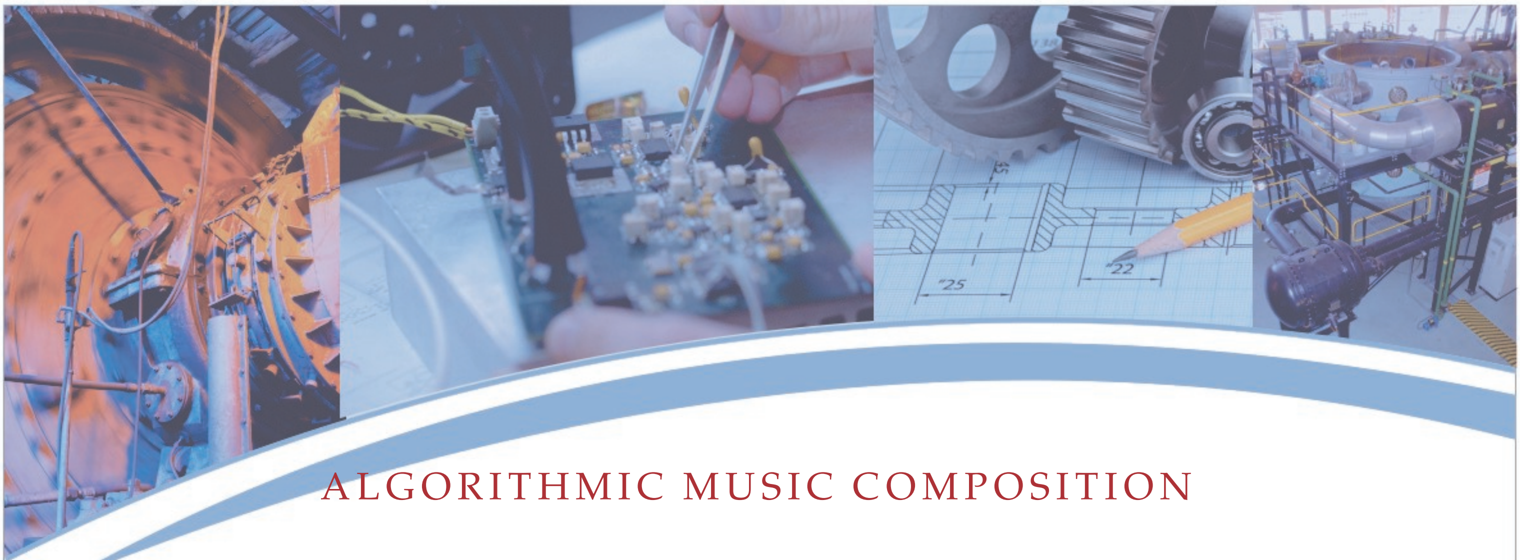

ALGORITHMIC MUSIC COMPOSITION

by:

S. Jacholke 23671750

submitted in persuit of the degree

BACHELOR OF ENGINEERING
in
COMPUTER AND ELECTRONIC ENGINEERING

North-West University Potchefstroom Campus

Supervisor: Mr. A.J. Alberts
Potchefstroom
June 2, 2015
version 2.2

It all starts here

NORTH-WEST UNIVERSITY ®
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
POTCHEFSTROOM CAMPUS

ABSTRACT

Obtaining the required music licensing may not be viable for certain individuals and business owners. The alternative to compose and produce music self requires time and skill.

For this project we investigate another alternative, to produce music using computer algorithms. The field of algorithmic music composition has received a lot of attention.

We investigate the relevent research, implement the required algorithms and develop a coherent system that produces simple monophonic melodies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

ANN Artificial Neural Network

GA Genetic Algorithm

MIDI Musical Instrument Digital Interface

NCD Normalized Compression Distance

NID Normalized Information Distance

IGA Interactive Genetic Algorithm

SRN Simple Recurrent Network

LTSM Long term short term

ART Adapative resonance theory

RNN Recurrent neural network

Part I

INTRODUCTION

1 ALGORITHMIC MUSIC COMPOSITION

1.1 INTRODUCTION

Music is the art form of sound. Certain characteristics and patterns can be identified that belong to certain types of music.

There has been a lot of work done in computer generated music and computer assisted music composition. Recent research into music composition by means of evolutionary genetic algorithms and other machine learning algorithms such as neural networks have met some success.

Music generated algorithmically by computers might some day share the success known by modern and historic music.

Computer generated music is an old idea, however existing solutions that generate music use outdated procedural algorithms and do not match the quality of proper historic music. The variety and quality of existing solutions is limited. Most current solutions were implemented to accommodate the research being done and are not suitable for end-users.

This section will outline the problem that is faced with conventional music playback and propose a solution to the problem. Some of the ideas behind systems that learn music composition will be outlined and described. The problem of implementing and creating a solution will be detailed as well as the steps required to complete such a project.

1.2 PROBLEM

In order to avoid copyright infringement music has to be licensed for use in commercial applications. Small business owners, independent developers and other smaller organizations may not have the required funding in order to obtain the relevant licensing.

The alternative solution is to produce the required music self, however this requires time and skill.

A solution to this problem is to generate music by artificial intelligent means. Algorithmic music composition is of theoretical interest and research has been done for composing music by genetic algorithms and neural networks.

A software application that is able to compose and play monophonic melodies according to a certain style by means of machine learning algorithms could solve the licensing and resource problem.

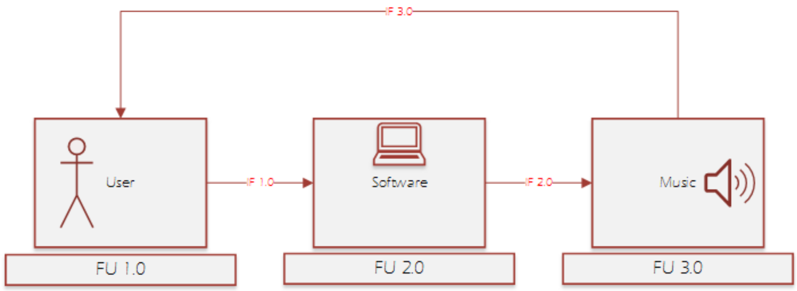

Figure 1: Figure of high-level architecture of project

1.3 SOLUTION

The solution to the problem is to create an application that is capable of composing and playing simple monophonic melodies using machine learning algorithms.

An application is required that has the following features :

1. Be able to compose and play monophonic melodies
2. Melodies are to be composed using machine learning algorithms
3. Melodies should be composed according to a certain style
4. The user should be able to select the style
5. The application will feature a MIDI library which is to be used as training data for the machine learning algorithms

Figure 1 indicates the interaction between the user and the software.

Since the quality and aesthetics of a musical melody is subjective, it is beyond the scope of the application to ensure that each melody produced is objectively good[1]. Composed musical pieces will take the form of simple melodies.

1.4 ALTERNATIVE SOLUTIONS

A wide variety of research has been done and in this section we will briefly look at some of the existing solutions and their drawbacks

1.4.0.1 *NEUROGEN*

NEUROGEN is a system that was designed to compose small diatonic, western-type, four part harmony compositions based on a training set of example musical fragments. NEUROGEN tries to produce coherent music of the type that is typically found in traditionally hymns [2]. However NEUROGEN is only tries to fulfill its main research goals.

1 The focus is rather on the generated musical pieces not being objectively bad

1.4.0.2 *CONCERT*

CONCERT incorporates psychologically-grounded representations of pitch, duration and harmonic structure in order to compose novel melodies. CONCERT uses a neural network in order to do note by note predictions.

CONCERT struggles to compose natural music as the music produced was not globally coherent [3].

CONCERT also only tries to fulfill its research goals, that is, to compose music on a note by note basis usign a neural network.

More recent studies into recurrent neural networks based on Long term short term (LTSM) indicate that it is indeed possible to compose music using recurrent neural networks that have global structure [1].

