



DEGREE PROJECT, IN SIGNAL PROCESSING , SECOND LEVEL  
*STOCKHOLM, SWEDEN 2015*

# Automatic Melody Generation

THAYABARAN KATHIRESAN

KTH ROYAL INSTITUTE OF TECHNOLOGY  
SCHOOL OF ELECTRICAL ENGINEERING



ROYAL INSTITUTE  
OF TECHNOLOGY



# Automatic Melody Generation

Thayabaran.Kathiresan

22 June 2015



Master degree of European Master of Research on Information and  
Communication Technologies (MERIT)

Academic Supervisor : MATS BENGTTSSON  
Associate Professor- KTH Royal Institute of Technology  
mats.bengtsson@ee.kth.se

Institute Supervisor : ESTEFANIA CANO  
Researcher - Fraunhofer IDMT  
cano@idmt.fraunhofer.de

# Acknowledgments

*First and foremost, I'd like to give heartfelt thanks to my academic supervisor Mats Bengtsson for showing a special interest in my thesis and my institute supervisor Estefania Cano for providing me such a wonderful opportunity to work in Fraunhofer IDMT, Germany.*

*Thanks to Semantic Music Technologies team in Fraunhofer IDMT who provided constant support and valuable insights on my work.*

*Special thanks to Professor Climent Nadeu from Universitat Politècnica de Catalunya, Spain for being a valuable academic tutor throughout my master studies. I would also like to thank KTH Royal Institute of Technology, Sweden for supporting me through an Erasmus placement grant.*

*And finally I'd like to thank all my friends especially, Winnie for encouraging and never letting me down in my hard times.*

# List of Abbreviations

*AI* Artificial Intelligence

*ANN* Artificial Neural Networks

*CBR* Creation By Refinement

*CD* Categorical Distributions

*CDF* Cumulative Distribution Function

*CONCERT* CONnectionist Composer of ERudite Tunes.

*CSPs* Constraint Satisfaction Problems

*DNN* Deep Neural Network

*FFNN* Feed Forward Neural Networks

*HMM* Hidden Markov Model

*MIDI* Musical Instrument Digital Interface

*MIR* Music Information Retrieval

*MM* Markov Model

*MRS* Music Recommendation Systems

*MUSHRA* MUltiple Stimuli with Hidden Reference and Anchor

*NND* Next Note Distributed

*NNL* Next Note Local

*RNN* Recurrent Neural Network

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Research Questions and Project Goal . . . . .	5
1.2	Methodology . . . . .	5
1.3	Ethical Considerations . . . . .	6
<b>2</b>	<b>State of the Art</b>	<b>7</b>
2.1	Algorithmic Music Composition . . . . .	7
2.1.1	L-Systems . . . . .	7
2.1.2	Neural Networks . . . . .	12
2.1.3	Markov Model . . . . .	17
2.1.4	Parametric Extension . . . . .	20
2.1.5	Hierarchical Grammars . . . . .	23
<b>3</b>	<b>Proposed Method</b>	<b>25</b>
3.1	Data . . . . .	25
3.1.1	MIDI Features Extraction . . . . .	26
3.2	Melody Generation . . . . .	27
3.2.1	Constraint Satisfaction Problems . . . . .	27
3.2.2	MM with CSPs . . . . .	30
3.2.3	Melody Constraints . . . . .	32
3.3	Rhythm Generation . . . . .	38
3.4	Melody and Rhythm Generation using HMM . . . . .	41
3.4.1	Categorical Distribution . . . . .	41
3.4.2	HMM with Categorical Distribution in Generation . . . . .	43
<b>4</b>	<b>Results</b>	<b>46</b>
<b>5</b>	<b>Conclusion</b>	<b>52</b>
<b>6</b>	<b>Appendix</b>	<b>55</b>
6.1	MIDI ASCII Note Table . . . . .	55
6.2	Notes Name and Position in the Staff . . . . .	55
6.3	Music Symbols and its Property . . . . .	56
6.4	Musical Score of <i>Bee Bee</i> Folk Song . . . . .	56
6.5	Melody Generated by HMM with CD Model in the Style of <i>Green Mountain</i> (Taiwanese Folk Song) . . . . .	57

# 1 Introduction

Music composition is a fine art that exists from the origin of human race in various forms. Music acts as a universal language uniting people's emotions across cultures. As music is an important aspect of daily life:

“ Whenever humans come together for any reason, music is there: weddings, funerals, graduation from college, men marching off to war, stadium sporting events, a night on the town, prayer, a romantic dinner, mothers rocking their infants to sleep, music is a part of the fabric of everyday life.”

