrete fitness function experiments, see figures 4.10, 4.13 and 4.16, there is loosely the same pattern. During the approximately 60 first generations in each case are spent with the best individuals staying in the 6-7 fitness score range. After that they either entirely or mostly pass over to 7-8.

This is somewhat different from the 'smooth' evolutions where the 100Hz, 1000Hz is a bit of a comparatively dysfunctional outlier on the whole, and the 200Hz, 1500Hz pair, and the 300Hz, 1800Hz pair ended up with comparatively similar results as all the discrete experiments save that they found their solutions much faster.

5.1.2 Apparent Ceiling of computations

The apparent ceiling of fitness constituents is interesting, both in that it is consistent across all experiments(except the 'smooth' 100Hz, 1000Hz experiment that did not reach it) and in that there was typically a movement in the best individuals in the range of the higher fitness scores. One possibility might be that a higher maximum generation number could have yielded a greater climb. Another possibility however, could be that the best results ever achieved in the main experiments, were the best possible configuration space to evolve. While the exact configuration frequencies of the best individuals varied, all best fitness constituents had the pattern $(3, 5, 0, 2)$. 3 true positives and 5 true negatives.

5.1.3 Computation space

On the topic of whether it is at all possible to develop computation in amorphous silicon solar panels, the approach undertaken in this project is perhaps the most likely candidate for success. That is, a frequency discriminator on differing high frequency, low frequency blinking lights seems a good fit as a foundational structure already exists in the sense that the material already possesses the ability to generate electricity when light is applied.

What else must be built? Evolution-in-Materio evolution develops some manner of circuitry in the material under test and in this project that would be the pattern of excited material during the run of the fitness function. The configuration pins causing their respective connected leds to blink and the input pin running its test input, switching its led between the high frequent blinking and low frequent blinking. Over the course of the run of the fitness function something seems to be developed in the material.

The results in figures 4.1, 4.3, 4.4, 4.7, 4.12 have a development throughout their evolutions that make it difficult to say that they undergo an actual development rather than a randomized search in the configuration space. Or perhaps, that the evolutionary algorithm is not so much of help there, and the 'better' part of the results merely shows the span of the mutated populations.

There were some results however, where there seemed to be more grounds for arguing that actual development occurred. Figure 4.6 shows a clear transition from generation 15 to a consistently better fitness score for the best individuals plot. Furthermore, As the fitness scores for the best individual improves more towards the end, so appears the median individual score.

Figures 4.9, 4.10, 4.13 and 4.16 shows a consistent first period of a relatively lower fitness score and then a fitness score climb at generation 33 for the best individuals. This climb plateaus at a higher fitness score.

The developed circuitry is unknown, but the evolutionary behaviour seems consistent enough to assert that these 'climbs' represent some actual change in the circuitry.

Reducing scope

The final experiments, see figure 4.19 and 4.20 represent an effort to understand if the different evolutions of the "1010101010" frequency discriminator perhaps built up recognition of the fitness test through development of some memory circuitry. The nebulous idea being that perhaps a climb in fitness score similar to the aforementioned experiments would occur, but maybe faster. The actual results were that both cases of 80 evolutionary runs each saw near instantaneous success. Every single run in both cases reached their maximum fitness scores. At a population size of 10, the "10" frequency discriminator, see figure 4.19, saw 77 out of 80 evolutionary runs hit maximum fitness score on generation 1. The "1010" frequency discriminator, see figure 4.20, saw 66 out of 80 attempts get maximum fitness score on generation 1.

One interpretation of that data is that the solar panel is simply very suitable for frequency discrimination. The "10" case can on its own be viewed as a linear function. "Output higher signal first, then low." That it almost instantly found a correct configuration for satisfying the fft-to-dominating-frequency calculation, see chapter 3.3.1 is slightly less so. But that the "1010" frequency discrimination evolutions also gets such a high near-instant success might indicate that frequency discrimination is in fact a quite simple task for the solar panel. If that is the case then perhaps the problems with stalled out fitness scores in the original experiments could be dealt with.

