Fast Evolution Computing for Synthesizer Parameter Finding

Cillian Patrick Greene
14311676

Final Year Project – 2018
B.Sc. Computer Science/Computer Science and Software Engineering

Department of Computer Science

Maynooth University

Maynooth, Co. Kildare

Ireland

A thesis submitted in partial fulfilment of the requirements for the

B.Sc. Computer Science and Software Engineering

Supervisors: Dr. Joe Timoney, Dr. Edgar Galvin

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study as part of B.Sc. in Computer Science & Software Engineering qualification, is *entirely* my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

I hereby acknowledge and accept that this thesis may be distributed to future final year students, as an example of the standard expected of final year projects.

Signed: Cillian Greene Date: 27/03/2018

Acknowledgements

I'd like to thank both my supervisors for helping me find my way through this project. I ran into many problems, but both Joe and Edgar were of great help whenever I needed it.

Abstract

Musical instruments are programmed using sound synthesis techniques. These techniques can be time consuming and inefficient. This project aims to use a process to solve these problems called Evolutionary Algorithms (EA), specifically Differential Evolution (DE) which is a subset of EA. Using a sound synthesizer, these algorithms will try to replicate the sound from a real musical instrument, specifically a live recording of a trumpet. Two different methods were chosen to test the viability of EA, one method which is a mathematical function called the objective method and the other which uses human judgement as a measurement called the subjective method. In both methods, the fitness of the sounds improved over time as the variants were run. The objective method gives inconsistent results, while the subjective method replicates sounds on average far better.

List of Tables

Chapter 1: Introduction

1.1 Topic Addressed in this Project

The main topic of this project is to develop an evolutionary algorithm that will output optimal parameters for a synthesizer based on a target sound. For example, a target could be a recording of an instrument such as a trumpet, the program will then attempt to output parameters, created by an Evolutionary Algorithm (EA). These parameters are then fed into a synthesizer which will output solutions. These solutions are then tested to see if this method of sound synthesis is a good solution to manual sound synthesis.

1.1 Motivation for Synthesis

Sound synthesis has been used for a long time to recreate the sound of real musical instruments. In fact, the first form of audio synthesis was done over 100 years ago. It is an incredibly useful tool for musicians, audio producers, and audio researchers. It automates the use of a playing an instrument, meaning the musician doesn't need years of exhaustive practice with an instrument. They can easily create melodies very quickly with the use of synthesizers. Synthesizers have been a staple of popular music from the past 50 years. In theory, any sound can be recreated with a synthesizer meaning musicians no longer need to carry around heavy instruments and can create sounds with instruments they never dreamed of being able to use. This unbridled access to a seemingly infinite number of possibilities has allowed the creativity of artists to flourish. Synthesizers do have their downsides though. They can give off a "synthetic" sound that can be unappealing to the listener. A lot of music programmers do not have a definite vision of what sound they want, they usually go down various paths until they find a sound that they think is good. When working with audio programming, there can be a lot of different parameters, and they do not necessarily respond linearly to being changed. This makes programming instruments or sounds on a synthesizer incredibly difficult, and the programmer needs to have had a lot of knowledge on the subject before attempting it. As well as that, real instruments have a randomness aspect to them. Playing the same instrument twice will give a slightly different result. Synthesizer has problems replicating some of the more complicated, realistic aspects of instrument generation. This project will consider some of these disadvantages and see if genetic algorithms can be applied to solve these problems.

1.2 Motivation for Evolutionary Algorithms

Evolutionary algorithms (EA) are fundamentally based on Darwin's theory of evolution[1]. It works by creating a population of individuals that have completely random parameters. These individuals are then tested against a fitness function that determines how fit they are. The more fit individuals will be more likely to reproduce to create new individuals. These new individuals are put into the population and the process repeats itself. Each repetition of the process is called an evolution. The process can only be stopped when either a fitness threshold is reached or a certain number of evolutions has passed. As instruments create an organic sound, it seems appropriate to try and synthesize new sounds using evolutionary algorithms. Most synthesizers hardcode each instrument manually and will never have a true replicate of the sound. Evolutionary algorithms can be applied generally and from the beginning are never looking for a perfect solution, but one that is "good enough" and is better than current synthesizer techniques.

1.3 Problem Statement

As described above, manual sound synthesis has many downsides, such as not giving off a “realistic” sound and the process of manually programming each individual instrument is extremely time consuming and the programmer needs to have a lot of experience to be able to synthesis good sounds. This project considers other methods of creating sounds, imparticular using evolutionary algorithms, to produce more “realistic” sounds as well as automating the process as much as possible.

1.4 Approach

The first aspect of addressing the problem was structuring the problem into two parts. The first part is the Synthesizer. An efficient, fast synthesizer would be needed as evolutionary algorithms can be computationally expensive, but a synthesizer that could only create very simple sounds would not be useful either as the program is attempting to recreate realistic sounds.

The second part of the problem is the evolutionary algorithm. A good understanding of evolutionary computing was needed to do the research to find what type of evolutionary strategy would be the most optimal for this problem. Evolutionary algorithms solve a problem by creating a population of individuals. These individuals are built with the parameters needed to solve the solution. Initially, each individual has random parameters, the individuals are then tested against a “fitness function” which is the measure of how well the solution is for the problem at hand. In the beginning, these solutions will be terrible, but due to randomness, some will be “better” than others. The best fit individuals in the population and their parameters are combined to create children individuals. The new children are replaced by the population and the testing commences again. This process repeats for however long the user wants it to. Over time, the average fitness of the population should go up, meaning that better and better solutions are found. Eventually, optimal solutions can be found. The process is general and would be suitable for this problem

After getting both a synthesizer and an evolutionary algorithm selected, the next step is creating a program that would combine the two aspects and output a result. After results are collected, they are analysed to see if any conclusions could be made.

1.5 Metrics

In the project, measuring how “good” a sound is probably the most difficult and important aspect of this project. There are two fundamental ways to determine how much the synthesized sound sounds like the target sound. One of these ways is the subjective method, which evolves human testing, where a human participant would rate sounds based on their subjective opinion. The next way of measurement is the objective method, where candidate sounds would be objectively compared with the target sound using mathematical formulas that have already been created to measure the difference between sounds. In this project, finding the best method of measurement is of high importance. Which is better? The subjective method or the objective method? Or could there be a third way of measuring the result using a combination of both the objective and subjective methods? This project will detail the benefits of each method and ultimately find the best option for measurement.

1.6 Project

From

- Build a program that uses evolutionary algorithms to replicate a target sound as best as possible
- Find methods of testing fitness
- Implement these methods then compare and contrast the results

Chapter 2: Related Work

Summary

In this chapter, the related topics of the project will be discussed as well as the technical detail and research that is needed to have a foundation of understanding of the results of this project.