— DANIEL J. LEVITIN, *This Is Your Brain on Music*

The richness of a culture is measured by various factors, among them, music plays a key role. The importance of musicians and their work has always been appreciated despite language or culture. The creativity of composing music involves various factors that composers possess. However music communicates certain things that cannot be conveyed by other means. Also emotions are shared effectively to listeners by different perceptual features such as energy, rhythm, temporal, spectral, and harmony [2]. Music cannot be evaluated by their semantic meanings [11] so, it is very difficult to measure its correctness. Having understood the importance of a music, the music industry is getting more and more advanced day-by-day. The technologies includes music synthesis, recording, sound production, audio quality and many more. Advancement in music industry helps engineers and musicians to work together to come up with cutting edge music applications such as Music Recommendation Systems (MRS) [2]. It enables people to enjoy music where ever they want.

The field of Artificial Intelligence (AI) is advancing with the technology development [12]. Music is a challenging area for AI to solve many real time problems like algorithmic composition [12] and music information retrieval (MIR). The usage of AI in music composition is also referred to as *the synthesis of works of art* which could be partial or fully automated [12]. Most of the algorithms in music composition are biologically inspired. Bio-inspired techniques often involve the method of specifying a set of simple rules, a set of simple organisms which adhere to those rules, and a method of iteratively applying those rules<sup>1</sup>. The outcome of these iterations are

---

<sup>1</sup>Bio-inspired techniques are closely related to the field of AI, as many of its pursuits can be linked to machine learning. It relies heavily on the fields of biology, computer science and mathematics.

interpreted as musical elements or notation. This is how bio-inspired algorithms are used in a music composition.

## 1.1 Research Questions and Project Goal

Questions that this thesis addresses:

- How can we generate the melodies using AI algorithm with the style similar to human composition?
- How can we include our musical constraints along with the melodies being generated (in reference style) using AI algorithm?
- How the user constraints influence the *pleasantness* of the generated melodies?
- How to make a melody generation model which handles both notes and duration together?

The primary objective of this thesis is to design an AI algorithm which should generate melodies in the given reference style which should satisfy the user constraints. To evaluate the significance of the melodies generated with different constraints, a comparison test needs to be done. A standalone melody generation model which is able to handle both notes and duration together has to be designed. The designed model should generate melodies which should remain in the reference melody style.

## 1.2 Methodology

The methodology selected to achieve the thesis goals are by exploring the most common techniques for automatic music composition which is explained in section 2. Section 3 investigates a new combination of existing techniques. This thesis also proposes a new model in algorithmic composition which is explained in section 3.4. Section 4 compares the result of the proposed models and section 5 discusses the conclusion and future work.

### 1.3 Ethical and Social Considerations

Texts belonging to other authors that have been used in any part of this thesis have been fully referenced in an appropriate format.

The questionnaire has been designed to collect information directly related to the research questions, and no private or personal questions were asked from participants who participated listening test.

The protection of the privacy of research participants has been ensured.

The thesis work is going to be used in music training application to compose novel melodies. It may create an impact over the music practitioner's learning skills through the training content. The melodies generated by the AI algorithm are not as much natural as human composition in terms of creativity and style. There is a risk that the music practitioners those who are get trained through the algorithm composition may compose music similar to machine composition.

Human compositions were used to train the AI algorithm, which would then generate new melodies, based on the training data, that are statistically similar to the reference melodies. There exists, however, a chance to reproduce the original composition which was done by human composer or it may compose a melody that might be a future work of the human composer.

## 2 State of the Art

### 2.1 Algorithmic Music Composition

From a computational view, melody composition is described by the chronology of notes with constraints and rhythmic patterns [12]. From an artistic view, melody composition is an outcome of human creativity. In computer composition, we attempt to replicate the artistic approach with designed rules that characterize the algorithmic function [16]. In AI, numerous techniques for algorithmic composition have been proposed. Every technique has its own merits and limitations, the following sections explore some of the most important algorithmic composition techniques.

#### 2.1.1 L-Systems

Aristid Lindenmayer, Hungarian botanist and theoretical biologist introduced a string rewriting grammar in 1968 which he named as L-Systems were initially presented “as a theoretical framework for studying the development of simple multicellular organisms” [22]. L-Systems have traditionally been used as a popular method for modelling of space filling curves, biological systems and morphogenesis. L-System helps in representing complex patterns that have some degree of repetition, self-similarity or developmental complexity. Botanist and computer scientist used L-Systems to model and study the growth of living organisms. Later, the representation of a plant growth in L-Systems helped in modeling them using computer graphics to visualize the development process. Graphical representations of L-Systems were first published in 1974 by Frijters and Lindenmayer, and by Hogeweg and Hopper [22]. L-systems are not limited to plants and trees, but also included a wide range of fractal curves such as description of *kolam* patterns (an art form from Southern India) [22].