1.4.0.3 *GenJam*

GenJam - Genetic Jammer is a IGA that learns to improvise jazz [4]. The system is able to:

1. Play full-chorus improvised solos
2. Respond interactively when the trumpet is played to trade fours or eights
3. Perform a smart echo of improvisation

The main drawback of GenJam is that it is an interactive algorithm and as such requires input from the user. The creates a performance bottleneck as it is time consuming for the user to rate musical pieces. See section 5.6 for work on IGAs

1.4.0.4 *Other*

Most of the work done in the field has been aimed on narrow application of the ideas being investigated. Thus algorithms and applications constructed were aimed only to experimentally study the ideas being investigated and not to serve as a end-user product.

Some other small application exist which employ Interactive Genetic Algorithms (IGAs) such as:

1. Evolutune[2]
2. Song Builder[3]
3. DarwinTunes[4]

Even though more recent research into algorithmic music composition yields promising results there is no single application combining the more recent research into a coherent application that is able to play monophonic melodies.

2 http://askory.phratry.net/projects/evolutune/
3 http://www.compose-music.com/
4 http://game.darwintunes.org/

1.5 FEASIBILITY

There has been a plethora of research done into algorithmic music composition. Some of the more recent research that utilizes machine learning algorithms is investigated in chapter 3

Some machine learning methods to compose music algorithmically include:

- Recurrent neural networks [3, 1, 5]
- Genetic algorithms using NCD as fitness functions [6, 7]
- Genetic algorithms using neural networks as fitness functions [2, 8, 9]
- Genetic algorithms employing Zipf's law and cosine similarity [10, 11, 7]
- Interactive genetic algorithms [8, 4, 12, 13, 14]

A variety of ideas and techniques in algorithmic music composition have already been researched. The difficulty arises in the implementation of the ideas (as the process is only fundamentally documented), representation of music in data structures and the implementation of the relevant machine learning algorithms.

1.6 METHODOLOGY

In Royce's original waterfall model the following steps are followed when developing software:

- Requirements specification
- Design resulting in the software architecture
- Construction
- Integration
- Testing and debugging
- Installation
- Maintenance

Since there are a variety of algorithmic music composition strategies, and no formal analysis that compares the quality of the resulting music of each strategy (difficult as it is a subjective matter) the software prototyping methodology could be used to test these different ideas and to construct a trade-off study to determine the most feasible techniques.

Software prototyping is the development of prototypes - incomplete versions of the software. Some basic principles of software prototyping include:

- Prototype selected parts
- Breaking project into smaller segments
- User or client may be involved

- Iterative modification to meet user demands

The main ideas of the systems engineering process, as applicable to software will be followed. Ideas from the waterfall model and software prototyping will be incorporated.

The basic formulation of the design of the project is as follows:

1. Conceptual design
    - Identify problem
    - Identify requirements
    - Resource allocation
    - Literature study
    - Feasibility analysis
    - Trade off study
2. Preliminary design
3. Detail design
    - Design of software architecture
    - Prototyping of different strategies
4. Construction
    - Code is written
    - Code is tested
    - Iteratively improve until project meets specifications
5. Phase out, support, maintenance

In order to identify suitable algorithms a literature review will be done. The various algorithms will be prototyped and tested and the best few resulting algorithms will be used. Software development will follow the waterfall model.

1.7 CONCLUSION

Since the act of generating music using algorithmic means is of theoretical interest there is a plethora of research available, however most of these implementations only aim to explore the main idea being researched. Thus there is a lack of accessible, user-friendly applications that are able to generate music algorithmically.

The aim of this project is to construct an application that is able to compose music algorithmically. The application will utilize machine learning algorithms in order to compose monophonic melodies according to the style of a certain author or theme. These ideas have already been researched and thus the application is feasible. The only cost is time and effort.

The methodology for successfully completing the project is outlined and a schedule for completing each goal is presented. This should assist in meeting the deadlines and completing the project successfully according to the specifications.

Part II

LITERATURE REVIEW

2 MUSIC AND MUSIC-REPRESENTATION

2.1 MUSICAL ELEMENTS

Music is an art form for which the medium is sound. The common elements in music are:

- Pitch - Subjective sensation reflecting lowness and highness of sound, also represented more objectively as frequency.
- Rhythm - Arrangement of sounds and silences.
- Dynamics - Execution of a given piece (speed, volume)
- Timbre - Tone

Music composition refers to the creation and recording of music through a medium that others can interpret. Music can be composed for repeated playback or it can be composed on the spot.

A melody is a set of notes (or rests) that are performed in series. Each note may have a different length and different stress. These notes are arranged in a certain rhythmic pattern.

Notes were traditionally given a letter to represent the pitch of the note. The names come from the set $\{A, B, C, D, E, F\}$. Notes may have their pitch modified by additional symbols such as a sharp(#).

Notes and rests may have different lengths.

Li and Sleep have found that a given piece of music remains recognizable when the length of the notes are randomized [15].

2.2 MIDI

Music can be stored digitally in a variety of formats and encodings. This allows repeated performance of the same track and also eases distribution of the music.

Musical Instrument Digital Interface (MIDI) is a standard which describes the protocol, digital interface and connectors that allows electronic instruments to communicate with one another.

The MIDI 1.0 detailed specification fully describes the MIDI interface [16]

The MIDI file format describes the way MIDI information is stored in a file. Each MIDI file starts with a header chunk that describes the time division and the number of tracks. After the header chunk multiple track chunks occur. Each track contains multiple MIDI events.

The header chunk is described as follows:

```
"MThd" + [header_length] + [format] + [n] + [division]
```

where

[header_length] always 6 bytes

[format] 0 - single track format, 1 multiple track format, 2 multiple song format

[n] number of track chunks

[division] unit of time for delta timing

After the header chunk [n] track chunks follow. Each track chunk is composed as follows:

```
"MTrk" + [length] + [track_event] <+ [track_event1] + [track_event2] +
    [...] + [track_eventn]>
```

where

[length] number of bytes in track chunk

[track_event] sequenced track event, described next

The track chunk can be seen to contain multiple track events. Each track event consists a delta time and either a midi event, meta event or sysex_event:

```
[v_time] + [midi_event] | [meta_event] | <sysex_event>
```

The meta-event contains additional information such as text which can be displayed, instrument names, lyrics and so on.

Table 1: Table containing some midi events

Command	**Meaning**	**parameters**	**param 1**	**param 2**
0x80	Note-off	2	key	velocity
0x90	Note-on	2	key	velocity
0xA0	Aftertouch	2	key	touch
0xB0	Continuous controller	2	controller #	controller value
0xC0	Patch change	2	instrument #	
0xD0	Channel Pressure	1	pressure	

Table 1 shows some of the possible midi events. Each event has a command and additional arguments.

The full midi specification can be ordered online at midi.org[1]

2.3 REPRESENTATION

Representation of music is important to achieve successful generation of a correct solution [2]. However the search space of machine learning algorithms may be too large if limitations are not introduced [17].

1 http://www.midi.org/

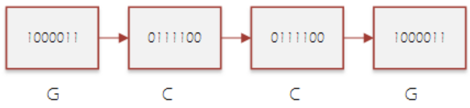

Figure 2: Simple melody represented as a bit string

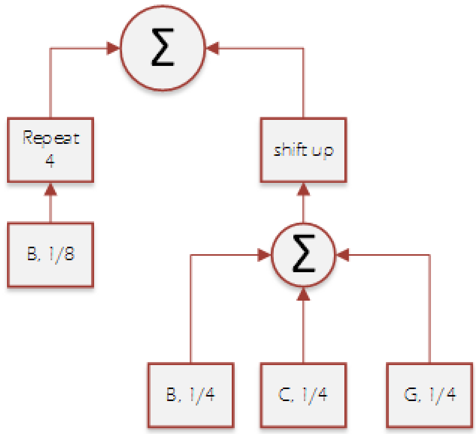

Figure 3: Simple melody represented in tree form

Figure 4: Training chords used in [1]

The musical data from musical pieces need to have a proper representation in order to be used by a machine learning algorithm.