2.1 Sound Synthesis

Sound synthesis is the digital representation of a sound. The synthesizer can be used to replicate instruments or create new sounds [2]. When converting analogue sound to digital sound, a process called sampling must be performed. Sampling takes a measurement (called a sample) of the analogue sound at regular intervals. The *Sampling Rate* is the amount of samples recorder per second. The industry standard sample rate is 44,100Hz. This is fine for human hearing, as the range of human hearing goes from around 20 to 20,000Hz.

There are many ways to synthesize sound, some simple and some complicated. The method used in this project is called "Modulator FM Synthesis (ModFM). It was first developed by Jon Chowning in the late 1960's at Stanford University [3]. It was an extremely popular synthesis method in the 1980s especially because it was the mode of synthesis on the DX7 sound synthesizer that was used heavily in that decade. FM synthesis works by generating a modulator waveform, then altering the shape of that modulator with an envelope and using the altered modulator to vary the frequency of another wave, called a carrier wave. It can be used with low frequency oscillators to make the sound more dynamic. The equation for ModFM synthesis is

$$\cos[\omega_c t + k sin(\omega_m t)] = \sum_{n=-\infty}^{\infty} J_n(k) \cos(\omega_c t + n\omega_m t) \tag{1}$$

As described in [4], where k is the index of modulation, $\omega_c = 2\pi f_c$ and $\omega_m = 2\pi f_m$: f_c and f_c are the carrier and modulator frequencies respectively, Eq (1) shows sets the perceptual quality of FM sounds.

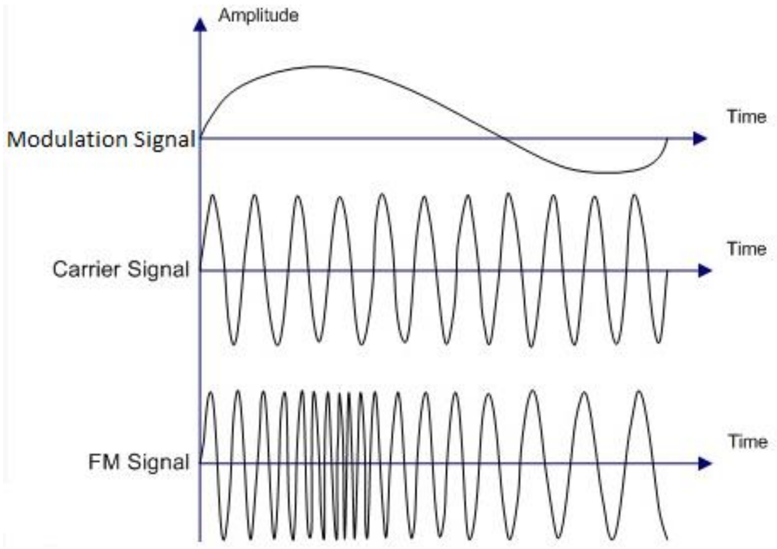


Figure 1. Shows the modulation signal(top), the carrier signal(middle) and how they are combined to create the FM signal(bottom).

2.2 Evolutionary Algorithms

An Evolutionary Algorithm (EA) [5] is an algorithm that is based on Darwin's Theory of evolution. Simply described, given a population of individuals exists in an environment that has limited resources, the individuals will compete for those resources causing survival of the fittest. This will cause the population to get more fit over time. This can be applied to an algorithm by initialising a population of individuals with random attributes. The individuals are then evaluated against a fitness function via genetic operators. The more fit individuals are then selected to be bred for the next generation. This is done through crossbreeding different individuals together or by using a random mutation. As each new population is tested, the average fitness should, in theory, get better.

2.2.1 Differential Evolution

Differential Evolution (DE) is described as "a heuristic approach for minimizing nonlinear and non-differentiable continuous space functions as presented". DE doesn't need to know the specifics of the fitness function, they act like a "black box"[6]. The DE works by first creating random population. It then selects two individuals from the population, calculates a distance vector between these two variables for a parameter and adds this to the parameter value for a third individual in the population. This new solution is called a mutant. Crossover is then processed with the mutant and a randomly selected individual called the original. Then the crossover mutant and the original are evaluated and whichever has a better fitness function is put into the population. The process repeats until all pre-set generations have completed. DE is used for a multitude of reasons. The fact that it represents the parameters are real numbers can be very helpful to the developers as it makes it easier to understand the process. Figure 2 gives a graphical representation of how DE works.