A L-system (or Lindenmayer system) is a formal rewriting grammar, i.e. a set of rules and symbols. The essential difference with Chomsky grammars<sup>2</sup> is that the rewriting is parallel not sequential. That means that in every iteration all the symbols of the string are simultaneously replaced [21]. An application of L-systems to the algorithmic generation of musical scores was introduced by Prusinkiewicz in 1986 [21]. By defining the basic rules of the musical elements, L-Systems create a long sequence of symbols that could be interpreted as notes, velocity, pitch and other musical parameters. Prusinkiewicz proposed an interpretation system that

---

<sup>2</sup>In the 1950s Noam Chomsky applied string rewriting grammar to describe the syntax of natural languages and it is well known as Chomsky grammars.

works by performing a spatial mapping of the L-system output. The generation process includes 2-dimensional graph using the turtle graphics<sup>3</sup>.

The idea behind the turtle interpretation is defining the position of the turtle in Cartesian coordinate system  $(x,y)$  and angle  $\alpha$ , gives the direction in which turtle is facing. So, the turtle is represented by a triplet  $(x,y,\alpha)$ .

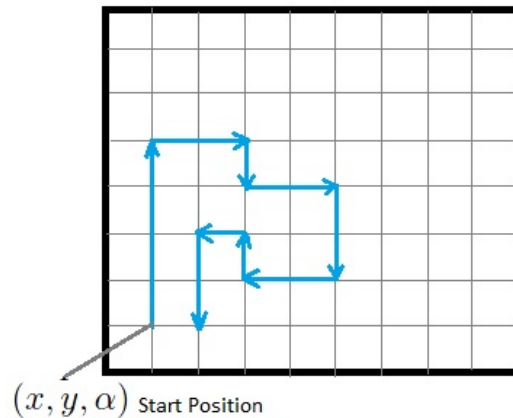


Figure 1: Turtle movement in the Cartesian coordinate system for a given command **FFFF-FF-F+FF-FF-FF-F+F+FF**

To move the turtle in a coordinate system we need to give the step size  $d$  and an incremental angle  $\delta$ . The movement also needs a set of symbols such as F, f, - and +. Each symbol performs its own operation as given below,

Where,

- F Move forward with step size  $d$  by drawing a line.
- f Move forward with step size  $d$  without drawing a line.
- + Turn left by incremental angle  $\delta$  from its actual position  $\alpha$ .
- Turn right by incremental angle  $\delta$  from its actual position  $\alpha$ .

Figure 1 shows the turtle movement in the Cartesian coordinate system for a sequence of commands with an assumption of incremental angle  $\delta=90^\circ$  and  $\alpha =$  Turtle facing upward in start position  $(x,y)$ .

---

<sup>3</sup>Turtle graphics is a term in computer graphics for a method of programming vector graphics using a relative cursor (the “turtle”) upon a Cartesian plane. Turtle graphics is a key feature of the Logo programming language.

L-Systems also perform a string rewriting. String rewriting is an important feature of an L-Systems and it is performed on every iteration by replacing a symbols simultaneously. This process is very similar to the Koch<sup>4</sup> construction. L-Systems need a production rule  $p$  with axiom or starting string  $\omega$  to perform a string rewriting process.