In [1] Eck a form of 12-bar blues was used with 8 notes per bar. He used the following representation for the LTSM system:

1. 12-bar musical pieces were used
2. 8 notes per bar were used
3. The same chords were used between songs, see figure 4
4. The melody notes were built using the pentatonic scale
5. The neural network inputs indicate whether a certain note is on or off

A melody would be represented differently in a linear genetic algorithm. A simple melody consisting only of a few notes could be represented as seen in figure 2. Note the duration of a note is not defined and could be consider fixed.

However it might be preferably to encode more events such rests, chords, dynamics, note duration and so on.

Tree based representations are found in genetic programming [18]. Minsky has argued that tree representation is more suited to music as it mimics the hierarchical structure that is found in music [19].

A melody could be represented in tree form as seen in figure 3. The bottom leaves denote the pitch and duration. The higher level nodes perform operations.

Johanson and Poli employ several different operators including mirroring, concatenation, repetition and mirroring [14]

Tree based representations do not have a fixed size, as is commonly found in linear genetic representations. Care must be taken to not let the structure become too large. This is typically done by specifying the maximum depth of the tree.

3 MUSIC COMPOSITION ALGORITHMS

3.1 NEURAL NETWORKS

3.1.1 *Background*

An artifical neural networks consists of layers of artificial neuron[1] (see figure 5) that are connected to each other in a certain topology [20].

Figure 6 shows a multilayer ANN in feed-forward topology. The first layer is referred to as the input layer, the right-most layer is referred to as the output layer, the layers in between the input layer and the output layer are known as hidden layers.

Neurons have a number of inputs which are summed into an activation function. The activation function provides the output of a neuron. In figure 5 we have

$$o_j = \varphi(\sum_{i=1}^{n} x_i w_{ij})$$

A common activation function is the continous sigmoid function given by

$$\varphi(t) = \sigma(t) = \frac{1}{1 + e^{-\beta t}} \tag{1}$$

where β is the slope parameter. The derivative of the sigmoid function is easy to obtain and is given by:

$$\frac{\sigma(t)}{dt} = \sigma(t)(1 - \sigma(t)) \tag{2}$$

1 neurons are commonly referred to as units

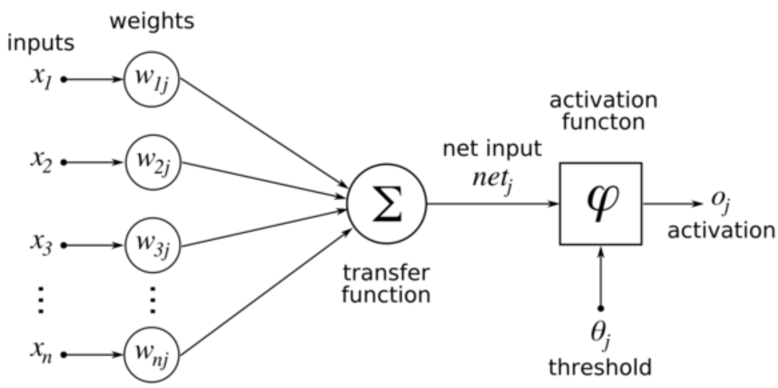

Figure 5: Model of artificial neuron

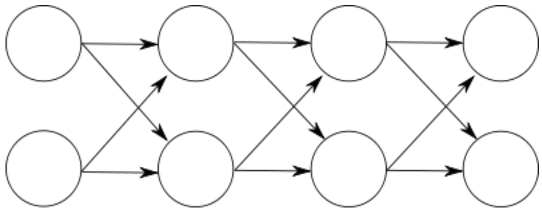

Figure 6: Figure of ANN in feed-forward topology

for $\beta = 1$

In order to determine weights for a two-layer neural network we need to minimize the error between the output of the neural network and the given target value for a set of inputs.

Thus for the output neurons we have

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - \sigma_d)^2 \tag{3}$$

where D is the set of training examples and t_d is the target output for training example d and o_d is the output of the artificial neuron for training example d We need to minimize equation 3. Obtaining the deriviative of E with respect to w_i yields:

$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)(-x_i d)$$

where x_{id} is the input for component x_i

Gradient descent is a optimization algorithm that takes steps proportional to the negative of the gradient. This we update the weights by:

$$\vec{w} \leftarrow \vec{w} - \eta \nabla E(\vec{w})$$

Applying the gradient descent algorithm we obtain a weight update rule of

$$w_i \leftarrow w_i + \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

For a multi layer network with multiple output units the back propagation algorithm needs to be used. The error needs to be summed over all the network output units

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - \sigma_{kd})^2$$

The backpropagation algorithm to determine the weights of a feed-forward neural network is given in appendix A

3.1.2 *Composition*

Neural networks have been used in music classification, in genetic algorithms as fitness functions and more complex neural networks have even been used in algorithmic music compositions.

Simple feed-forward neural networks do not contain a mechanism to remember past history.

In [3] Mozer created a recurrent connectionist neural network **CONCERT**[2][3]. The system works as follows:

- The network is trained on sample melodies from which it learns melodic and phrase constraints
- Representations of pitch, duration and harmonic structure that are based on psychological studies of human perception, based on Laden and Keefe's work [21]

The system yielded good results on simple structured artificial sequences however the system performed poorly on natural music[4]. Mozer described the system as lacking musical coherency [3]. Furthermore the system performs poorly as the length of the pieces increases.

Mozer stated the reason for failure is likely due to the Recurrent neural network (RNN) not being able to track more distant events that build global structure [3] however a LTSM recurrent network is able to achieve this goal [1].

In order to solve the problem of global structure Douglas and Jurgen attempted to use a **LTSM** network to compose musical pieces [1]. In this attempt the network was successfully able to learn a form a blues music and stay close to the relevant structure. The system used cross entropy as the error rate:

$$E_i = -t_i \ln(y_i) - (1 - t_i) \ln(1 - y_i)$$

where y_i is the output activation and t_i the target value for the $i-th$ output unit. The topology of the network was arranged as follows:

1. Four cell blocks are connected to the input units for chords
2. The last four cell blocks are connected to the inputs units for melody
3. chord cell blocks have recurrent connections to themselves and melody cell blocks
4. melody cell blocks have recurrent connections to other melody cell blocks
5. output units for chords are connected to cell blocks for chords and to input units for chords
6. output units for melody are connected to cell blocks for melody and to input units for melody

The underlaying chord structure was kept fixed.

The results indicate that a LTSM network is able to compose with both local- and global structure from a set of training data [1].

2 CONCERT - connectionist composer of ERudite tunes
3 The ER may also be read as ERratic or ERsatz
4 One critic described the resultant melodies as compositions only a mother could love

3.2 CONCLUSION

Simple feed-forward neural networks do not have the functionality to remember past history and thus do not have the capability to evaluate repetitive rhythmic patterns.

Recurrent neural networks are able to encode temporal information though initial investigation by Mozer lead to limited success as compositions lacked global structure.

Further investigations by Douglas and Jurgen indicated that LTSM networks are able to compose with local and global structure.

3.3 GENETIC ALGORITHMS

3.3.0.1 *Background*

Genetic Programming, not to be confused with evolutionary or genetic algorithms is a evolutionary algorithm based methodology to find computer programs that perform defined tasks by simplistically mirroring biological evolution. The more fit programs carry on their chromosomes into future populations. Fitness is rated by a fitness function. Other genetic operators such as recombination and crossover are also usually applied. These evolved genetic programs are usually represented in tree form. Figure 7 indicates the function $(2.2) - (\frac{X}{11}) + 7\cos(Y)$ written in tree form.