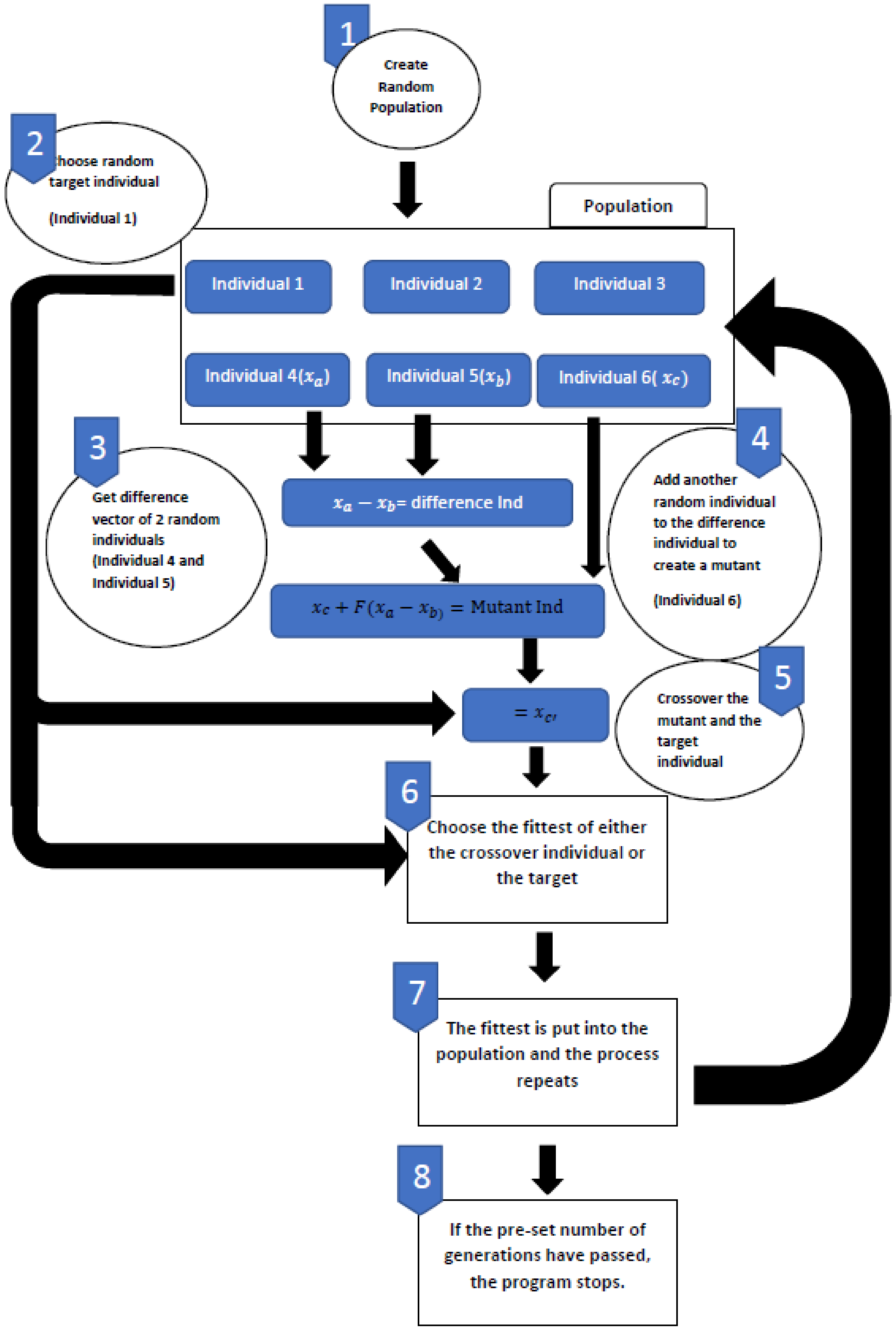

1
Create Random Population
2
Choose random target individual
(Individual 1)
Population
Individual 1
Individual 2
Individual 3
Individual 4(x_a)
Individual 5(x_b)
Individual 6(x_c)
3
Get difference vector of 2 random individuals (Individual 4 and Individual 5)
$x_a - x_b$= difference Ind
4
Add another random individual to the difference individual to create a mutant
(Individual 6)
$x_c + F(x_a - x_{b)} =$ Mutant Ind
$= x_{c'}$
5
Crossover the mutant and the target individual
6
Choose the fittest of either the crossover individual or the target
7
The fittest is put into the population and the process repeats
8
If the pre-set number of generations have passed, the program stops.

Figure 2 shows the process of Differential Evolution using the formula: $x_{c'} = x_c + F(x_a - x_{b)}$ (2)

where $x_{c'}$ is the candidate solution, $x_{c,}x_a$, x_b are the randomly chosen vectors, F is the differential weight.

For a minimization problem, the mutation step size adapts to how a population is configured and will trend toward zero. This can cause genetic drift, where all the individuals in the population will converge to one point. This stunts the DE and it cannot progress from this point.

2.3 Literary review

2.3.1 Work Done in Evolutionary Algorithms

Evolutionary music is the field of study in which this project is categorised in. There has been some work in this field although the field itself is very niche. Using an EA is much better than using a typical heuristic search function because it is limited to producing only a small number of sounds whereas an EA is well suited for the task. The process itself is still not perfect as much of what we know about sound perception is very limited. The idea that the human ear process the difference between sounds as a distance function is supported[7] however, the perception of sound timbre is not widely understood and this can have problems when using an EA to synthesise sounds.

There has been some work done in sound synthesis with EA's. An example is a study that attempts to design sound synthesis algorithms[8]. This is study is slightly different from this project as it attempts to synthesize the algorithms that c synthesize sound while this project attempts to synthesize the sounds themselves. In that study genetic programming was used while this project uses DE's. In that study, they found that evolutionary methods could potentially create algorithms to synthesise sounds. Only a small number of tests were run so the results are not meaningful.

A Genetic algorithm(GA) was used to recreate the sounds of a plucked string[9]. In this project, the fitness function was a comparison of the transformed spectra of the candidate sounds and the target sounds. Another report used a GA [10]with 100 individuals running for 100 generations. They found inconsistent results, but they found that over time the average fitness does increase.

There is another project that has similar aims to this one. They tried to synthesize musical instruments using evolutionary methods. In that project, genetic algorithms were used instead of DE's in this project. There are also different methods of synthetisation used in that project. It concluded with "The original aim, to convincingly synthesize a real musical instrument, has proved to be feasible when applied to the synthesized sound output from the developed synthesis technique" [2]. This project gave inspiration for this one as it's aims were so similar. It also had some problems such as having a lack of range of musical instruments (it could only do instruments at the same pitch). It also was deemed too computationally expensive to be used in real time.

Other experiments looked into using EAs to play musical pieces[11]. Although this is different to the aim of this project, some of the aims were similar. A standard GA was used and they found some good results, the fitness of the generated pieces did improve over time. It also means the idea of crossover and mutation are very different than in GA's

2.3.2 Work done in Differential algorithms

After much research, it was found that there are no other experiments that use DE when trying to synthesize sounds. This is interesting as DE can offer interesting differences to GA's that were used in all the experiments before this. DEs work with actual numbers represented as vectors. This can make it easier to understand for the developer. As there is no research with DEs being used to synthesise sounds, this project considers using them and see how they compare to GA's used in other reports.

2.3.3 Measuring Fitness

In most of the work done in this field, the measurement of fitness is usually done by objectively measuring the sound using a mathematical function on the spectra of the sounds. Then the distance between the target sound and the candidate sound are calculated and this distance value is used to calculate the fitness. In one of the experiments to synthesise sounds using GA's experiment, 4 different fitness methods and tested to see which one had the best results [9]. These 4 distance measurements were Attributive distance, pointwise distance, DFT distance and composite distance. It was fond that DFT distance gave the best results. DFT means Discrete Fourier Transformation and outputs a value from the spectra of the sounds. This value can then be used to measure the distance between 2 sounds. The only problem with using a DFT is that it can be slow to compute and there are other methods that give the same result the DFT gives but are much quicker computationally. One of these methods is called Fast Fourier Transformation (FFT). FFT is an efficient implementation of DFT [12] . Since it is much less computationally expensive and gives the same results as DFT, it was chosen to be the objective measurement of fitness in this project

There is also another method of measurement called the subjective method. This method relies on human engagement rather than a mathematical function. The users' judgement acts as the fitness function and the user decides, subjectively if an individual in the population is fit or not. A report was done that shown a system can generate musical phrases successfully by combining GA's with human judgement [13].

This project looks at both an objective measurement of fitness as well as a subjective method of fitness. It compares the two methods and ultimately discovers which if any is more suited to synthesising replica sounds.

Chapter 3: Proposed Approach

Summary

This chapter discusses how the problem was analysed and approached. The objective of this solution is to recreate a target sound as best as possible. But how can a "sound" be measured? In this project, two ways of solving the problem are discussed. The first is a subjective method that uses human judgement to test the fitness of sounds. The other is the objective method that uses mathematical functions to objectively judge the sounds.

3.1 Problem Analysis

The problem with sound synthesis is that it can be extremely time consuming and difficult to manually program each instrument. For example, an audio programmer would have to constantly listen to the target sound (e.g. guitar) and then manually tweak the parameters until they get a sound that is somewhat like the target sound. There must be a better solution that will remove much of the human interaction. As EA are general, they can be applied to this problem. As well as that there can be many solutions to the problem and can simultaneously handle multiple, potential solutions in the so-called population. In this project, two different variants are created. One that uses the subjective method to measure fitness and the other that uses the objective method. The two variants are almost identical, except for user interface components in the subjective version and the fitness function that is only used in the objective method. Both programs use a simple ModFM synthesizer to create the sounds. The synthesizer uses four parameters.

