A REPORT ON

# AI IN MUSIC: AUTOMATIC COMPOSITION

Jhanavi Sheth
(2013A7PS096P)

Sanket Shah
(2013A7PS119P)

Akhilesh Sudhakar
(2013A7PS173P)

# TABLE OF CONTENTS

# INTRODUCTION

Algorithmic composition of music using artificial intelligence began in the early 90's and since then a variety of approaches have been employed for the same. However, the concept of music being formed from a set of rules and structural norms has existed from many years and this underlying organized structure in music has probably led to the idea of training a system to make music without human intervention. This problem does not merely mean generating music electronically using a given input sequence but looks at the more complicated task of generating new sequences itself, but yet ones that are pleasant to the ear, rhythmic and melodic. The 'intelligence' factor lies in bringing out the creativity that humans have in the art of making good music, yet not in a completely random fashion. The task of composing music algorithmically will be implemented using three algorithms- genetic algorithm, Markov chain and artificial neural network. A comparison will be drawn between the performance of all 3 algorithms and the complexities involved in each of these.

The need for the algorithmic approach is that the human brain does not have unlimited creativity in composing music and that one believes that interesting musical patterns will arise when artificially composed. It is possible that our minds are restricted to only a certain domain of creativity and certain genres of music, while a computer has no such restrictions. One hopes that this approach will lead to breakthroughs like discovering a new genre itself or making better music in the same genre.

The challenges faced all arose out of the fact that art is not something that can be computed or quantified easily and hence, one has to ensure a way in which the computer does not generate random chaotic music but one that is still artful and brings with it melody and rhythm.

# LITERATURE REVIEW

**ANN:**

- I started with the survey paper (Fernández, Vico 2013) that talked about the history of multiple algorithms and their use in artificial music composition.
- I read some of the older papers but a quick look at their results were far from satisfactory. (Todd, 1989), (Duff, 1989)
- Because I was forced to stick to only one algorithm, I went with this simplistic approach but realized that it was unsatisfactory.
- I finally referred to the blog (http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/) for specifics on the implementation using ANNs.

**Genetic Algorithm:**

- I referred the paper by (Alfonseca, Cebrian, Ortega undated) on fitness functions for genetic algorithms in music, where I got the perspective of using compression distance algorithms as fitness function
- Next, the IEEE paper (Maeda Y, 2010) suggested using Interactive Genetic Algorithm, which I used as a performance measure
- The blog was a huge insight into the different parameters one had to keep in mind while choosing the best samples from the population during the iterations
- Performance measures were explored in (Madsen, Widmer, 2005) http://www.cs.uml.edu/ecg/uploads/Alfall11/final-paper.pdf

**Markov:**

- In Simone Hill, Markov Melody Generator, it's been mentioned how one can divide the task of music generation through markov algorithm into two parts: analysis and generation. This can be implemented using transition matrix.

# DESCRIPTION OF ALGORITHMS USED

## 1) Markov chains

In their simplest incarnations, Markov chains can be represented as labeled directed graphs: nodes represent states, edges represent possible transitions, and edge weights represent the probability of transition between states. However, Markov chains are more commonly represented as probability matrices. The probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it.

### Applying Markov Chains to Music

When Markov chains are applied to music composition, the most common way to use them in research is to induce the probability matrices from a corpus of pre-existing compositions (training set). In a first-order chain, the states of the system become note or pitch values, and a probability vector for each note is constructed, completing a transition probability matrix. To calculate the transition probability matrix, one can use anything, including arbitrary data. However, it may be more interesting to model the melody generator after other melodies.

The main algorithm using Markov chains can be split into 2 algorithms:

1. **Analysis of algorithm**

- This algorithm builds the transition probability matrices after reading in the inputs.
- Transitional probabilities are stored in a list of lists matrix that allocated on each row and column to each feasible rhythmic subdivision of a musical measure.

**1. Generating algorithm**

- This would generate the Markov chain.
- To do this, a key would be randomly selected as the starting point and then randomly select another state it would go to.
- The next key chosen would be dependent on its frequency. After that, the new key would be chosen as the ‘starting’ point and the process would start over.

Two different types of Markov chain may be used for the algorithm:

1. Simple Markov Chain(First Order)

   This only looks at the previous note and determines what to play next.

1. Complex Markov Chain(nth Order)

   This looks at the last n nodes and chords before and based on those determines which note to play next.

Since Complex Markov Chain looks at more chords and notes in the past to determine what note to play next it seems that it will fare better.

**Pseudo code:**

The following steps list out the algorithm in detail:

Steps 1 to 8 entail analysis algorithm and later steps entail the generate algorithm for single markov chain algorithm.

1. Start
2. n=0
3. repeat steps 3 to 7 till n=MAX_N
4. read midi file
5. populate transition matrix
6. n=n+1
7. Goto step 3
8. for each i,j transitionmatrix(i,j) = transitionmatrix(i,j)/sum where sum = sum of entries in row i
9. initialise other attributes of the midi file
10. randomly generate a chain of notes
11. Convert matrix to midi format
12. Write the same to a new midi file
13. End

Time complexity:

O(nlogn)

where n is the number of notes

Space complexity

O(n^2)

**2)**

**Genetic Algorithm**

Evolutionary algorithms are biologically inspired and metaheuristic. The mechanisms used are akin to evolution in biology, i.e.,

1) Reproduction
2) Mutation
3) Recombination
4) Selection

An appropriate fitness function, which evaluates individuals produced during the iterations is designed based on the problem.

Genetic algorithms are a class of evolutionary algorithms that use crossover techniques, with an array of bits as the standard representation of the genotype.

Any evolutionary algorithm involves the following stages:

- Designing a representation
- Deciding how the population must be initialized
- Mapping genotype to phenotype
- Finding a method of evaluating an individual
- Further:
- Designing suitable mutators and then recombinators
- Defining a criterion for selection of individuals who will be parents
- Defining replacement criterion for individuals
- Stopping conditions

In the context of music composition algorithmically, the algorithm will be implemented in the following manner:

Compositions will be represented as an array of notes, each note itself being a structure composed of three parameters- pitch, octave and duration.

As input, the following parameters are considered:

- Values that show how similar the current composition is to the original reference composition
- Interval values
- Set of tones that are considered ‘good’ and those considered ‘bad’
- Allowed deviation from reference values
- Weights for each of the evaluation criteria

The output is a musical composition.