5.2 Stability of results

As is demonstrated in every test of stability, see figures 4.2, 4.5, 4.8, 4.11, 4.14, 4.17, two things stand out. Number one is that the re-tested individuals never achieve the fitness score of the original evolution from which they came. Second is that their scores are erratic, fluctuating in a span of scores. This shows that the evolved configurations are highly volatile.

The re-evolutions, seeding in the best individual and cloning it as the starting population shows this very clearly. None of the re-evolutions manage to re-create the fitness scores from the original evolutions from the beginning(nor supersede the old best fitness score for that matter). The re-evolutions that managed to re-create the equivalent fitness score of the original evolution were 4.6, 4.12, 4.15 and 4.18. The discrete fitness function version of the high frequency 1500Hz, low frequency 200Hz re-evolution, see figure 4.15, was the fastest re-evolution to re-obtain the equivalent fitness score of the original evolution at generation 4.

This shows that 'saving' an individual for reuse garners no advantage in the project as configurations only seem useful within an evolutionary run.

5.2.1 Dependencies between tests

All of the evolutions have a particular oddity about them. The developed evolutionary algorithm explicitly promotes the best individual of a generation, clones it and mutates everyone in the new generation save one, see chapter 3.3.3. This should mean that the result graphs all show the 'best individuals' plot as stagnant or increasing. Never decreasing. That is never the case in any of the experiments. This could indicate electrical charge still remaining in the solar panel between tests, thereby causing one test to affect the next one.

5.2.2 Environmental effects

The hardware for the experiments was situated in a laboratory in which, several experiments were conducted and all manners of electrical work was performed. The solar panel might have been insufficiently shielded from other light sources, such as additional light slipping in through the holes were the leds were situated. And with a lot of computers and occasional soldering occurring in the lab, varied heat might be a factor that affected results.

5.3 Using the Mecobo system

The Mecobo platform is a research project and some issues were had with the operational stability of the system while running. It was on account of this that the maximum 150 generation number was set, as it became somewhat common for the system to crash beyond that point.

5.4 Design constraints

By the time the main experiments were conducted, the decision had been made to simplify the workings of individuals. The original setup for an individual, see figure 3.3, would create individuals as having more complex configuration pins then the final design. The current state has a configuration pin having one frequency over the course of the entire fitness function run, and this frequency value is subject to randomization if it is selected for mutation.

The original design would see a configuration pin be assigned a list of chronologically non-overlapping, potentially differing frequencies in a given time-slice over the course of the fitness function run. The duration of a particular frequency could vary. Mutation, should the given configuration pin be selected, would then be that a [time-slice,frequency] pair would be selected at random. Then, both the frequency, and the duration could be mutated. That is, the time-slice could be extended or contracted(subject to the global time constraint of how long the fitness function runs), and the frequency value could be altered just as in the final design.

The reason for going with a simpler design was that a more complex design might make it difficult to interpret results. However, the stalling-out fitness scores and fitness score climbs observed in the results might mean it would be prudent to try it, as it might

be able to provide more fine-grained resources for the evolutionary algorithm instead of time-wise global frequencies on the configuration pins.

Chapter 6

Conclusion

This thesis has explored the idea of developing computation in an amorphous silicon solar panel through the means of Evolution-in-Materio. The goal was to study the capacity for developing computation in the material using light. The experiments presented in the thesis show that repeatable and functional, though unstable frequency discriminators can be successfully evolved in an amorphous silicon solar panel using controlled exposure of light. This means that there might be some viability to the material/controlled light exposure scheme as an approach to developing computation.

6.1 Future work

Evolution-in-Materio can be a powerful field when done correctly, using an evolutionary algorithm to configure a physical material for computation. A path one could take to take this work further could be to re-create the fitness function as consisting of a fitness score as used in this thesis and a test of reproducibility. A persistence-of-result test that seeks to reward stable configurations.

A scheme used in the project is to take the fast-fourier-transformed voltage buffers3.3.1, seek out the dominating frequency and then merely checking if that frequency is smaller than the average value of the current experiment frequency pair or not. Assigning a logic value accordingly. Another way could be to take the spectral centroid[25] of the fast-fourier-transformed voltage buffers, thereby calculating a value based on the whole frequency spectrum, not just which one is dominating.