- Carrier frequency: The frequency of the carrier wave that creates the sound

- Modulation frequency: The frequency that will modulate the carrier frequency
- Modulator index: Describes how much the modulated variable varies around its unmodulated level
- Sample Rate: Describes how many samples are recorded per second

The output of this will be a synthesized sound. The sample rate decided is 44,100 and will always be a static variable. This is because as discussed before in chapter 2, 44,100 is the industry standard and is more than enough to cover the range of possible sounds that humans can hear. That leaves three parameters that can be any number within their ranges. Carrier frequency has a range [2, 20000] as does the modulation frequency. This range is the same range as human hearing. Mod Index has a range of 100.

In both methods, the sound to be replicated is that of a trumpet. Many different instruments were tested, but the trumpet seemed to give the best results. This could be because the trumpet gives a low, bass sound that is noisy and easier to replicate than other sounds which have little noise and are much more difficult to replicate. Other brass instruments such as the tuba and French horn would also be viable. Having one instrument makes it easier to compare testing methods and results.

In the DE, a population of size *N* is created. Each individual will then have the three parameters stated. At first, the population will be randomised. At this point, the program for each different measurement method will differ.

3.2 Subjective method

The subjective method relies heavily on human interaction. The program starts by initializing a population of random individuals. The population is then iterated, and the user is asked to judge 2 sounds, the individual on a scale from 0-5(with 0 being unrecognisable to the target sound and 5 being an exact replica of the target sound) and a mutant sound which is created by the DE. Whichever individual that has the highest rating is put back into the population. The user continues this process until all the sounds in the population have been rated. The process then repeats for how many generations were set. An example would be if there were 4 individuals and 10 generations. This would mean the user would have to rate 80 sounds.

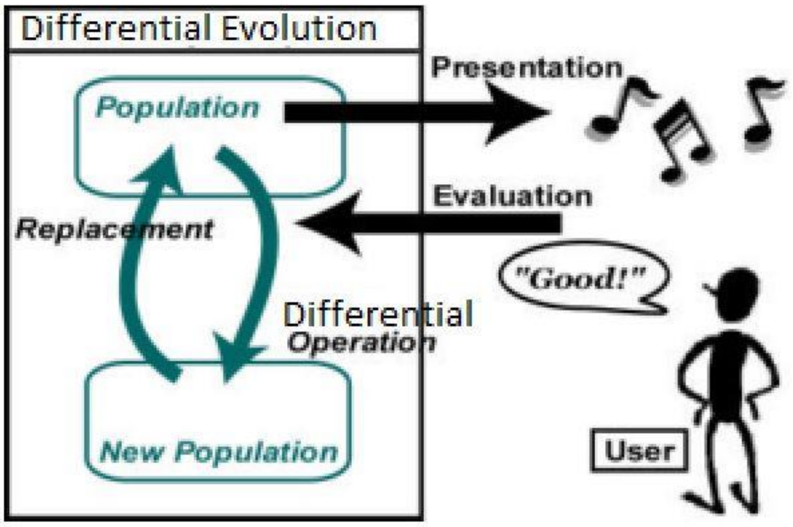


Figure 3. A simple graphical summary of how the users' judgement acts as the fitness function

Since the projects aim is to replicate sounds that will be ultimately judged by human ears, it can be argued that the subjective method is superior since it will be humans judging the performance of the

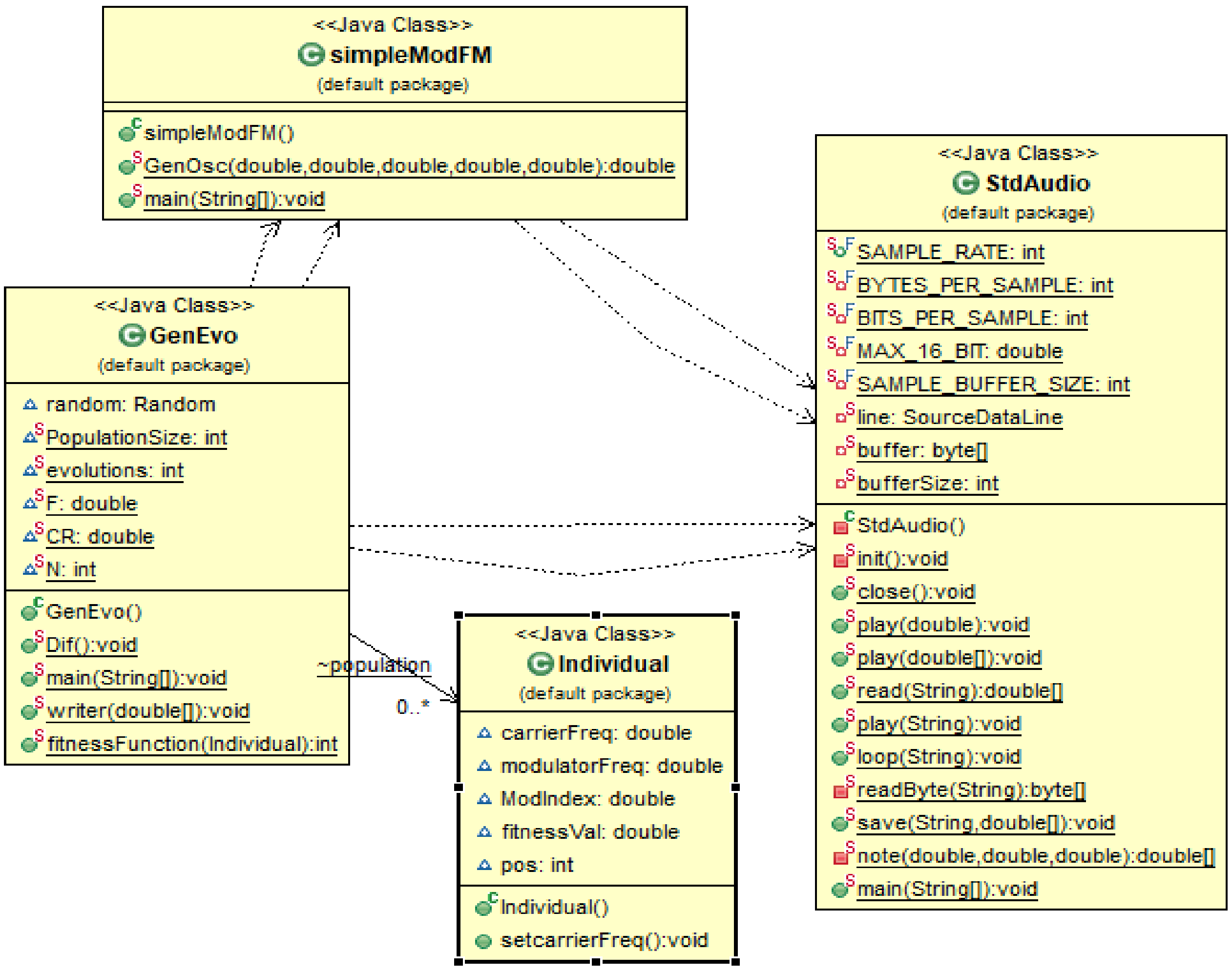
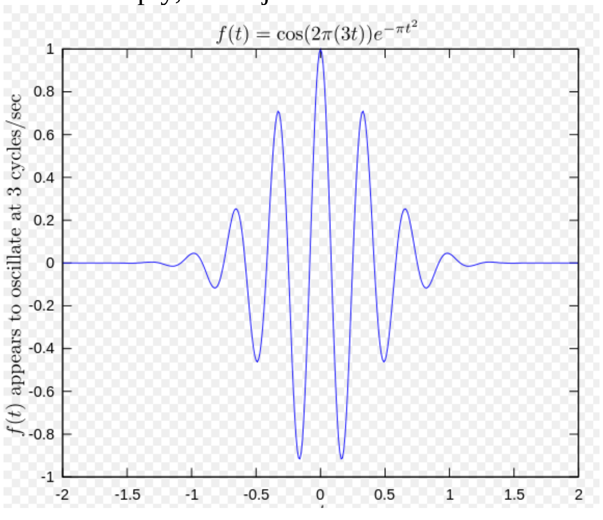