**Working:**

The initial population of compositions is either randomly selected or selected based on input parameters (for example, existing ‘good’ compositions). In each iteration, the Genetic Algorithm scans the search space (the current population) and calculates the fitness function for each individual composition in the population. A sort operation is done on the individuals based on their fitness. Further it checks the best individual (the one with the best fitness function value) and sees if this individual’s fitness is greater than equal to a preset threshold fitness value. Then it checks if the iteration count has exceeded the preset iteration limit. If both these conditions are false, then it proceeds to pick out the top one-third of the entire search space that consists of the most fit individuals. The mutation operators apply on these picked out individuals, to generate new genetically formed individuals, who are added back into the population. From this expanded population, the composition with the lowest fitness values are removed to maintain a fixed population size. The next iteration begins after this. The algorithm stops when the iterations exceed a limit, in which case the ‘most fit’ individual in the current population is output or when the threshold of fitness is satisfied by a composition, in which case it is declared as the best composition and hence, the output of the algorithm.

Since music is very subjective, designing a fitness function that can be calculated by the computer might not be the best possible solution. Hence, a variant of Genetic Algorithms are used, called Music Interactive Genetic Algorithms, where the fitness function also assigns a weight to the user’s intuitive evaluation.

**Time complexity:**

- O(k.n.a), where a is the number of iterations used, n is the number of sample data points and a is the number of attributes considered per sample point
- High for genetic algorithms, because of combinatorial explosion

**Space complexity**:

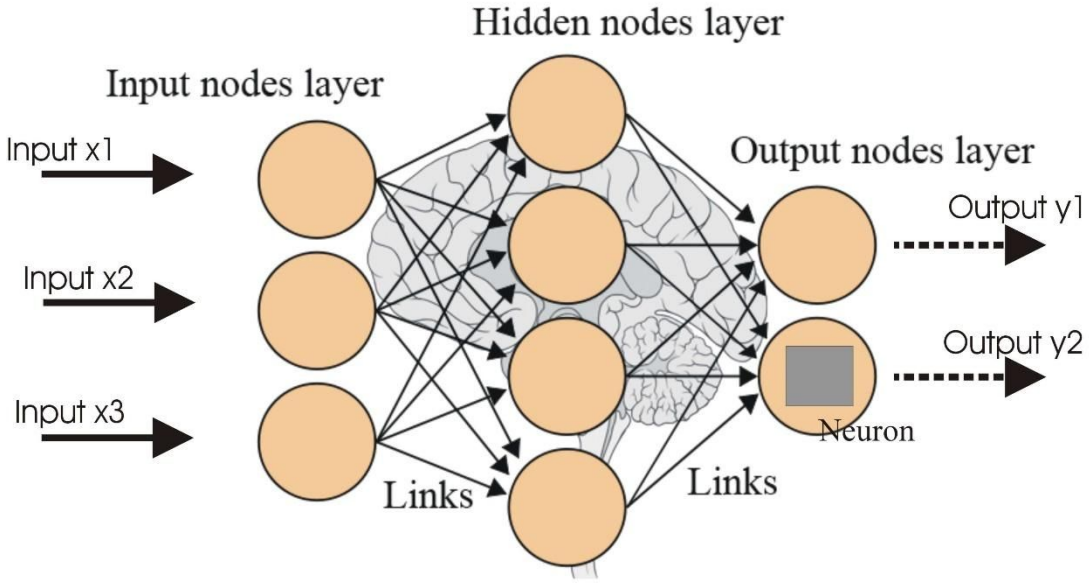
- O(n)

### 3) Artificial Neural Networks (ANN)

**Overview**

ANNs are biologically inspired computation models that were created to simulate intelligence. They are based on the human brain and emulate neurons and synapses. The synapses are modeled as nodes in a network with the artificial neurons being the links between them. These links have a specific direction as well as weight. In the network, the nodes are organized into three so-called interconnected layers – input, hidden and output. The input layers receive input from the data, the output layer produces the data to be read and everything in the middle is considered to be a black box, or hidden layer (it can consist of many layers internally). These nodes each have an activation function that transforms data from the various input nodes using the weights of different connections and a mathematical

function to useful output, which is then propagated through the system.




*Structure of ANN (courtesy: http://futurehumanevolution.com/artificial-intelligence-future-human-evolution/artificial-neural-networks)*

This network learns through a method called back-propagation and is a supervised learning technique. The training data consists of inputs and desired outputs. Initially, the connections are assigned random weights. The input is then fed into the network and the output is generated. This is then compared to the desired output and the difference is used to update the weights of the different connections to get a better output. This is repeated iteratively to get a closer value.

**Use in music composition**

Here, input configurations are provided and are mapped to windowed segments of songs (monophonic tones) that are temporally separated. Different configurations are mapped to different outputs and a new input configuration will then provide music intermediate to different compositions that have been entered. This is just one method as mentioned in (Todd

& Loy, 1991), however the relation between input and output can be modeled differently and this will impact the result as well.

ANNs can also be used to automate musical composition tasks such as harmonization and not just entire musical sequences.

**Pseudo-Code:**

1. Model ANN appropriately.
2. Initialise weights.
3. Supply input and predict output.
4. Calculate cost/error using metric.
5. Update weights via back-propagation.

**Space Complexity:**

1. Similar to graphs with each element as a node in the Neural Network.
2. More space taken up by the weights of each node than the representation.

**Time Complexity:**

1. Dependent on the size and complexity of the network as well as back-propagation technique and activation function used.

# PERFORMANCE MEASURES

Music is an art and as a consequence the performance of the algorithm would depend on the aesthetic quality of the output generated by it. In this case, there need not be any consensus on what is considered to be good or bad. As a result, any performance measure is based on how an algorithm/technique performs with respect to a certain metric, but in this case, the metric itself is hard to design. These are the approaches taken by different papers, almost all of which require human input at some level or another:

1. **Human Critics:** Users are expected to sit through and rate the music based on their preference. These are used to rate the performance of the algorithm as well as to optimize them. (Tokui and Iba, 2000)
2. **Artificial Critics:**
    a. Correlation with popular music: The correlation between popularity and aesthetic sensibility is assumed and the number of downloads from a popular website is assumed to be an indicator of popularity. In (Manaris et al., 2007), they download the most and least downloaded songs and create a feed-forward ANN to extract features from each set. The music generated by an algorithm is then passed through the ANN, which scores it. These artificial critics are in themselves not completely accurate and have accuracy of 85-90% when tested by human critics.
    b. Heuristic based: Many have defined fitness and performance of their results by their closeness (Euclidean or some other metric) to a corpus of existing compositions. Others have judged the originality of their music by taking the distance of the generated melody from the training data used to generate it. A combination of different such parameters then, can be combined to define performance of such an algorithm.
    c. Rule based: An expert system created based on a specific genre (jazz music, for eg.) and the rules related to it (balance of tone, sticking to a scale, novelty, etc.) can be used to evaluate and train the algorithms in use. (Spector and Alpern, 1994)

In our study, we used human critics to evaluate the results of our algorithms and they provided performance measures based on 3 attributes- RHYTHM, MELODY and DIVERSITY.