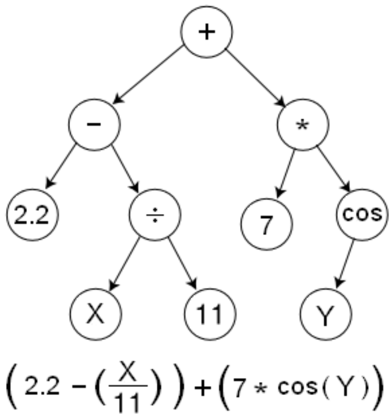

Figure 7: Figure of a example genetic program tree

A genetic algorithm consists of the following components:

1. Representation for chromosomes[5]
2. Initial population of chromosomes
3. A set of genetic operators to alter the population
4. A fitness function to assess the fitness of an individual
5. A selection method to determine which individuals in a population survive

5 Also commonly referred to as an individuals

The algorithm proceeds as follows:

1. Initial population is randomly generated
2. The fitness of each individual is assessed
3. Individuals are selected to which genetic operators are applied, e.g. Two parents are selected to generate a new child with crossover Random mutation occurs in individual
4. Various forms of selection are available that determine which individuals will be in the next generation

Genetic operators are used to generate diversity A genetic algorithm has a fixed set of genetic operators. Operators may include:

- Reproduction - A parent from the population is carried over to the next generation
- Crossover - genotype of both parents are combined using different procedures
- Mutation - A single mutation is applied to a chromosome at a set mutation rate

The choice of a **fitness function** is a big problem when using a genetic algorithm to compose music [22, 23]. There is no objective method to rate whether a melody is good or bad [24]. Traditionally when posed against such tasks the fitness function is provided interactively by the user, i.e. the user rates whether the piece is good or bad

The interactive GA approach is an approach to the fitness function where a human interactively rates the quality of a composition (fitness). A well known software that utlizes a neural network and a interactive interaction for a fitness function is GenJam [8]. The drawback of having the user interactively evaluate the fitness of individuals is that it is time consuming and poses a processing bottleneck [1]

Selection is the choice of which individuals will be chosen for the next generation. Selection concerns the reproduction and crossover operators. Some methods of selection include:

- Roulette wheel selection - chance of individual being chosen is proportional to fitness
- Tournament selection - tournament is staged between two individuals to detemine which one gets selected[6].

3.3.0.2 *GA approaches to music composition*

Genetic algorithms have already been used in a variety of work in algorithmic music composition. Some of these include:

- Thematic bridging [25]
- Composition systems using IGAs [26]

6 This is tournament selection in its simplest form

- IGAs to improvise jazz solos [8, 4]
- Integration between interactive genetic algorithms and genetic programs [27]
- Hybrid approaches employing statistical, connectionist and evolutionary elements [10]
- Various work into different fitness functions

In [28] the generated musical pieces had the style of well-known authors even when the fitness function only took relative pitch envelope into account and all generated note lengths were of fixed duration.

Thematic bridging is the application of an initial music pattern to a final pattern over a specified duration [25]. In this approach Horner modified or reordered elements in a music pattern through various operations. For example:

Given the initial note pattern of:

```
Gb Bb F Ab Db
```

and a final note pattern of:

```
F Ab Eb
```

the musical output could be:

```
Gb Bb F Ab Bb F Ab Gb Bb F Ab Eb F Ab F Ab Eb
```

by means of various operations such as mutation, rotation, deletion and so on. For thematic bridging a composite fitness function was used which rates how close the developed pattern matches the final pattern and whether the ordering of the elements are correct

Similarly a system of **variations** was developed Jacob that proceeds as follows [17]:

1. Define a primary set of motives
2. Compose phrases by layering and sequencing new motives
3. New motives are created by variations of motives already in the phrase
4. Phrases are combined together

Jacob's system had a human judge evaluate the individual chromosomes.

Normalized Compression Distance (NCD) has also been commonly used as a fitness function, for more information about the normalized compression distance see sections 4.2 (in music classification) and 5.5 (as a fitness function). Alfonseca proposed the following genetic algorithm scheme for composing melodies [28]:

1. Define a set of M musical pieces for a guide set G

2. Encode both the guides and individuals in the population as pairs of integers where the first integer represents the note interval and the second the length as a multiple of the minimum unit of time.
3. See eq. 5 on page 28 for the fitness function used
4. The 16 lowest fitness genotypes of every generation is removed
5. The 16 highest fitness genotypes of every generation are paired by means of genetic operations.

3.3.0.3 *Conclusion*

There are various variations on genetic algorithms and genetic programs, however evolutionary algorithms are a viable means to compose music.

The encoding of a musical piece as a chromosome affects the interactions of the genetic operators on the musical piece and most authors encode the problem differently.

It is important to restrict the domain of problem otherwise the search space for the genetic algorithm may be too large [17]. Most of the studies listed in this document had restricted goals. For example, using only two octaves for the notes significantly reduces the size of the search space and many real melodies comply with it [28]

The fitness function is an important part for having the genetic algorithm result to good melodies. See section 5 for insight literature has on fitness function for evolutionary algorithms. The representation of melodies for the algorithm is arguably just as important.

For this project the focus is on evolutionary algorithms and as such other procedural means of music composition will be neglected. There is too much work in music classification and as such the focus will be on only a few possible algorithms.

Fitness functions for genetic algorithms are considered in chapter 5

4 MUSIC CLASSIFICATION

4.1 INTRODUCTION

In this section we will investigate methods of music classification. If some algorithm is a good method to rate the closeness of a song to a genre or style then it may also serve as a good fitness function for a evolutionary algorithm.

4.2 NORMALIZED COMPRESSION DISTANCE

4.2.1 *Background*

The Kolmogorov complexity of piece of text is the measure of the computable resources needed to specify the text. The complexity of a string is the length of the shortest possible description of the string in a fixed universal description langauge [29].

The information distance between two string x and y is defined as the length of the shortest program p that can compute x from y and y from x The length of p can be expressed using Kolmogorov complexity [30]:

$$|p| = \max\{K(x|y), K(y|x)\}$$

The information distance p is a absolute measure. A more useful similarity metric is one that expresses the distance in relative terms. The Normalized Information Distance (NID) is given by [31] :

$$\mathrm{NID}(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}$$

The concept of NID is important, however it is not computable. An approximation of the normalized information distance is commonly used. $K(x)$ is approximated by $Z(x)$ where $Z(x)$ is the binary length of a data x compressed by a compressor Z.

$$\mathrm{NCD}(x,y) = \frac{Z(xy) - \min\{Z(x), Z(y)\}}{\max\{Z(x), Z(y)\}}$$

where $Z(xy)$ is the length of $x+y$ compressed by Z. Any good compressor may be used for Z such as

- gzip
- bzip2
- Lempel-Ziv and its variations

4.2.2 *Literature*

Normalized compression distance has been used in a variety of cases. It has been used in applications of general clustering and classification of data in arbitrary domains. This includes music classification [26].

Cilibrasi and Vitiyani used the Normalized Compression Distance to approximate the Kolmogorov Distance between different musical pieces as a method to compute clusters of music [26]. The MIDI files were pre-processed such that when two notes occur at the same time only the note with the highest pitch is kept. The music was represented as a string and the distances between different musical pieces was computed.

Ctaltepe, Sonmez and Adali used the normalized compression distance and used it to classify music pieces using k-nearest neighbors [32]. The training data has a label associated. The closest k training data (by NCD) to a song is obtained and the most frequent label in the k set is used to classify the musical piece. Ctaltepe Sonmez and Adali found that the distance measure works better when more training data is available and the performance is dependent on how the input data is pre processed. The best results were obtained when the midi files were sampled at 1ms and the $k = 1$ nearest neighbor identification was used. The music was represented in the following format: outputting the first note and then the difference in pitch between consecutive notes. Using the above means a classification accuracy of 79% was achieved on 57 midi files.