Figure 4.The UML diagram for the subjective program. GenEvo is the main class that Includes fitnessFuncton() which processes the fitness value of each sound. Individual contains the parameters of each sound. simpleModFM takes in the parameters from GenEvo and outputs a double array that can be used with StdAudio to play or save sounds

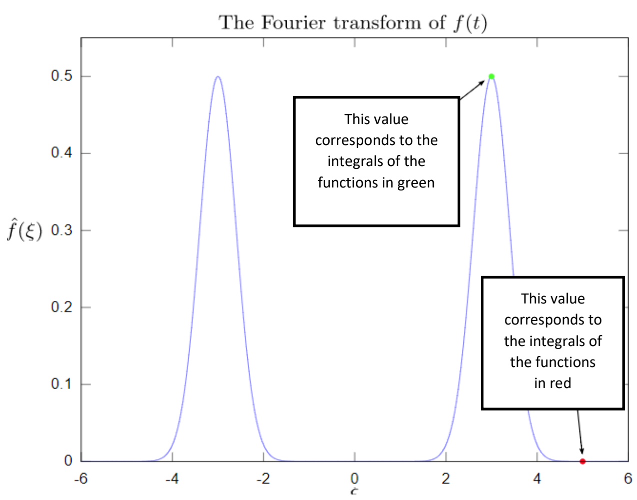
algorithm at each step. This method also has huge downsides, the main one being the length of time it takes to complete a simulation. Using human's judgement as a device in the algorithm makes the procedure extremely slow, and only a small amount of generations can be achieved as there is just not enough time for a person to sift through hundreds of generations. The method also suffers from human bias, a person can hear two similar sounds, or even the same sound twice and yet judges them differently. This must be considered when judging how to measure results.

3.3 Objective Method

The objective method of measurement is much like how most evolutionary methods measure fitness. Put simply, the objective method is the fitness function of the algorithm. It evaluates each individual



(a)Initial sound



(b) sound after FFT is completed

Figure 5. The initial sound (a). The FFT process is then applied and the result (b) shows the individual waves that combined to make the sound

in the population each iteration, as well as evaluating new candidate individuals. Measuring an objective value for a sound can be difficult. A method for doing this is called is called Fourier transformation. A Fourier transformation splits the audio signal into the frequencies that make it up. In the program, a Fourier transformation is computed for both the target audio to be replicated and the synthesized audio created from the DE, the two Fourier transformations are then compared against each other and the distance between them is calculated. This distance is the fitness function. The closer the distance, more fit the sound is and in theory the more the sound should sound like the target sound.

One of the benefits of the objective method is that the population size can be much larger than the subjective method and many more generations can be simulated. This is because there is no need for human input, the algorithms do everything. Theoretically, this should mean that the objective method will find a better solution than the objective method. But this may not be entirely true. While a sound might have a good “fitness value”, this does not necessarily correlate with a sound that a human would recognise to be of good fitness. This is because the human brain does not measure sound in a mathematically objectively way like the Fourier transformation does.

The objective method uses the DE described in chapter 2. As the fitness function in the DE is a distance measurement, this means to smaller the distance, the more fit the sound is. This method is minimizing for an optimal solution.