# DATASETS AND PREPROCESSING

## Datasets:

Two datasets were used as learning corpuses for all the three algorithms

- http://www.piano-midi.de/chopin.htm- This website contains MIDI files of songs composed by the famous classical musician Chopin.
- http://www.piano-midi.de/mozart.htm- This website contains MIDI files of songs composed by the famous classical musician Mozart.

Dataset description and reason for choosing them:

Both the datasets contain songs that are monophonic- they have tracks that are solely played on the piano. These are ideal for our algorithms as they are more fundamental and have less chances of leading to chaotic melodies.
There are 24 tracks of Chopin and 21 tracks of Mozart that are learnt from by the algorithms.

## Pre-processing:

The datasets did not require pre-processing as all the songs were in the required format (MIDI) and were free of distortions and artefacts. All the data points were ready to use.

# IMPLEMENTATION ISSUES AND FLOWCHART

**For Genetic and Markov:**

One major implementation issue was reading the MIDI files in an appropriate form, to process the attributes of the song present in the files. A thorough understanding of the MIDI files and the nuances of classical music was required for us to implement, which in turn required an understanding of, at least, basic music theory. In order to program the machine to make good music, we had to learn how to interpret the midi files.

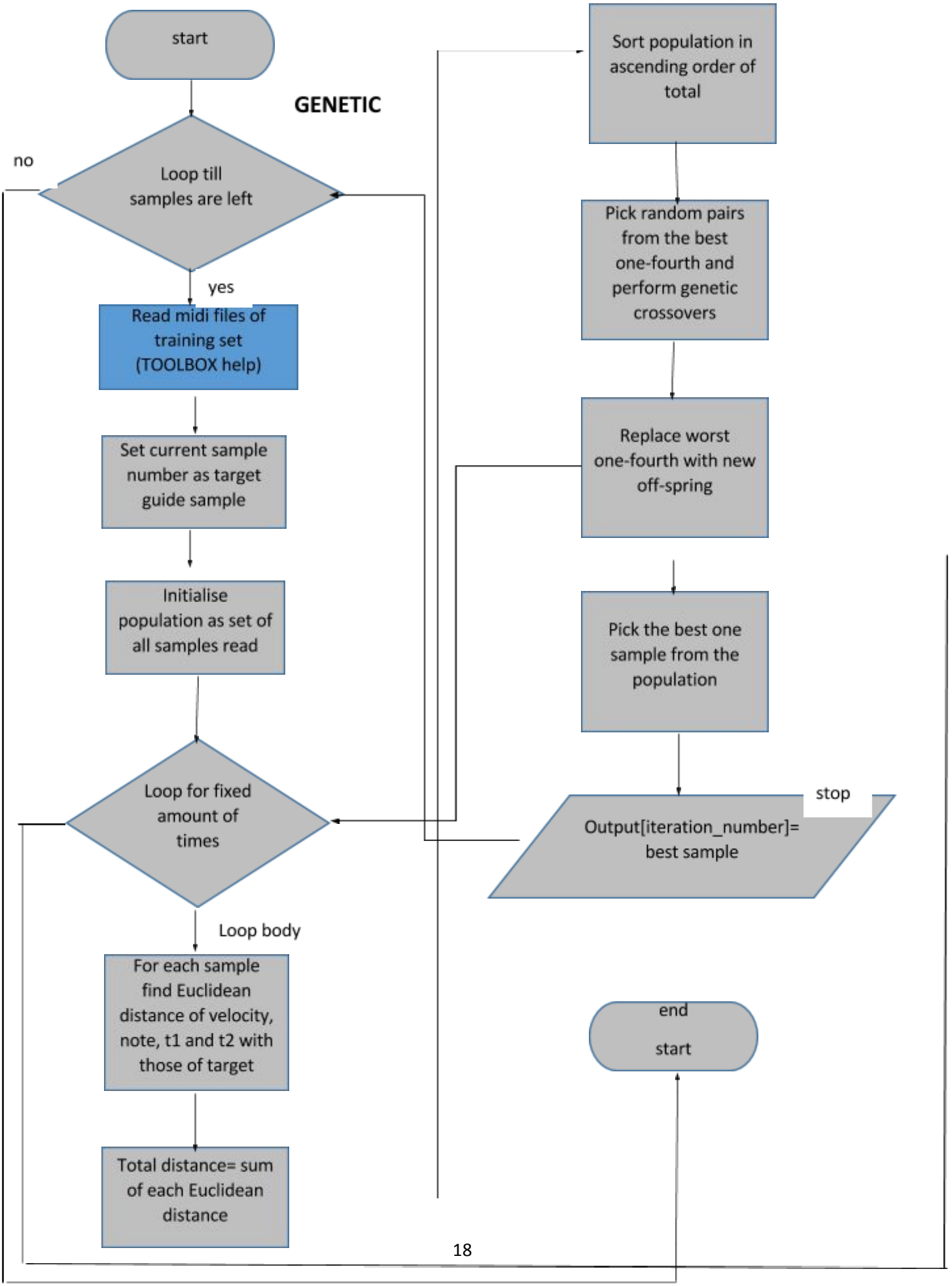
**For ANNs:**

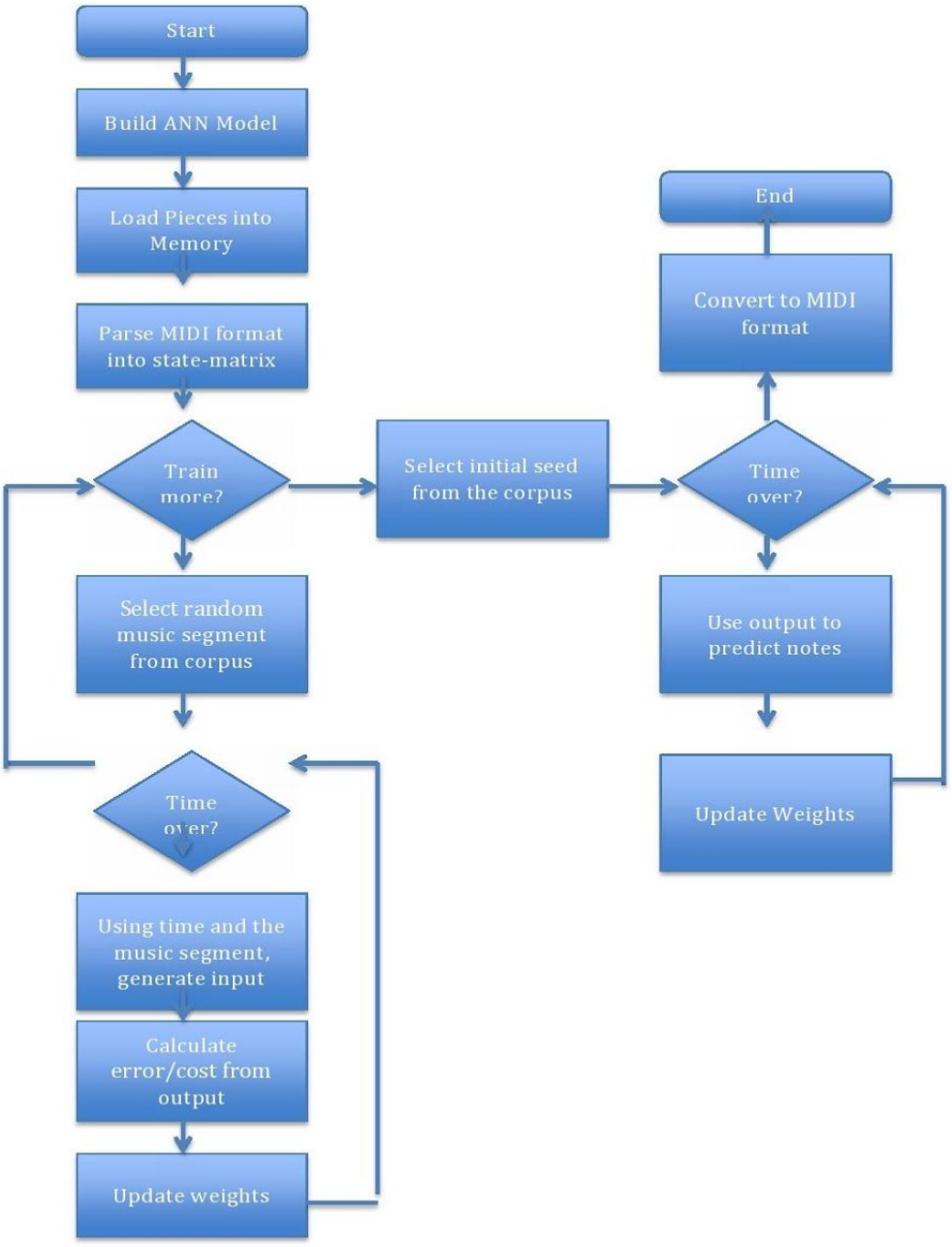
- In the literature survey, most methods involving ANNs were usually coupled with other techniques as well. As a result only very old and primitive techniques were described using pure neural networks.
- When I tried simple feed-forward networks (using the Matlab toolbox), there were implementation problems like – the network can't learn from what has been played previously.
- As a result, I tried to use a recurrent network. But again, it had problems like – it can only play one note at a time. To do this, it had to have various RNNs in parallel for each note.
- The Matlab toolbox didn't support this level of customization of the ANN and as a result, I was forced to shift platform. I found that Theano was a popular python based library for Neural Networks.
- Having had little experience with python, I looked for repositories to help and came across a blog post where someone had designed a complicated ANN for musical composition. It took care of all of the problems I had in my ANN design and even more that I hadn't considered.
- It is this code that I have submitted. I have, instead of trying to create a network on my own, tried to understand this implementation.

# Flowcharts:

1) **Genetic**
2) **ANN**
3) **Markov**

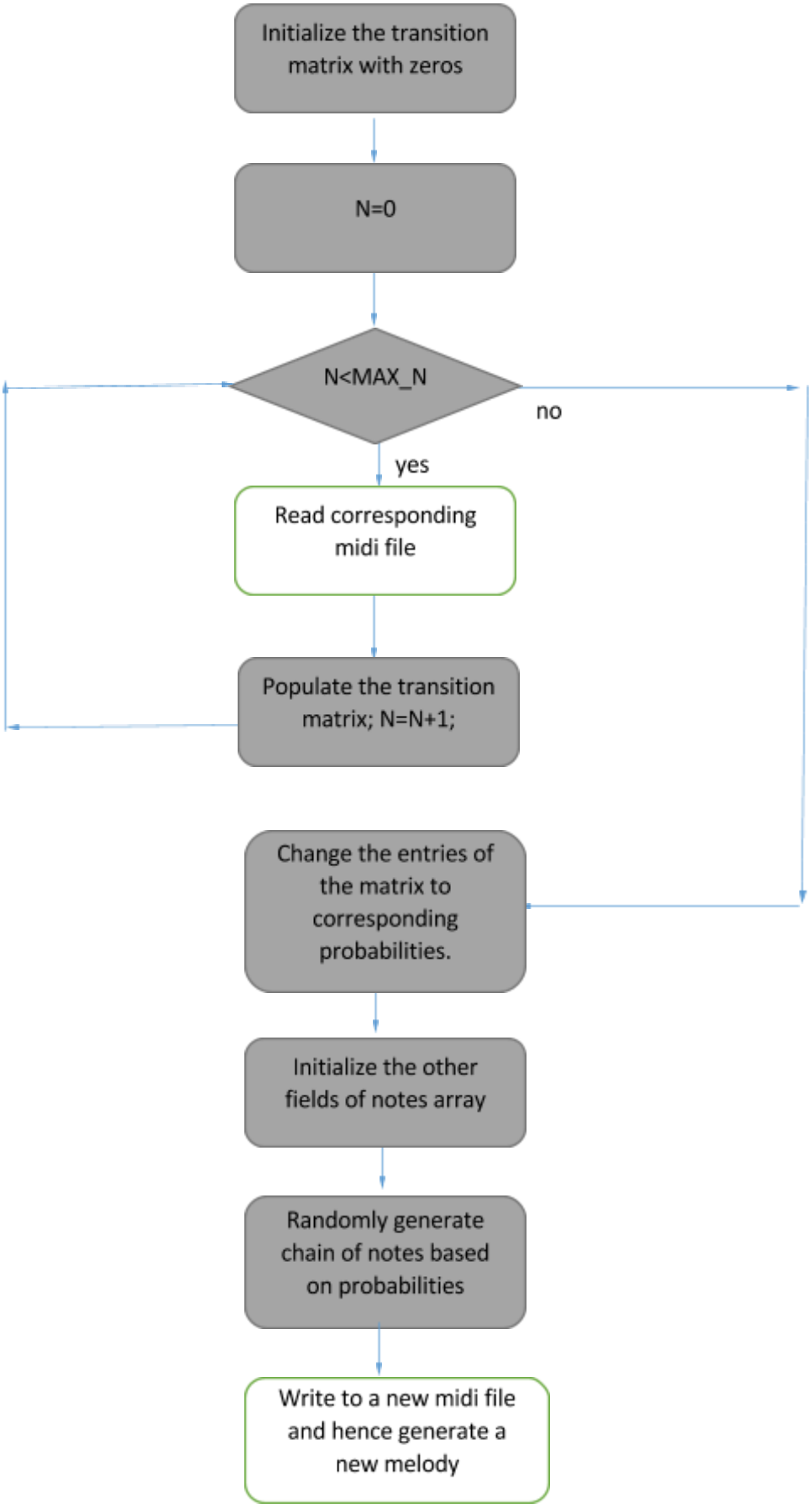
In the above order in the next 3 pages.