Li and Sleep have also found that the 1-nearest neighbor with a Lempel-Zip compressor outperformed more complex statistical methods and compressors. Using relative pitch intervals in the music representation outperformed using absolute pitches. A performance of 92.35% was obtained [15]. The midi files were organized into four categories Beethoven (302 files), Haydn (261 files), Chinese (80 files) and Jazz (128 files). The dataset is unbalanced and the study does not make it clear which partition of the dataset was used as training samples and what partition was used for verification.

In [28, 33, 15, 6] it was found that the normalized compression distance serves as a promising fitness function for genetic algorithms for automatic music generation. Thus NCD may be viably used as a fitness function for a genetic algorithm and as a metric to help classify music.

4.3 NEURAL NETWORKS

McKay investigate using K-nearest neighbor techniques and artificial neural networks in order to classify MIDI music by genre. He included some of the following metrics as input to the neural network [34]:

- Number of notes - standard deviation of number of notes activated in each channel
- Note duration - standard deviation of total duration of notes
- Dynamics - standard deviation of average volume of notes

- Melodic Intervals - average melodic interval
- Simultaneity - average number of notes that are played concurrently
- Note density - average number of notes per second
- Average time between attacks - average time between note activations
- Initial tempo - tempo in beats per minute
- Pitch variety - number of pitches used at least once
- Most common pitch class - Most common pitch divided by number of possible pitches

For a full list please see his MacKays dissertation on MIDI classification [34] Using the complete list of metrics neural networks had a success rate of 83%.

5 FITNESS FUNCTIONS

5.1 INTRODUCTION

In this section we will briefly review some functions that may be used as a fitness function for a genetic algorithm.

Algorithms that help classify music could possibly be used as fitness functions, as this would rate the similarity of the evolved piece of music to a genre or style. These algorithms are found in section 4. If work has been done that has used the music classification algorithm as a fitness function it will be listed in this section.

5.2 ZIPF'S LAW

Zipf's law states that the frequency of an event is inversely proportional to its statistical rank, that is:

$$f \propto r^{-a}$$

where f is the frequency of occurence of a particular event and r is the statistical rank. a is close to 1. Zipf's law can also be stated as:

$$P(f) \approx \frac{1}{f^n}$$

where $P(f)$ denotes the probability of an event of rank f and n is close to 1

One can determine whether melodic intervals follow Zipf's law by counting the melodic intervals in a piece. The result is usually plotted on a logarithmic scale and is known as a rank-frequency distribution

Zipf found evidence for the theory in music. An analysis of Mozart's Bassoon Concerto in Bb Major revealed an inverse relationship between the length of intervals between repetitions of notes and the frequency of their occurrence [35].

Several other different rank frequency distributions can be obtained for a musical piece. These include [11]:

- Pitch - distribution of MIDI pitches
- Chromatic tone - distribution of 12 chromatic tones
- Duration - note durations
- Melodic interval - distribution of melodic intervals
- Melodic bigrams - distribution of adjacent melodic interval pairs

Linear regression is performed to obtain the slope of the distribution. The coefficient of determination R^2 is also computed in order to determine how well the slope fits the data.

1 Figure generated by Jensen for his thesis [7]

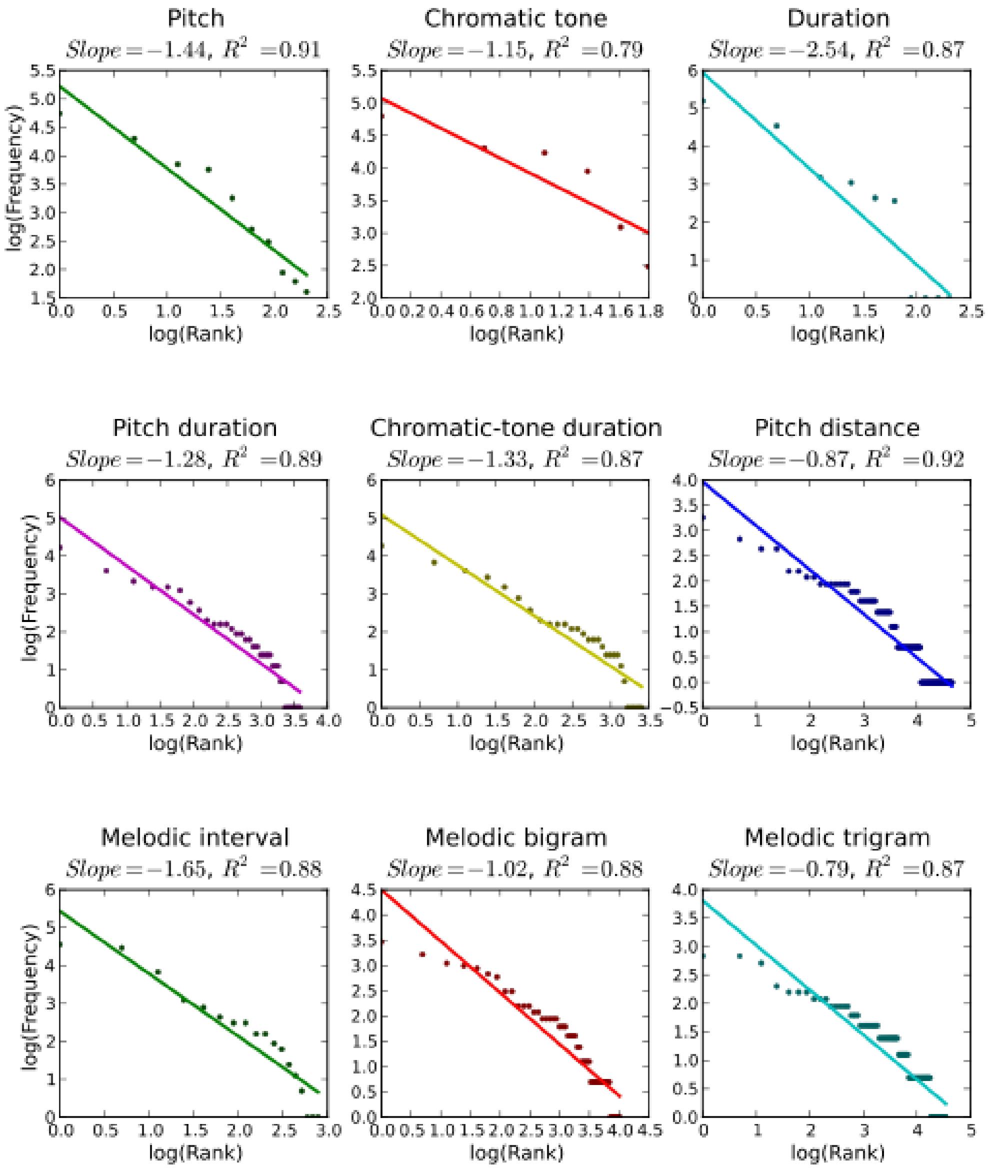

Figure 8: Rank frequency distributions and slopes of different metrics for The Beatles' Let It Be [1]

Figure 8 plots the rank frequency distributions of various metrics for the Beatles' song Let It Be. The figures were generated by Jensen for his thesis [7]. The metrics can be seen to follow a Zipfian distribution. Results by Manaris also indicate that most music pieces display near Zipfian distributions [11].

Zipf-based metrics capture essential parts of scaling properties in music. These metrics indicate that music follows a distribution balanced between near-zero slope and steep negative slope. Different styles of music have different slopes. There exists also a correlation between Zipf metrics and human preference [11].