As is mentioned in section 5.4, implementing a more complex setup for the individuals could be worthwhile in order to hand more fine-grained control over to the evolutionary algorithm.

Evolution-in-Materio might be showing a glimpse of a future where perhaps part of the computer design process comes to include an evolutionary process complementing the human design thinking strengths. Serving particular constraints as the input and having

the evolutionary algorithm seek out a configuration, or possibly configuration space which the designers can take and work with.

Bibliography

[1] , 2007. Apache software foundation. thrift.

[2] , 2010. Python software foundation and guido van rossum, python 2.7.

[3] , 2012. Canonical ltd. and ubuntu community, ubuntu 12.10.

[4] , 2016. *NASCENCE project website*. http://nascence.no [Accessed: 2016-11-13].

[5] , 2017. Flexible amorphous silicon thin film solar panel 2 v / 0.5 W solar cell flexible DIY solar panels. https://www.alibaba.com/product-detail/Flexible-amorphous-silicon-thin-film-solar_60477318002.html, [Accessed: 2017-02-13].

[6] , 2017. MECOBO software and harware development project site. https://github.com/NASCENCE/mecobo, [Accessed: 2017-02-10].

[7] , 2017. NASCENCE materials. http://nascence.no/index.php/materials, [Accessed: 2017-02-13].

[8] Broersma, H., Miller, J. F., Nichele, S., 2017. Computational Matter: Evolving Computational Functions in Nanoscale Materials. Springer International Publishing, Cham, pp. 397–428.
URL http://dx.doi.org/10.1007/978-3-319-33921-4_16

[9] David Patterson, J. H., 2012. Computer Organization and Design. Computer Organization and Design.
URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:No+Title{#}0

[10] Farstad, S., 2015. Evolving cellular automata in-materio[Accessed: 2016-03-08].
URL http://arkt.is/evolving-cellular-automata-in-materio-preproject.pdf

[11] Fernando, C., Sojakka, S., 2003. Pattern Recognition in a Bucket. In: Advances in Artificial Life. pp. 588–597.
URL http://www.springerlink.com/content/xlnymhf0qp946rce

[12] Gentleman, W. M., Sande, G., 1966. Fast fourier transforms: For fun and profit. In: Proceedings of the November 7-10, 1966, Fall Joint Computer Conference. AFIPS '66 (Fall). ACM, New York, NY, USA, pp. 563–578.
URL http://doi.acm.org/10.1145/1464291.1464352

[13] Harding, S., Miller, J., 2005. Evolution In Materio: Evolving logic gates in liquid crystal. Unconventional Computing 2005: From Cellular Automata to Wetware 2005, 133–148.

[14] Harding, S., Miller, J. F., 2004. Evolution in materio: A tone discriminator in liquid crystal. Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004 2, 1800–1807.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-4444334719{&}partnerID=tZOtx3y1

[15] Harding, S., Miller, J. F., 2005. Evolution in materio: A real-time robot controller in liquid crystal. In: Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on. IEEE, pp. 229–238.

[16] Hornby, G. S., Lohn, J. D., Linden, D. S., 2011. Computer-automated evolution of an X-band antenna for NASA's Space Technology 5 mission. Evolutionary computation 19 (1), 1–23.

[17] Iannucci, R. A., 1988. Toward a dataflow/von Neumann hybrid architecture. ACM SIGARCH Computer Architecture News.

[18] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., 1983. Optimization by simulated annealing. SCIENCE 220 (4598), 671–680.

[19] Lykkebø, O. R., Harding, S., Tufte, G., Miller, J. F., 2014. Mecobo: A hardware and software platform for in materio evolution. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 8553 LNCS. pp. 267–279.

[20] Marcus, M., Akera, A., 1996. Exploring the architecture of an early machine: The historical relevance of the ENIAC machine architecture. IEEE Annals of the History of Computing 18 (1), 17–24.

[21] Miller, J. F., Downing, K., 2002. Evolution in materio: Looking beyond the silicon box. In: Proceedings - NASA/DoD Conference on Evolvable Hardware, EH. Vol. 2002-January. pp. 167–176.

[22] Miller, J. F., Harding, S. L., Tufte, G., 2014. Evolution-in-materio: Evolving computation in materials.