start
GENETIC
Loop till samples are left
no
yes
Read midi files of training set (TOOLBOX help)
Set current sample number as target guide sample
Initialise population as set of all samples read
Loop for fixed amount of times
Loop body
For each sample find Euclidean distance of velocity, note, t1 and t2 with those of target
Total distance= sum of each Euclidean distance
Sort population in ascending order of total
Pick random pairs from the best one-fourth and perform genetic crossovers
Replace worst one-fourth with new off-spring
Pick the best one sample from the population
stop
Output[iteration_number]= best sample
end
start




**ANN Flowchart**

**MARKOV**


Initialize the transition matrix with zeros
N=0
N<MAX_N
no
yes
Read corresponding midi file
Populate the transition matrix; N=N+1;
Change the entries of the matrix to corresponding probabilities.
Initialize the other fields of notes array
Randomly generate chain of notes based on probabilities
Write to a new midi file and hence generate a new melody

# DEVELOPMENT ENVIRONMENT AND SYSTEM DETAILS

For the Markov Chain algorithm and Genetic Algorithm the IDE requirement was MATLAB 2014a,

Genetic Algorithm:

- MATLAB 2014a IDE
- MIDI toolbox –
    - Source: https://github.com/kts/matlab-midi/tree/master/src
    - http://kenschutte.com/midi - toolbox used
- System requirements- Platform that can run the above version of Matlab. Recommended- Windows 7 or later.

Markov chain Algorithm:

- MATLAB 2014a IDE
- MIDI toolbox –
    - Source: https://github.com/kts/matlab-midi/tree/master/src
    - http://kenschutte.com/midi - toolbox used
- System requirements- Platform that can run the above version of Matlab. Recommended- Windows 7 or later.

Artificial Neural Network algorithm:

- Python 2.7 and IDE to work in.
- Libraries: numpy, scipy, theano, theano-lstm python-midi, pickle.
- This method is very compute intensive and so a high-powered system is suggested.

# RESULTS AND DISCUSSION:

**Human Rating Parameters:**

- Rhythm – Musical regularity
- Melody – Aesthetic appeal
- Diversity – Dissimilarity to training data

**Chopin:**

Neural Networks:

| Song #1 | Rhythm | Melody | Diversity |
|---|---|---|---|
| 1 | 4 | 3 | 4 |
| 2 | 3 | 3 | 3 |
| 3 | 3 | 3 | 4 |
| 4 | 4 | 4 | 5 |
| 5 | 3 | 4 | 4 |
| Average | 3.4 | 3.4 | 4 |

Markov Chain:

| Song # | Rhythm | Melody | Diversity |
|---|---|---|---|
| 1 | 3 | 4 | 3 |
| 2 | 4 | 2 | 3 |
| 3 | 3 | 3 | 4 |
| 4 | 2 | 3 | 4 |
| 5 | 4 | 3 | 2 |
| Average | 3.2 | 3 | 3.2 |

Genetic Algorithm:

| Song # | Rhythm | Melody | Diversity |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

| | | | |
|---|---|---|---|
| 2 | 4 | 4 | 4 |
| 3 | 4 | 3 | 4 |
| 4 | 3 | 3 | 3 |
| 5 | 3 | 4 | 3 |
| Average | 3.2 | 3.4 | 3.6 |

**Mozart:**

Neural Networks:

| Song #1 | Rhythm | Melody | Diversity |
|---|---|---|---|
| 1 | 4 | 2 | 3 |
| 2 | 4 | 4 | 3 |
| 3 | 4 | 3 | 4 |
| 4 | 3 | 4 | 4 |
| 5 | 4 | 2 | 5 |
| Average | 3.8 | 3 | 3.8 |

Markov Chain:

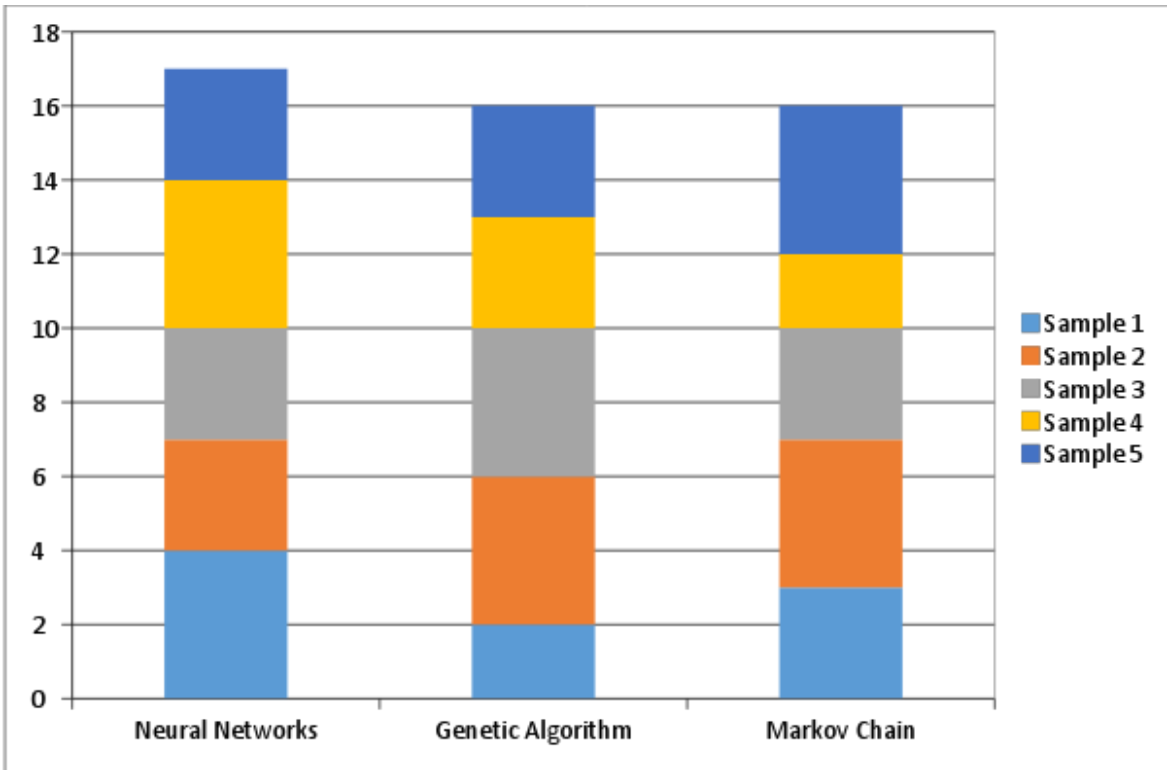
| Song # | Rhythm | Melody | Diversity |
|---|---|---|---|
| 1 | 4 | 5 | 3 |
| 2 | 4 | 4 | 5 |
| 3 | 3 | 2 | 3 |
| 4 | 5 | 3 | 2 |
| 5 | 4 | 5 | 3 |
| Average | 4 | 3.8 | 3.2 |

Genetic Algorithm

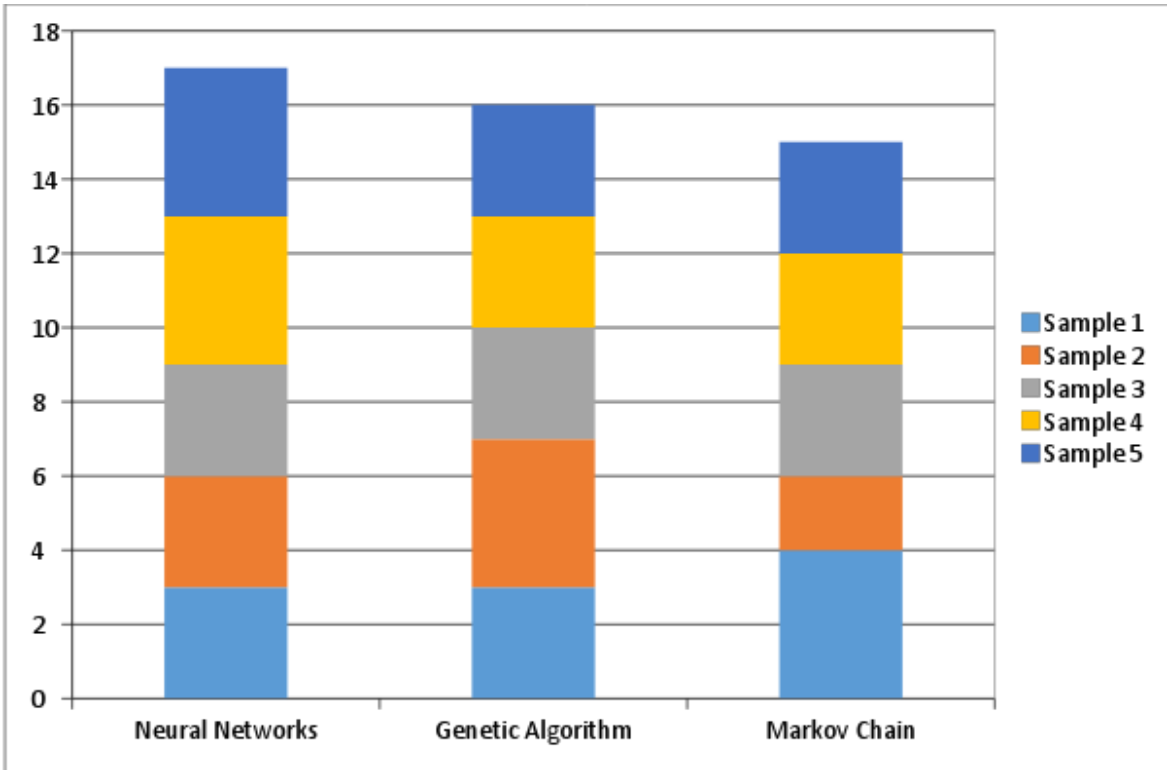
| Song # | Rhythm | Melody | Diversity |
|---|---|---|---|
| 1 | 4 | 4 | 1 |
| 2 | 3 | 3 | 3 |

| | | | |
|---|---|---|---|
| 3 | 4 | 3 | 3 |
| 4 | 5 | 4 | 2 |
| 5 | 3 | 3 | 3 |
| Average | 3.8 | 3.4 | 2.4 |