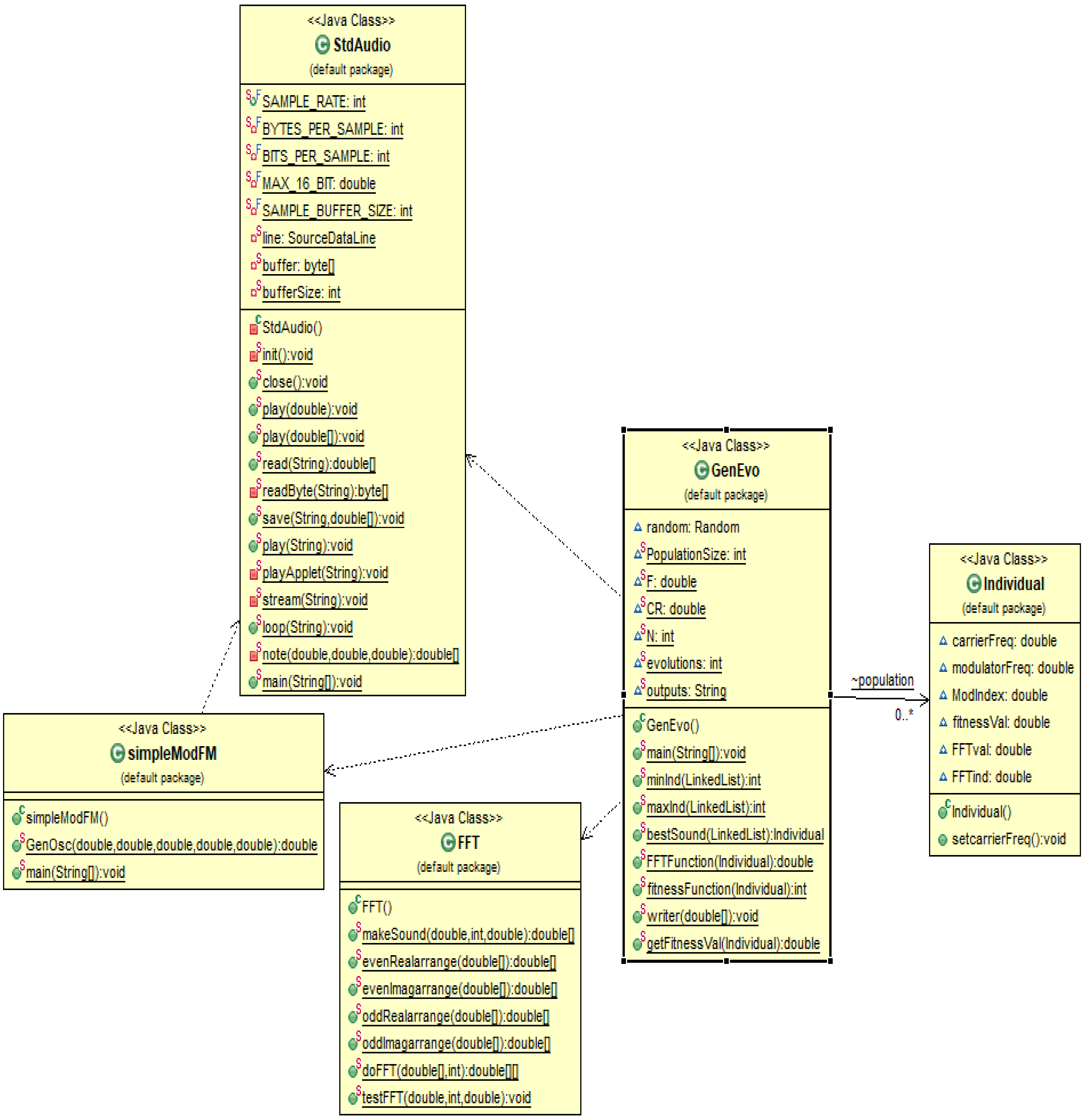


Figure 6. The UML of the objective program. GenEvo is the main class that executes the program. Individual contains the parameters of each sound. simpleModFM takes in the parameters from GenEvo and outputs a double array that can be used with StdAudio to play or save sounds. FFT is the Fourier Transform method that builds an fourier transform of the sounds and compares them

.

3.4 Differential Evolution in the Subjective Method

The method of differential evolution in the subjective differs slightly from the method described in chapter 2. This is due to necessity as the fitness function is the input from the user and a different DE method was needed because the standard method is too inefficient and slow for user interaction.

The user will choose a target sound to reproduce. It is suggested that they listen to this sound before they run the program so it is fresh in their mind for testing. After the initial population is randomised, the user will be asked to rate each of the initial sounds on a scale from 0-5. 0 meaning the sound sounds nothing like the target and 5 being an almost exact replication. The scale is not objective and each user will have a slightly different opinion on what rating to give each sound. After this initial phase is done, each individual in the population will be played and the user will rate, a mutant sound will also be played and the user rates this too. If the mutant sounds have a higher score it will be put into the population and the original sound from the population will be removed. This process repeats until all the individuals in the population are tested. This represents one population. The process repeats for how many generations the user has set.

As the DE in the subjective method is based on a rating scale, the higher the rating, the higher the fitness value. This method is maximising for an optimal solution.

Chapter 4: Experimental Approach

Summary: This chapter discusses different ways that the problem could have been solved and why the proposed approach was chosen over the other methods.

4.1 Creating the Synthesizer

After choosing to build a modFM synthesizer the next decision was whether to include an ADSR envelope or not. ADSR stands for Attack Decay Sustain Release and can provide a much more dynamic range to the outputted sound. It works by changing the amplitude of the signal over time. It is based on how real instruments work when played. For example, when a real instrument is played, for example, a trumpet, the sound takes a certain amount of time to reach its maximum amplitude because air must be blown into the instrument. The decay then decreases the amplitude until a sustained level is reached where the sound plays at the same amplitude until the release is hit. The release then slowly declines until the sound is complete.

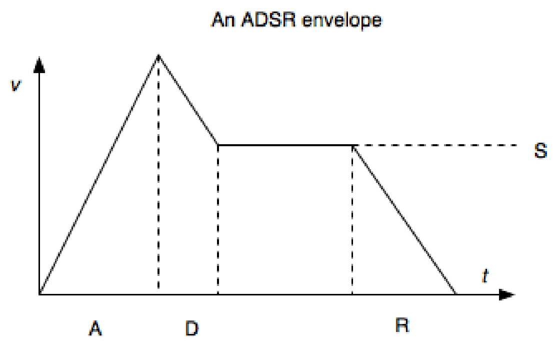
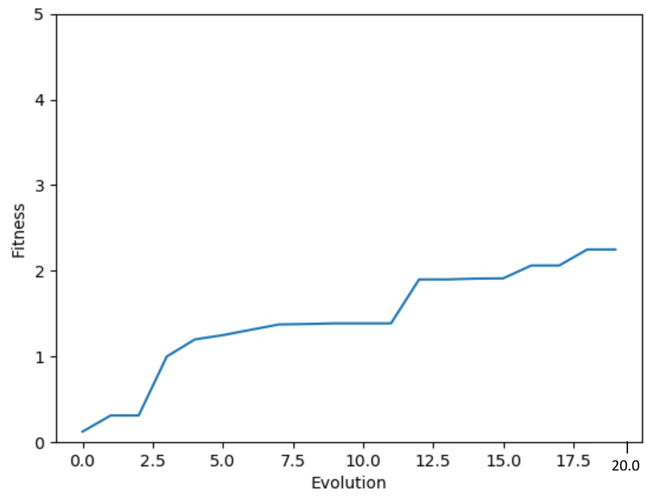


Figure 6. A visual representation of the ADSR, where *A* is attack*, D* is decay, *S* is sustain time, *R* is release (v is amplitude on the y-axis, and t is time on the x-axis)

Adding an ADSR to the program would increase the potential of getting realistic sounds which would give better results. But there are too many problems with adding an ADSR. The first is an only minor problem, but adding an ADSR means adding more parameters meaning the program will run slower. This alone is not too much of a problem but getting large data sets would take much longer using an ADSR. With the subjective method, the ADSR added to many parameters for any meaningful results to be made. In the testing, the maximum generations that were run were 20. This was chosen because it seemed to give some meaningful results with the simple version but didn't take an enormous


(a) Without ADSR

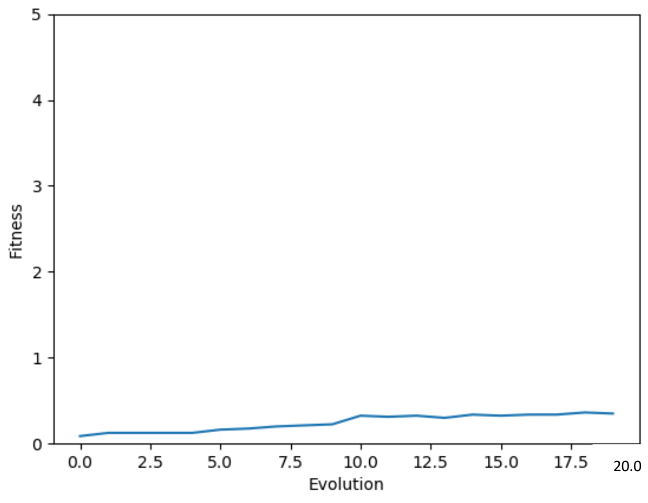

(b) With ADSR

Figure 7. Both simulations are run 10 times. Both have 20 evolutions and 4 individuals in the population. (a) shows the simulation that does not use an ADSR. While (b) does use an ADSR. The subjective method maximises for an optimal solution.