Jensen used a Gaussian to define the target fitness as [7]:

$$f_m(a, b) = e^{(\frac{b-a}{-\lambda})^2}$$

where a is the metric slope of an evolved piece of music, b is the target slope and λ is the tolerance

Since there are several metrics for a given piece of music the fitness function should incorporate these. Jensen used the weighted sum of several metrics.

$$f(\vec{a}, \vec{b}) = \sum_{i=1}^{N} w_i f_i(a_i, b_i)$$

Jensen has found that Zipf metrics can be used to evolve pleasant music using a tree-based representation, however the majority of the evolved melodies were unpleasant [7]. Zipf metrics only capture the scaling properties of distributions and ignore the musical events that account for different frequencies. Zipf's law neglects musical content and can be seen as knowledge weak. Jensen concluded that the Zipf metrics are insufficient for musical fitness alone.

Manaris had more success with Zipf metrics however he used them as input to an artificial neural network to evaluate the fitness of melodies, however Manaris states it is wiser to use the fitness function in a partially interactive system [11]

5.3 COSINE SIMILARITY

In Information Retrieval cosine similarity is commonly used to asses the similarity of two documents:

$$\text{sim}(\vec{A}, \vec{B}) = \cos\theta = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}||\vec{B}|}$$

where $\vec{A}$ and $\vec{B}$ are two document vectors

In order to rate the similarity between music scores features such as pitches and melodic intervals are used.

As with Zipf's law the fitness is the weighted sum of similarity measures:

$$f(\vec{A_1}, \dots, \vec{A_n}; \vec{B_1}, \dots, \vec{B_n}) = \sum_{i=1}^{N} w_i f_i(\vec{A_i}, \vec{B_i}) \tag{4}$$

The fitness function rates the fitness of the evolved individual with a target piece.

Jensen conducted multiple experiments using the fitness function in equation 4. At first only a single metric was included and thereafter multiple metrics. As more metrics were included the evolved melodies became more similar to the target piece. More pleasant melodies were evolved when the target piece was The Beatle's Let It Go than Mozart's Piano Sonato No. 16. There was no correlation between melody and rhythm. Metrics included for the fitness function were:

- Pitch
- Chromatic tone
- Melodic interval
- Melodic bigram
- Rhythmic interval
- Rhythmic bigram

Jensen concluded that the results obtained by the cosine simality fitness function were more pleasant than those obtained by Zipf's law as Zipf's law rates music on scaling properties only [7].

5.4 NEURAL NETWORKS

Different forms of networks have been used as a fitness function for evolutionary algorithms. Some of these include:

- Adaptive resonance theory neural networks using binary classification patterns [9]
- Recurrent neural networks
- Cascade correlation neural network designed to reduce GenJam bottleneck [8]

Some common problems with neural networks as fitness functions is that they require a lot of time to be trained, require a good representation for a set of inputs to map to an output and the structure of the neural network is fixed after training [9].

In [8] Biles tried to design a cascore neural network to rate musical scores. Since a neural network outputs fitness based on the input parameters the choice of input metrics are important. Metrics that included the number of new note events in a measure, the number of unique new note events, the size of the maximum interval, the number of changes in a direction between adjacent notes failed to capture the fitness for the Artificial Neural Network (ANN) [8].

Biles argues that the reason for this is that humans listen to music in more complex ways and that simple statistical measures fail to capture this. Zipfs law, which only captures the scaling properties of music also yielded poor results as a fitness function for similar reasons [7] (See section 5.2).

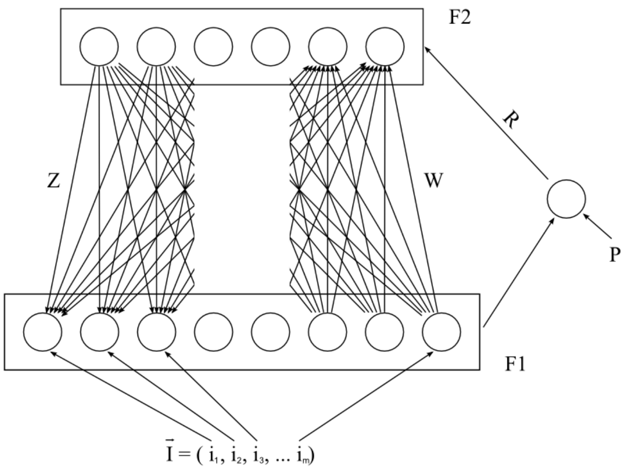

Figure 9: Figure of ART ANN topology

An **Adapative resonance theory (ART) network** has been used as a fitness function whereby a Genetic Algorithm (GA) utilizes clustered representations of rhythm styles to interactively generate rhythm patterns to according to a certain style [9].

An adaptive resonance theory network utilizes unsupervised learning and clustering algorithms to recognize patterns. New clusters are created if a pattern cannot be associated with existing clusters. Another characteristic of ART networks is that new training does not cause loss or corruption of old training data [36]

A ART1 network clusters binary vectors. The basic structure of an ART1 network involves (See figure 9) :

1. Input processing field - F1
2. Cluster units - F2
3. Mechanism to control degree of similarity of patterns in the same pattern
4. weighted bottom-up connections between F1 and F2 layers
5. weighted top-down connections between F2 and F1 layers

Burton had the ART network fitness function operate as follows [9]:

1. Each individual in the population is an input to the ANN
2. The network determines the winning cluster
3. The degree of similarity between the individual and the cluster is determined
4. If the degree of similarity is above a certain threshold the individual is added to the cluster

5. If no clusters match the individual closely enough a new cluster is created
6. Fitness is assigned as a degree of similarity to a cluster.

NEUROGEN is another attempt at using a neural network as a fitness function to compose small diatonic, western, four part harmony compositions. The system has shown limited success however it was able to produce 4 bars of music [2].

Chen used a **Simple Recurrent Network (SRN)** with composition rules on tonality and rhythm as a fitness function for a GA [5]. The simple recurrent network has an input layer that represents a measure at time T with the output layer representing the measure at time $T+1$. A recurrent network is used as a single step predictor to compose music. The network predicts notes at $T+1$ using the notes at time T. After the network has been trained it can be seeded with inital values to generate novel compositions. The following constraints were used:

1. Pitch diversity constraint - number of measures with unique pitch sequences
2. Rhythmic diversity constraint - number of measures with unique signatures
3. Measure density constraint - ratio of number of notes to maximum notes
4. Pentatonic pitch class constraint - number of notes that belong to pentatonic pitch class
5. Cell structure - ratio number of times cell pattern occurs to maximum number of patterns

The system was able to generate melodies with systematic structure however it lacks global structure. Individual measures sound pleasant and diverse however there is a lacking structure as a whole

5.5 NORMALIZED COMPRESSION DISTANCE

As noted in section 4 the Normalized Compression Distance has been used to help classify music genres. However Normalized Compression Distance has been explored as a possible fitness function for evolutionary algorithms [28, 33, 6]

The fitness function used by Alfonseca et. al for an individual x and a guide set G was defined as [6]:

$$f(x) = \left(\sum_{g_i \in G} \mathrm{NCD}(x, g_i) \right)^{-1} \tag{5}$$

Where g_i is a guide in guide set G containing M musical pieces and x is the set of differences between consecutive notes.

Alfonseca encoded the chromosomes as N vectors containing a pair of integers. The first integer denotes the note interval and the second represents the length as a multiple of the smallest unit of time [28].

5.6 INTERACTIVE FITNESS FUNCTIONS

Genetic algorithms which employ user interaction as a means of rating the fitness of are known as Interactive Genetic Algorithms (IGAs).

In [12] constructs a system that composes 16-bars music using a GA. The user rates individual chromosomes, new chromosomes are applied by genetic algorithm and the user is asked to rate the individuals again. Should the user find a good piece they may favorite it. The fitness of the chromosomes were seen to increase as the generations increased, however it is unknown whether the melodies were pleasant.

Using an interactive fitness function may lead to better results than most other fitness functions however it is a tedious and demanding process and may also lead to inconsistencies in evaluation. Some researches try to reduce this effect by constructing ANNs which learn the user's ratings, and as such may be used in place of the interactive fitness function [8, 13]. See section 5.4 for the use of ANNs as fitness functions.