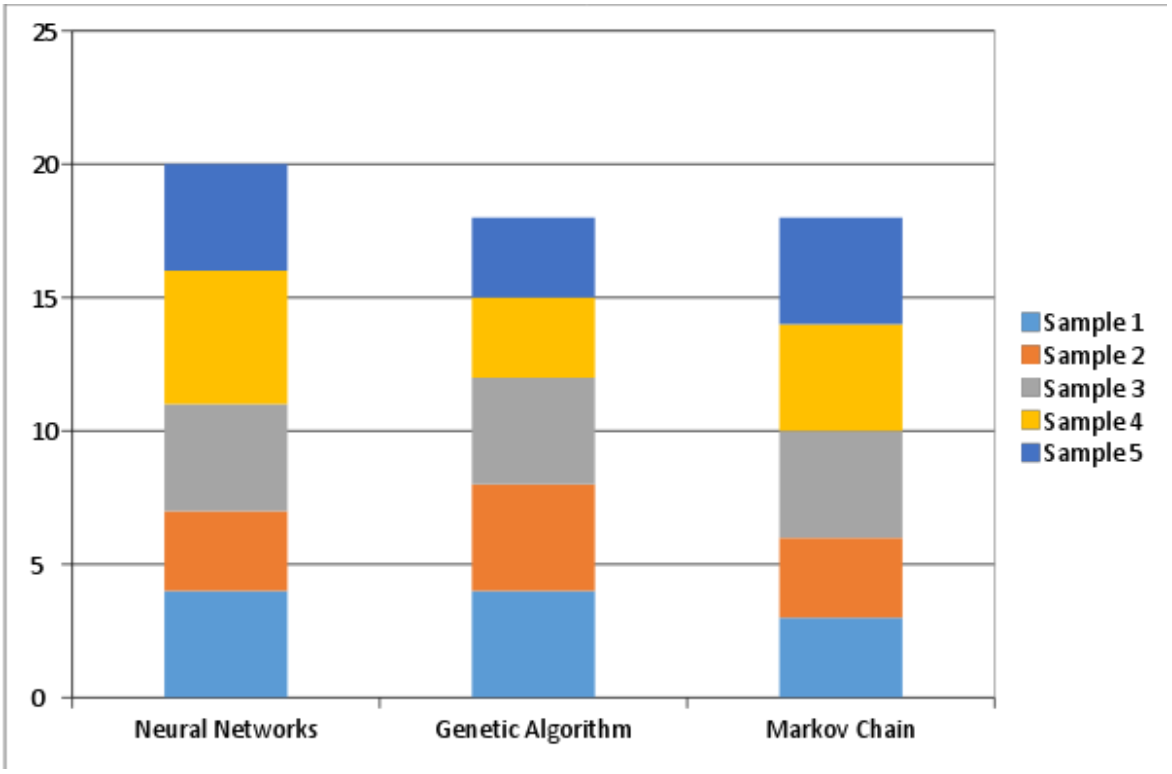
Chopin – Rhythm



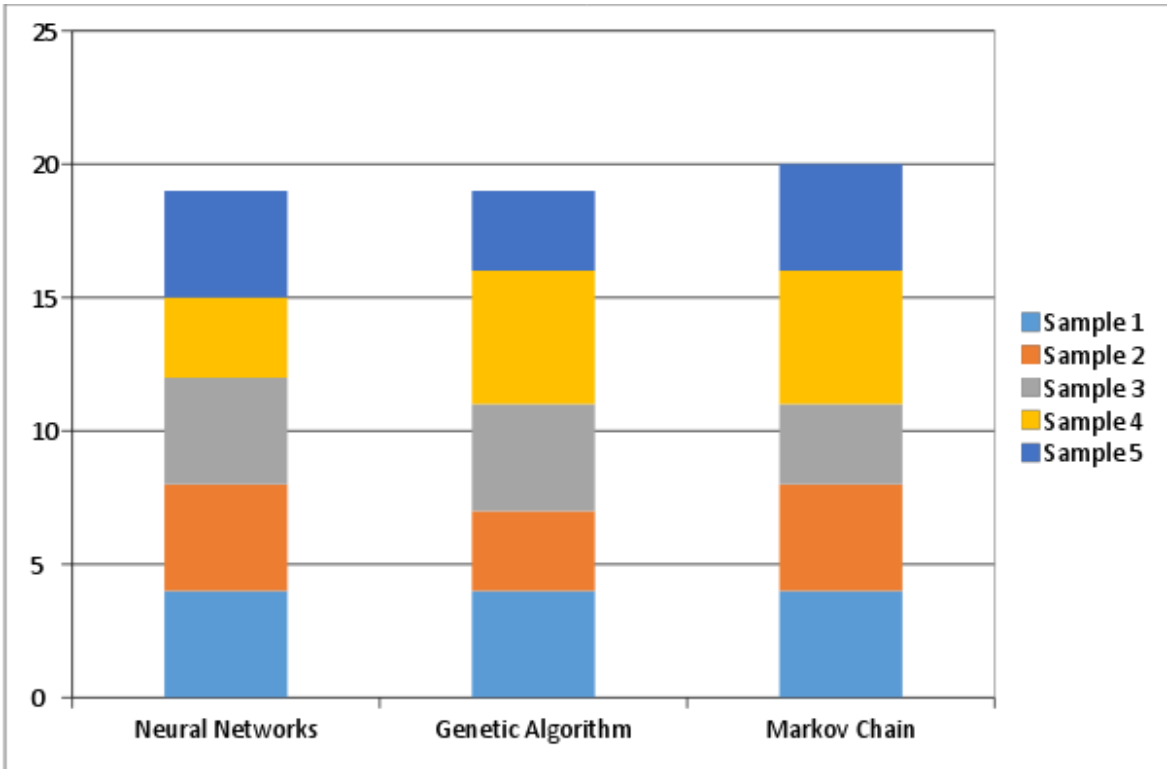

Chopin – Melody



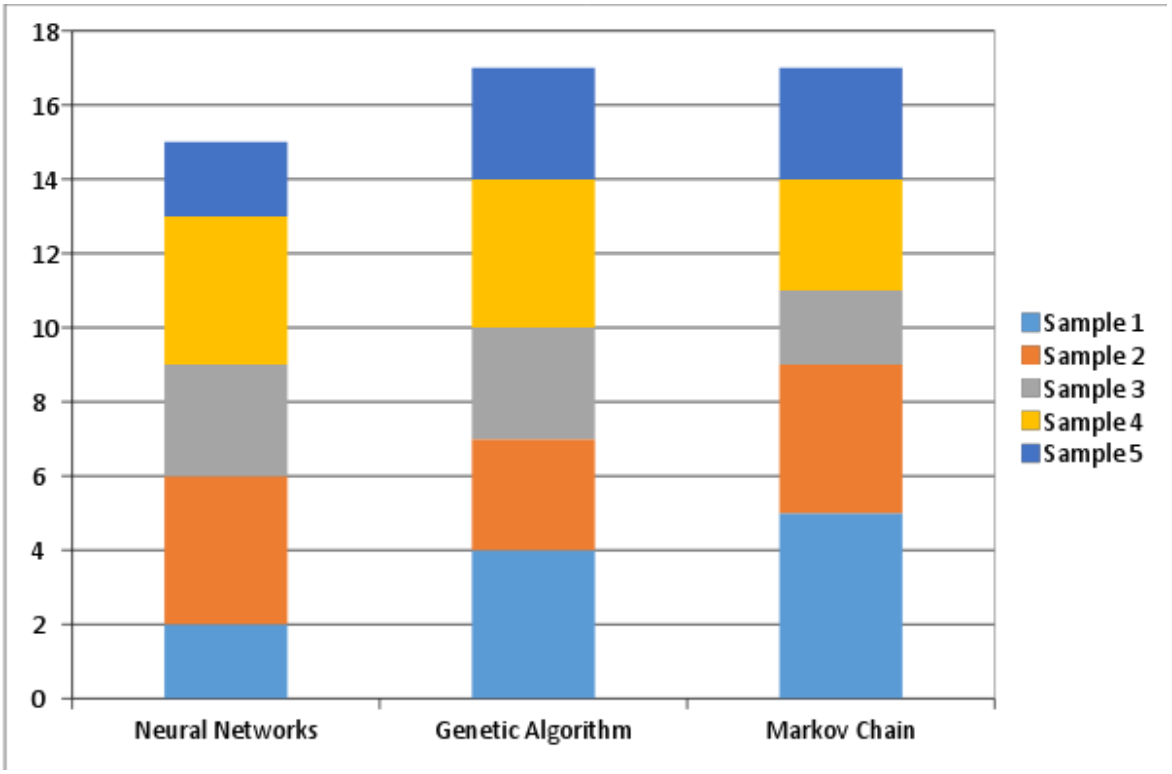
Chopin – Diversity



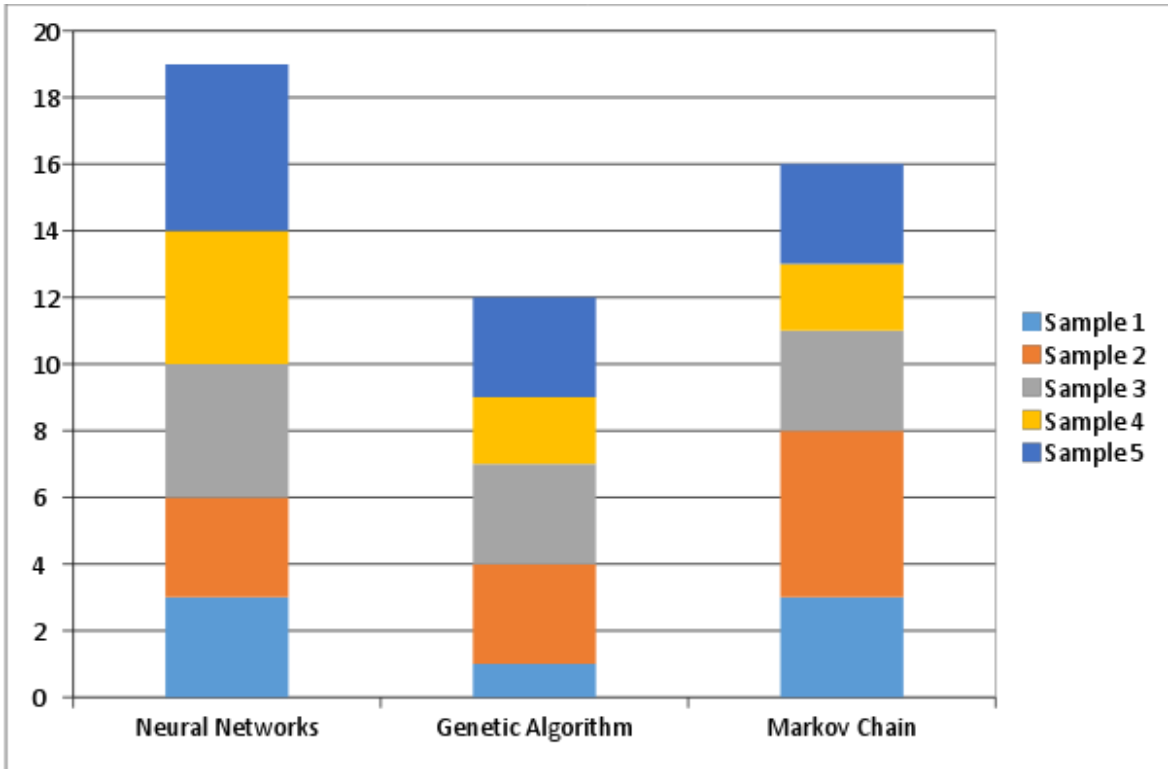

Mozart – Rhythm




Mozart - Melody



Mozart - Diversity




**Conclusion of results:**

- The **markov chain** algorithm is fundamentally based on the random selection of notes favoured by probability to play next. This will hence produce erratic results every time, sometimes leading to a good melody and sometimes maybe not. This can be seen in its graphs of chopin and mozart. It is observed that it performs better for mozart. One can never predict the performance of the algorithm.
- **Genetic algorithm** is heavily dependent on the function chosen for fitness measurement. In this implementation, each of the tracks are used in turn as "guide" tracks and the chromosomes that achieve similar properties to the guide are rewarded. Genetic algorithm showed fair performances in both datasets.
- **Neural Networks** are, in their most rudimentary form, unsuitable for music composition. Their strength lies in the ability to configure the network structure. In this implementation, a number of heuristic rules have been implemented in the design which make the music more pleasant to listen to. This can be seen in the music that has been generated and its likeability. While the algorithm is complex and takes a