(a) Without ADSR (b) With ADSR

Both are increasing over time, but the ADSR program increases very slowly while with the program without an ADSR clear growth can be seen. It was decided to not use the ADSR synthesizer for the subjective method as it would take too long to get meaningful results.

Table 2. Summary of the parameters used for testing the subjective method

Parameter	*Value*
Population Size	4
Generations	10
Tests Run	10

Table 3. Parameters of the synthesizer with no ADSR

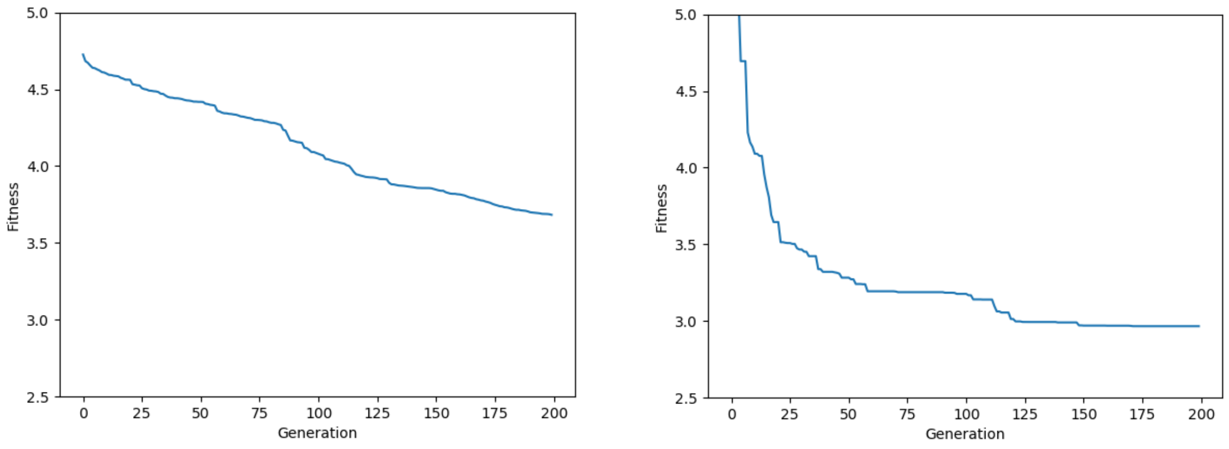
Parameter	Value
Mod Index	Random value
Carrier Frequency	Random value [20,20000]
Modulator Frequency	Random value [20,20000]

Table 4. Parameters of the synthesizer with an ADSR

Parameter	Value
Mod Index	Random value
Carrier Frequency	Random value [20,20000]
Modulator Frequency	Random value [20,20000]
Attack	Random value [0,1]
Decay	Random value [0,1]
Sustain	Random value [0,1]
Release	Random value [0,1]

4.2 ADSR in the Objective Method

As the objective method can process many more generations, the ADSR should be viable on it. Testing was done to compare what difference the ADSR makes to improved fitness. Comparing between the subjective and objective methods would be unfair as the objective method would run with a different synthesizer even though the ADSR synthesizer produces better results. It is



(a) Without ADSR (b) With ADSR

Figure 8. 2 simulations, 10 runs each. Both with a population of 10 and run for 200 generations. (a) uses no ADSR while (b) uses an ADSR. The objective method minimises for an optimal solution.

for this reason, the simple non-ADSR synthesizer is chosen for both the objective and subjective methods. The objective methods fitness is determined by the distance between the sound and the sound that it is trying to replicate. The lower the distance, the better the fitness. Therefore, in the solution is being minimized for, while the subjective method maximises.

for this reason, the simple non-ADSR synthesizer is chosen for both the objective and subjective methods

Chapter 5: Results and Discussion

Summary

This chapter will describe the results of the tests that were run as well as analysing the results and comparisons between the different methods

5.1 Subjective Method

The tests run for the subjective method were simple. A user was selected, and then when prompted the first sound will play. The user rates these sounds on a scale of 1-5. The user will also rate “candidate” sounds, the sounds that were mutated from the DE. Whichever sounds have a better fitness score are put back into the population and the process repeats.

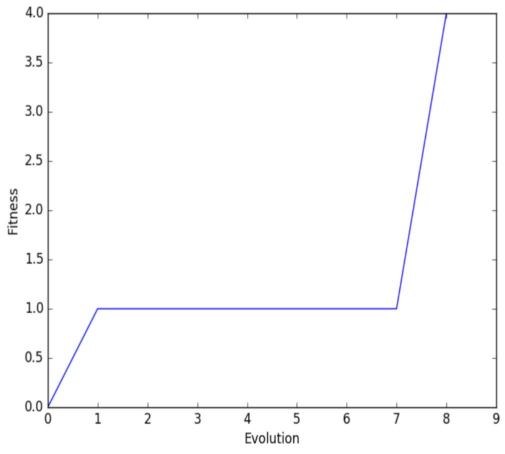


Figure 9. This is a sample test run, it shows the fitness maximising towards a solution over time. It is run once, with 4 individuals over 10 generations

Many different simulations of the subjective method were run. The problem with this method is the fact that human interaction slows the process down. This meant that large simulations were not feasible

10 users did a simulation each, these simulations were then averaged out to give a good representation of the results. For these tests, a sound of a trumpet was chosen to be the target sound. The sound lasts 1.5 seconds. The users rated all the sounds from 1-5 subjectively. All the simulations had 4 individuals and were run over 20 generations. 80 original sounds were played along with the 80 candidate sounds meaning that the user was required to input a value 160 times, from this relatively small evaluation it is clear why the subjective method is not feasible for large simulations. From the simulations that were run, the average fitness does increase over time. The figures only show the average of all the results in each generation. Many of the sounds did achieve ratings of up to four. On such a small scale, using a simple synthesizer it is impressive that results as good as these were shown

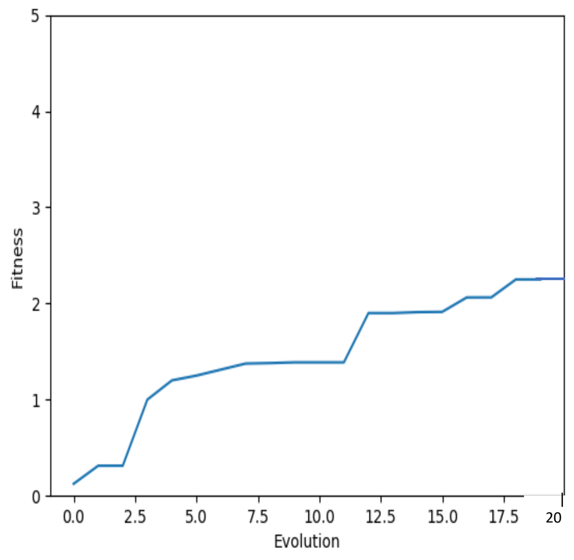


Fig 10. The average of 10 simulations (20 generations,
4 individuals

5.2 Objective Method

Objective method results with the objective method, many more simulations were run because it required little human interaction had a much larger scope than the subjective methods. The fitness recorded is the average of the distance between the candidate sounds and the target sounds. The closer to 0 you get, the more fit the population is. Over time the fitness does grow to show that the algorithm