Johnson and Poli had a user rate individual sequences and trained a neural network base automatic rater, which may replace the user in larger runs. The musical pieces generated by the automatic rater were pleasant but they were not as pleasant as the musical pieces generated by the user interactive runs [14]

The superiority of a interactive fitness can be seen, as a person can rate the pleasantness, or fitness of a song much more accurately than current quantitative fitness functions. However this imposes a bottleneck on the system as it is time consuming. A partially interactive system may yield a good compromise [8].

6 CONCLUSION

We have investigated numerous methods to compose music algorithmically. The two most prominent methods currently to compose music is through genetic algorithms and neural networks.

Genetic algorithms require proper music representation and a good fitness function. Multiple work has gone into investigating various possible fitness functions. The three most promising candidates are NCD, ANN and cosine similarity.

Interactive genetic algorithms yield good results although this imposes a performance bottleneck on the system, as the system is required to wait for user input. This is a time-consuming process although a partial interactive system might be viable.

Simple feed-forward neural networks are unable to compose music due to their inability to encode temporal information. Recurrent neural networks were investigated and early findings yielded poor results, though LTSM networks seem promising.

Algorithmic music composition is viable although there is large room for improvements to be made. Currently only short musical pieces sound pleasant. Longer pieces tend to be repetitive and lack global structure.

Most current methods restrict their domain in order to investigate only the main research questions.

A hybrid approach may yield good results.

7 NOTES

Part III

APPENDIX

A

ALGORITHMS

A.1 BACKPROPOGATION

For a feed forward network[1] with n_{in} inputs, n_{hidden} hidden units and n_{out} output units the back-progogation algorithm works as follows

1. Initialize network with random weights
2. Repeat until algorithm terminates
3. • $\forall(\vec{x}, \vec{t}) \in$ training examples do
   • a) Input instance $\vec{x}$ to network and compute $o_u \ \forall\ u \in$ network
     b) For each network output unit k calculate error term
     $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$
     c) For each hidden unit h calculate the error term δ_h
     $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\delta_k$
     d) Update each network weight w_{ji}
     $w_{ji} \leftarrow w_{ji} + \eta\delta_j x_{ji}$

The complete derivation of the back-propogation algorithm for feed forward artificial neural networks can be found in Mitchell (2007) [37]

1 Other topologies exist with more complex methods for obtaining the weights

B PLANNING

Project breakdown and schedule

B.1 WORK BREAKDOWN STRUCTURE

Figure 10 indicates the work breakdown structure of the project. The front end design refers to the user interface and the interaction between the user and the application (IF 1.0).

The back end design refers to the logic of the application and this includes reading the audio files, generating new music and playing music.

Research will be carried out in order to determine the algorithms to be used for algorithmically generating music and to find a suitable framework for developing the software (considering audio playback and similar factors)

Since there are a variety of algorithms, prototyping will be employed to test which algorithms are feasible by means of a trade-off study.

The work breakdown structure allows focus on critical elements of the project and their relation to the whole. This allows us to focus on discrete tasks that are realistic and achievable and keeps the project on track.

B.2 SCHEDULE

In order to successfully complete the project the following tasks must be executed:

1. Researching machine learning algorithms for music composition
2. Prototyping the algorithms in order to find a feasible subset
3. Collecting a library of audio files to be used as input for machine learning algorithms
4. Developing the required back end components for the software
5. Developing a user interface and designing the interaction components
6. Testing the application, fixing bugs and adding features until the specifications are met

Table 2 indicates the estimated length of these activities and the estimated time of completion. Figure 11 is a visual representation of the schedule using bar charts.

B.3 BUDGET

Since the project is a software application no external components will be required. The application will run on a platform that is capable of storing audio files and playing back sound.

Some unplanned costs that might arise include:

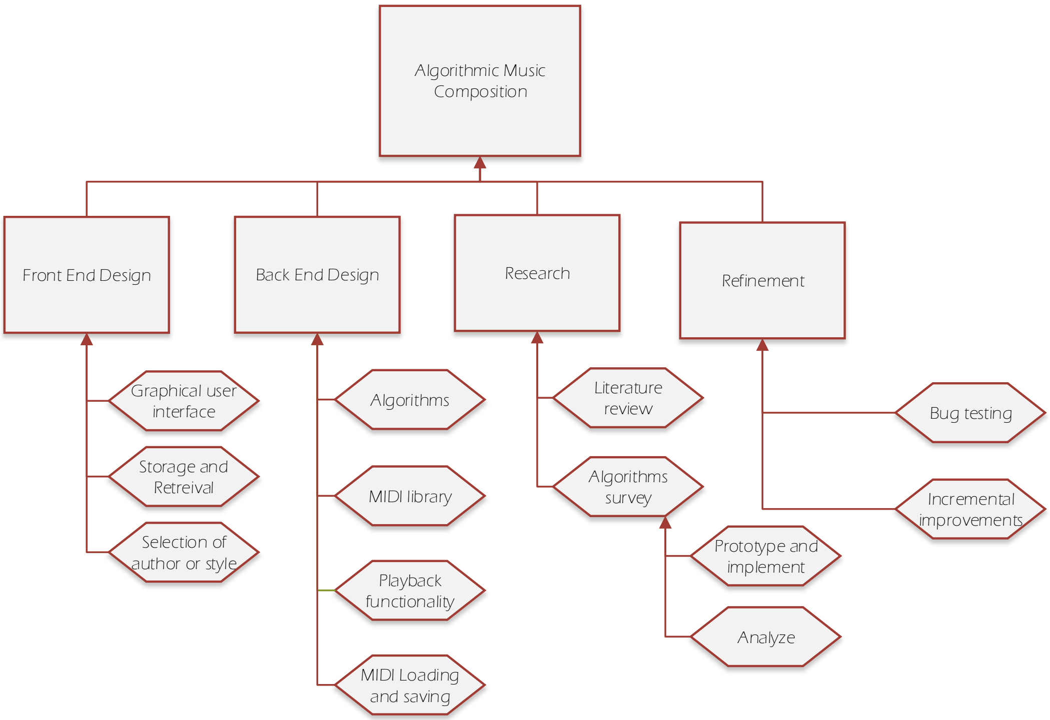

Figure 10: Figure of the work breakdown structure

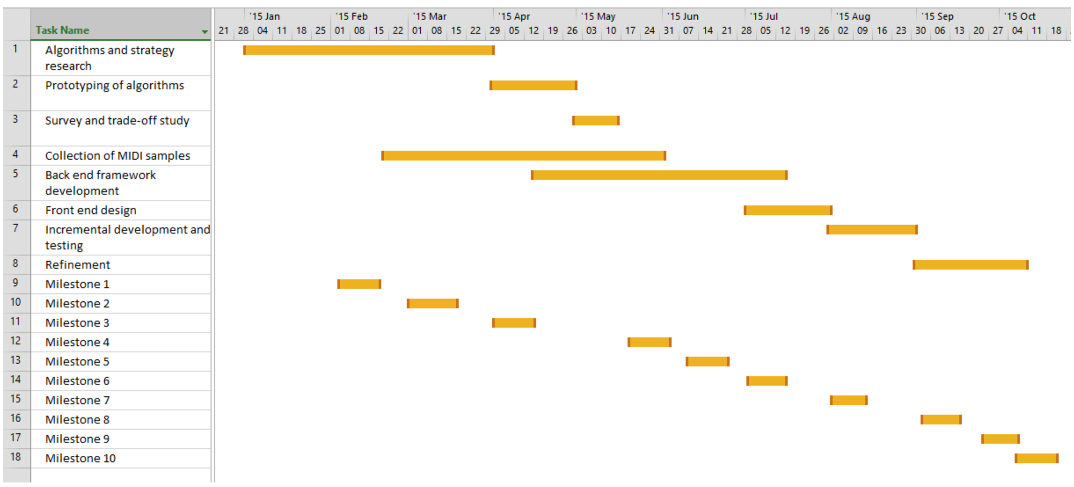

Figure 11: Gantt chart of project schedule

Table 2: Schedule for activities

Task Name	Duration	Start	Finish
Algorithms and strategy research	64 days	Thu 15/01/01	Tue 15/03/31
Prototyping of algorithms	23 days	Tue 15/03/31	Thu 15/04/30
Survey and trade-off study	12 days	Thu 15/04/30	Fri 15/05/15
Collection of MIDI samples	72 days	Fri 15/02/20	Mon 15/06/01
Back end framework development	66 days	Wed 15/04/15	Wed 15/07/15
Front end design	23 days	Wed 15/07/01	Fri 15/07/31
Incremental development and testing	22 days	Fri 15/07/31	Mon 15/08/31
Refinement	31 days	Mon 15/08/31	Sat 15/10/10
Milestone 1	11 days	Wed 15/02/04	Wed 15/02/18
Milestone 2	14 days	Sun 15/03/01	Wed 15/03/18
Milestone 3	11 days	Wed 15/04/01	Wed 15/04/15
Milestone 4	11 days	Wed 15/05/20	Wed 15/06/03
Milestone 5	11 days	Wed 15/06/10	Wed 15/06/24
Milestone 6	10 days	Thu 15/07/02	Wed 15/07/15
Milestone 7	10 days	Sat 15/08/01	Thu 15/08/13
Milestone 8	10 days	Thu 15/09/03	Wed 15/09/16
Milestone 9	9 days	Fri 15/09/25	Wed 15/10/07
Milestone 10	11 days	Wed 15/10/07	Wed 15/10/21

- Internet costs
- Acquisition cost of audio files
- Obtaining access to a study or article
- Audio equipment

BIBLIOGRAPHY

[1] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza . . .* , 2002.

[2] P. M. Gibson and J. A. Byrne, "Neurogen, musical composition using genetic algorithms and cooperating neural networks," in *Artificial Neural Networks, 1991., Second International Conference on*, pp. 309–313, IET, 1991.

[3] M. C. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, no. 2-3, pp. 247–280, 1994.

[4] J. a. Biles, "GenJam: A genetic algorithm for generating jazz solos," *Proceedings of the International Computer Music Conference*, pp. 131–131, 1994.

[5] C.-C. Chen and R. Miikkulainen, "Creating melodies with evolving recurrent neural networks," *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, vol. 3, 2001.

[6] M. Alfonseca, M. Cebrian, and a. Ortega, "A Fitness Function for Computer-Generated Music using Genetic Algorithms," *WSEAS Transactions on Computers*, 2006.

[7] J. H. Jensen, *Evolutionary Music Composition*. PhD thesis, 2013.

[8] J. a. Biles, P. G. Anderson, and L. W. Loggi, "Neural Network Fitness Functions for a Musical IGA," *Architecture*, 1996.

[9] A. R. Burton and T. Vladimirova, "Genetic Algorithm Utilising Neural Network Fitness Evaluation for Musical Composition," in *In Proceedings of the 1997 International Conference on Artificial Neural Networks and Genetic Algorithms*, pp. 220–224, Springer-Verlag, 1997.

[10] B. Manaris, P. Roos, P. Machado, D. Krehbiel, L. Pellicoro, and J. Romero, "A corpus-based hybrid approach to music analysis and composition," *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, p. 7, 2007.

[11] B. Manaris, P. Machado, C. Mccauley, J. Romero, and D. Krehbiel, "Developing Fitness Functions for Pleasant Music : Zipf's Law and Interactive Evolution Systems," *EvoWorkshops*, pp. 498–507, 2005.

[12] M. Unehara and T. Onisawa, "Construction of Music Composition System with Interactive Genetic Algorithm," *Evaluation*.

[13] L. Spector and A. Alpern, "Induction and Recapitulation of Deep Musical Structure," in *In Proceedings of the IJCAI-95 Workshop on Artificial Intelligence and Music*, pp. 41–48.

[14] B. Johanson and R. Poli, "{GP}-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters," *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 181–186, 1998.

[15] M. Ling and R. Sharp, "Melody classification using a similarity metric based on kolmogorov complexity," *Conference on Sound and Music Computing*, 2004.

[16] MIDI Manufacturers Assosciation, "MIDI 1.0 Detailed Specification," first edition of a recommendation, MIDI Manufacturers Assosciation, 1995.

[17] B. L. Jacob, "Composing with genetic algorithms," *In Proceedings of the 1995 International Computer Music Conference*, no. September, pp. 452–455, 1995.

[18] W. Langdon, *A Field Guide to Genetic Programing (Summary for Wyvern)*. No. March, 2008.

[19] M. Minsky and O. Laske, "a C Onversation With M Arvin M Insky," *AI Magazine*, vol. 13, no. 3, pp. 31–45, 1992.

[20] C. M. Bishop and Others, "Neural networks for pattern recognition," 1995.

[21] B. Laden and D. H. Keefe, "The representation of pitch in a neural net model of chord classification," *Computer Music Journal*, pp. 12–26, 1989.

[22] G. G. Wiggins, G. Papadopoulos, S. Phon-amnuaisuk, and A. Tuson, "Evolutionary methods for musical composition," *In International Journal of Computing Anticipatory Systems*, 1998.

[23] D. Matić, "A genetic algorithm for composing music," *Yugoslav Journal of Operations Research*, vol. 20, no. 1, pp. 157–177, 2010.

[24] M. Y. Lo, *Evolving Cellular Automata for Music Composition with Trainable Fitness Functions*. PhD thesis, 2012.

[25] A. Horner and D. E. Goldberg, "Genetic Algorithms and Computer-Assisted Music Composition," 1991.

[26] R. Cilibrasi and P. M. B. Vitanyi, "Clustering by compression," *Information Theory, IEEE Transactions on*, vol. 51, pp. 1523–1545, Apr. 2005.

[27] N. Tokui and H. Iba, "Music composition with interactive evolutionary computation," *Proceedings of the Third International Conference on Generative Art*, pp. 215–226, 2000.

[28] M. Alfonseca, M. Cebrián, and A. Ortega, "A simple genetic algorithm for music generation by means of algorithmic information theory," *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pp. 3035–3042, 2007.

[29] A. N. Kolmogorov, "On tables of random numbers," *Theoretical Computer Science*, vol. 207, no. 2, pp. 387–395, 1998.

[30] C. H. Bennett, P. Gacs, M. Li, P. M. B. Vitanyi, and W. H. Zurek, "Information distance," *Information Theory, IEEE Transactions on*, vol. 44, pp. 1407–1423, July 1998.

[31] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitanyi, "The similarity metric," *Information Theory, IEEE Transactions on*, vol. 50, pp. 3250–3264, Dec. 2004.

[32] Z. Cataltepe, A. Sonmez, and E. Adali, "Music Classification Using Kolmogorov Distance," *Representation in Music/Musical Representation Congress*, no. Bishop 57, 2005.

[33] M. Alfonseca, M. Cebrián, and A. Ortega, "Evolving computer-generated music by means of the normalized compression distance," in *WSEAS Transactions on Information Science and Applications*, vol. 2 of *SMO'05*, (Stevens Point, Wisconsin, USA), pp. 1367–1372, World Scientific and Engineering Academy and Society (WSEAS), 2005.

[34] C. Mckay, "Automatic Genre Classification of MIDI Recordings," *Technology*, vol. MA, no. June, pp. 1–150, 2004.

[35] G. K. Zipf, *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley Press, 1949.

[36] G. A. Carpenter and S. Grossberg, *Adaptive resonance theory*. Springer, 2010.

[37] T. M. Mitchell, *Machine Learning*, vol. 4 of *McGraw-Hill Series in Computer Science*. McGraw-Hill, 1997.

DECLARATION

Put your declaration here.

Potchefstroom, June 2, 2015

Stefan Jacholke