long time to train and run, it is undoubtedly versatile and can accommodate other algorithms in its design.

# REFERENCES


1. Todd, P. M., & Loy, D. G. (1991). *Music and Connectionism*. The MIT Press, Cambridge.
2. Fernández, J. D., & Vico, F. (2013). AI methods in algorithmic composition: a comprehensive survey. *Journal of Artificial Intelligence Research*, *48*(1), 513-582.
3. Tokui, N., & Iba, H. (2000, December). Music composition with interactive evolutionary computation. In *Proceedings of the 3rd international conference on generative art* (Vol. 17, No. 2, pp. 215-226).
4. Manaris, B., Roos, P., Machado, P., Krehbiel, D., Pellicoro, L., & Romero, J. (2007, July). A corpus-based hybrid approach to music analysis and composition. In *Proceedings of the National Conference on Artificial Intelligence*(Vol. 22, No. 1, p. 839). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
5. Spector, L., & Alpern, A. (1994) Criticism, culture, and the automatic generation of artworks. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI94) (pp. 3-8). Menlo Park, CA, and Cambridge, MA: AAAI Press/MIT Press.
6. Miranda, E. R., & Biles, J. A. (Eds.). (2007). *Evolutionary computer music*. Springer-Verlag London.
7. Ball, P. (2005). Making music by numbers online. *Nature News Online* (http://dx.doi.org/10.1038/050919-14).
8. http://futurehumanevolution.com/artificial-intelligence-future-human-evolution/artificial-neural-networks
9. http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7