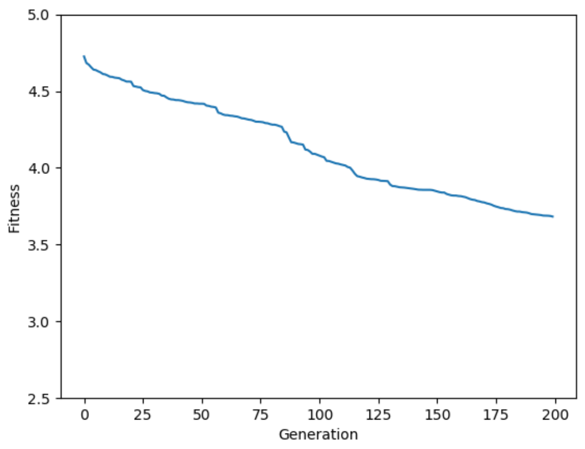


Figure 11. This is the average of 10 simulations run over 200 generations each with 10 individuals in each population. The objective method is minimizing for an optimal solution.

does work. From listening to the best sounds created, the results are very inconsistent. Some sounds sound very good subjectively, but others are incomprehensible. Even though a sound may have a good fitness value it doesn't mean it will sound good.

Chapter 6: Conclusions

6.1 Subjective Method

The results of the subjective method show that fitness does improve over generations. The sounds it created in many of the simulations were accurate according to the ratings that the user gave the sounds. The results only show the average of all individuals in the population. In many cases, the fit candidate had a fitness level of between three and five. For such a small scope, the results were very good. It is unfortunate that this method is so difficult to test with large scaled simulations. The results seemed to show that the program seems to show that the individuals did seem to converge after about 10 generations. This could be due to the limited nature of the modulator. Another problem with this method is that humans can be biased and the results will never be quantifiable. This is a problem when trying to compare this method with the objective method.

6.2 Objective Method

With the objective method, many more simulations were run because it required little human interaction had a much larger scope than the subjective methods. The fitness recorded is the average of the distance between the candidate sounds and the target sounds. The closer to 0 you get, the more fit the population is. From the simulations performed the fitness increases over time, but after reviewing the sound that it created subjectively, even the good fitness sounds were very inconsistent. They didn't sound like a replica of the original sound even though they had good scores. The objective method can't factor in how a human will judge the sounds and ultimately the project aims to create sounds that a human would call good, not the algorithm

6.3 Comparison of the two methods

It is incredibly difficult to compare the two methods. The objective method is quantifiable while the subjective methods results are obviously subjective and do not stand up to objective scrutiny. The scales are also different and there is no way to translate the results of one to the other. The subjective maximises for the solution while the objective minimizes for the solution

The results of the subjective method do give more consistent, more realistic sounds overall based on the user scores. Again, this is because the method is subjective, so it makes sense that the results will sound better to a subjective human. The objective methods result had a larger scope, but were very inconsistent. From the experimental approaches, an ADSR gave the objective method better scores. Comparing between the methods would not be meaningful as that would mean two different synthesizers would have been used. It would have been nearly impossible to use the ADSR with the subjective method because the time taken to test would have been much too long. The subjective method did much better at recreating the target overall although it wasn't perfect it was much better than the sounds the objective method created. If more time was available, trying to find and test more synthasisers would have been useful. Using the ModFM synthasiser on it's own was definitely limitng.

6.4 Final thoughts

The original problem was not only to recreate sounds using evolutionary algorithms but to see if the algorithms could be more efficient than manual programming. While the subjective method does give good results, the effort required in testing it is very large. It could be argued that the time spent trying to use the subjective method to collect sounds would be better spent manually programming a synthesizer. The objective methods results were too inconsistent and wouldn't be useful, even though they take very little effort and time to create. A better synthesizer would be useful. Due to time constraints, I was unable to build a more dynamic, realistic sound synthesizer. A better synthesizer would have a much larger scope and should, therefore, get more accurate results. More testing with the subjective method would be useful as well. There is not enough data on large generation sets. I ran out of time to develop an algorithm to work with the ADSR A. better algorithm that can incorporate the ADSR for the subjective method would be very helpful. The algorithms for the objective method could be improved. There are many ways of tinkering with the DE to make the solution more optima.

When comparing with other experiments in the field of evolutionary music, the results seem to be similar. Most other experiments show inconsistent results, with fitness increasing in general. The biggest difference in this project was the subjective method, which was only used in a very small amount of other experiments. The subjective results were impressive and show that for evolutionary music experiments, it could be argued that subjective measurement could be superior to objective measurement.

References

[1] C. Darwin, "The Origin of Species." John Murray, 1859.

[2] C. Eade, "Musical Instrument Synthesis Using Genetic Algorithms," 2003.

[3] J. Timoney and T. Lysaght, "Introduction to Sound Synthesis," 2017.

[4] V. Lazzarini and J. Timoney, "Theory and practice of modified frequency modulation synthesis," *AES J. Audio Eng. Soc.*, vol. 58, no. 6, pp. 459–471, 2010.

[5] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[6] E. Galván-López, T. Curran, J. McDermott, and P. Carroll, "Design of an autonomous intelligent Demand-Side Management system using stochastic optimisation evolutionary algorithms," *Neurocomputing An Int. J.*, vol. 170, pp. 270–285, 2015.

[7] M. J. Ryan and a S. Rand, "The Royal Society is collaborating with JSTOR to digitize, preserve, and extend access to Philosophical Transactions: Biological Sciences. ® www.jstor.org," *Society*, vol. 351, no. 1278, pp. 187–195, 1996.

[8] R. A. Garcia, "Growing Sound Synthesizers using Evolutionary Methods," *Synthesis (Stuttg).*, vol. M, no. Garcia 2000, pp. 99–107, 2001.

[9] J. Riionheimo and V. Valimaki, "Parameter Estimation of a Plucked String Synthesis Model with Genetic Algorithm," *Icmc 2002*, pp. 791–805, 2002.

[10] S. Editors, R. Th, and E. J. N. K. H. P. Spaink, "The Art of Artificial Evolution," Springers, pp. 81–100.

[11] E. R. Miranda, *Evolutionary Computer Music*. Springer.

[12] K. R. Rao, D. N. Kim, and J.-J. Hwang, "Fast Fourier Transform - Algorithms and Applications," 2010, p. 1.

[13] N. Tokui and H. Iba, "Music composition with interactive evolutionary computation,"

Proc. Third Int. Conf. Gener. Art, pp. 215–226, 2000.