[23] Mohid, M., Miller, J. F., 2015. Evolving robot controllers using carbon nanotubes. In: Proceedings of the European Conference on Artificial Life. pp. 106–113.

[24] Pask, G., 1959. The natural history of networks. Proceedings of International Tracts In Computer Science and Technology and their Application 2, 232–263.

[25] Peeters, G., 2004. A large set of audio features for sound description (similarity and classification) in the CUIDADO project, (section 6.1.1). CUIDADO IST Project Report 54 (0), 1–25.
URL http://www.citeulike.org/group/1854/article/1562527

[26] Snow, E. S., Novak, J. P., Campbell, P. M., Park, D., 2003. Random networks of carbon nanotubes as an electronic material. Applied Physics Letters 82 (13), 2145–2147.

[27] Thompson, A., 1997. An evolved circuit, intrinsic in silicon, entwined with physics. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 1259. pp. 390–405.

[28] Wolfram, S., 1983. Statistical mechanics of cellular automata. Reviews of Modern Physics 55 (3), 601–644.

[29] Wolfram, S., 2002. A new kind of science. Vol. 5. Wolfram media Champaign.

Appendix

6.2 Fitness function smoothness curve

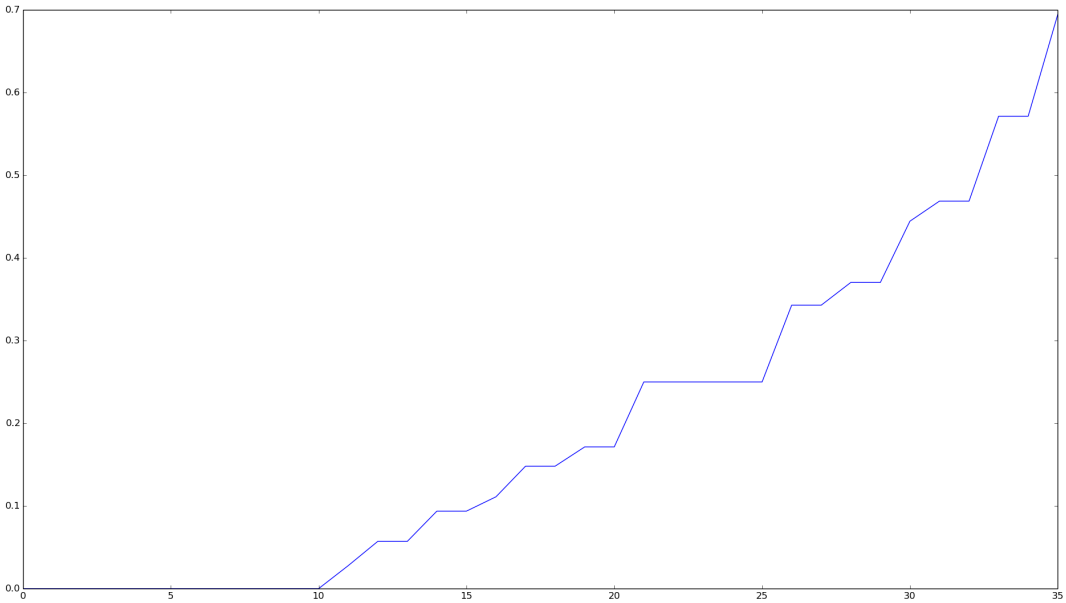

Figure 6.1: Distribution of theoretically obtainable fitness scores given a fitness test with binary representation "1010101010" and the fitness function F where $F$is $F = \frac{tp*tn}{(tp+fp+1)*(tn+fn+1)}$. The beginning left side of the plot shows all the instances where the numerator is zero and then the plot climbs all the way up to a maximum score where the correctness buckets look thusly: $tp : 5, tn : 5, fp : 0, fn : 0$ making the fraction $\frac{5*5}{(5+0+1)*(5+0+1)} \rightarrow \frac{25}{36} \approx 0.7$ as can be seen on the right side of the curve. The curve is not smooth, but possibly smooth enough to allow for somewhat fluid fitness score movement.

6.3 Code repository and experiment results

The zip file uploaded with the thesis contains the code repository and the experiment results.