For an example,

$$\omega = F-F-F-F$$

$$p = F \rightarrow F \ F + F + FF \ F \ F + F$$

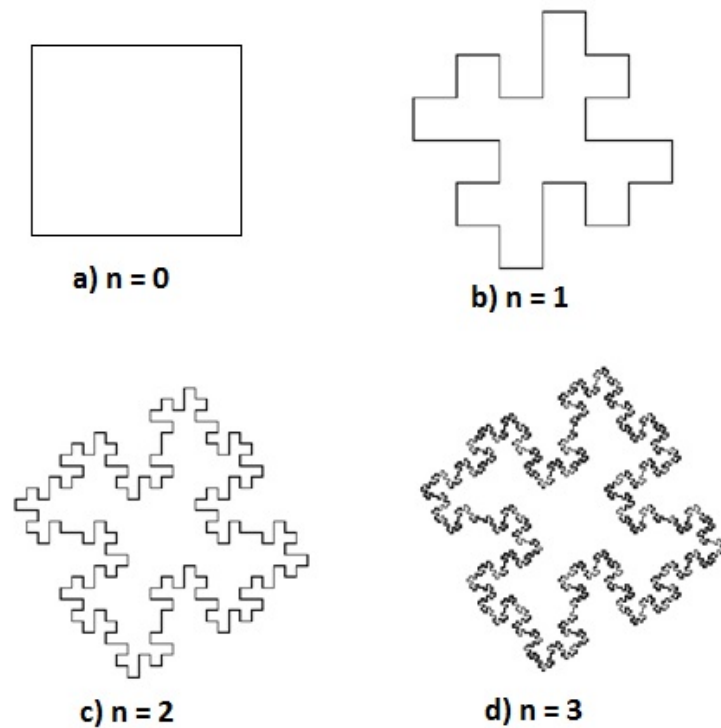


Figure 2: Generation of Koch curves using L-System string rewriting technique

For every iteration ( $n$ ) the string symbols are replaced as per the production rules. Figure 2 a) is an axiom ( $\omega$ ) which leads to the structure shown in Figure 2 b) on the first iteration. After the second iteration ( $n=2$ ), the structure changes are shown in Figure 2 c). The structure in Figure 2 d) (obtained in third iteration  $n=3$ ) is also called as quadratic Koch island. For the development of quadratic Koch island we had an assumption of incremental angle  $\delta = 90^\circ$ .

<sup>4</sup>Koch curve is a fractal curves using which Koch snowflakes can be made.

### L-Systems in Music Generation

Music synthesis involves repetition of some basic sequence of notes or at some higher levels of structural grouping [11]. L-Systems provides a compact way of representing complex patterns that have some degree of repetition, self-similarity and developmental complexity [11]. Deterministic, context-free L-grammars (DOL-Systems) is a part of L-Systems that is represented by three parameters, namely,  $\Sigma$ , P and  $\alpha$ , where:

$\Sigma$  - Alphabet of the system or universal set of the system.

$\alpha$  - Axiom or starting string ( $\alpha$  should not be empty).

P - Production rules

For example, the string regeneration using DOL-Systems in music synthesis is given below,

DOL-Systems:  $\langle \Sigma, P, \alpha \rangle$

$\Sigma$ : A B C D (Strings represent the musical notes)

$\alpha$ : D A

P: (All rules applicable for every iterations)

- $A \rightarrow A C D$
- $B \rightarrow D A C$
- $C \rightarrow B B A$
- $D \rightarrow C B$

Iteration	Strings
1	D A (Axioms)
2	C B A C D
3	B B A D A C A C D B B A C B

Table 1: String regeneration using DOL-Systems

Concatenating the strings from the top to the bottom of Table 1 gives the final sequence, “D A C B A C D B B A D A C A C D B B A C B”. These strings

can be interpreted as musical notes. The number of iterations in a string rewriting process is varied based on the required length of the melody. The generated musical elements can be played as sound using Musical Instrument Digital Interface (MIDI)<sup>5</sup>.

In 1986, P. Prusinkiewicz introduced L-Systems with turtle graphics to generate both notes and duration for musical scores [21]. He used three steps to generate musical scores:

- A string of symbols is generated by an L-system.
- This string is interpreted graphically as a sequence of commands controlling a turtle.
- The resulting figure is interpreted musically as a sequence of notes with pitch and duration determined by the coordinates of the figure segments.

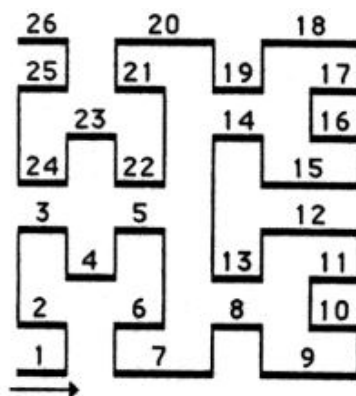


Figure 3: Traversing the Hilbert curve<sup>6</sup>, Musical score generation using L-Systems.



Figure 4: Musical score representation of Hilbert curve [19].

<sup>5</sup>MIDI is a technical standard that describes a protocol in which messages specify notation, pitch and velocity, control signals for parameters where they control sound generation and other features.

<sup>6</sup>Hilbert curve is also known as a Hilbert space-filling curve. It is a continuous fractal space-filling curve first described by the German mathematician David Hilbert in 1891.

P. Prusinkiewicz used traverse of Hilbert curve for his musical score generation. Hilbert curve's  $X$ - axis represents the note duration and  $Y$ - axis represents the pitch of the note. Figure 3 shows an example of a two dimensional Hilbert curve [19]. It is also assumed that the notes belongs to the C-major scale. Figure 4 represents the musical interpretation of a Hilbert curve produced using L-Systems. Musical score generation using L-Systems notes are not accidentally merged together. Also they are relatively complex in structure.

### 2.1.2 Neural Networks

Artificial Neural Networks (ANNs) are an important biologically inspired techniques in AI and machine learning . The activity and function of the central nervous system and brain inspired the concept of ANN. Artificial neurons are interconnected to form a network where each neuron is responsible for processing numerous inputs into a single numeric output using a simple but nonlinear function [12]. Mostly ANN is used for approximation of a function that depends on numerous inputs.

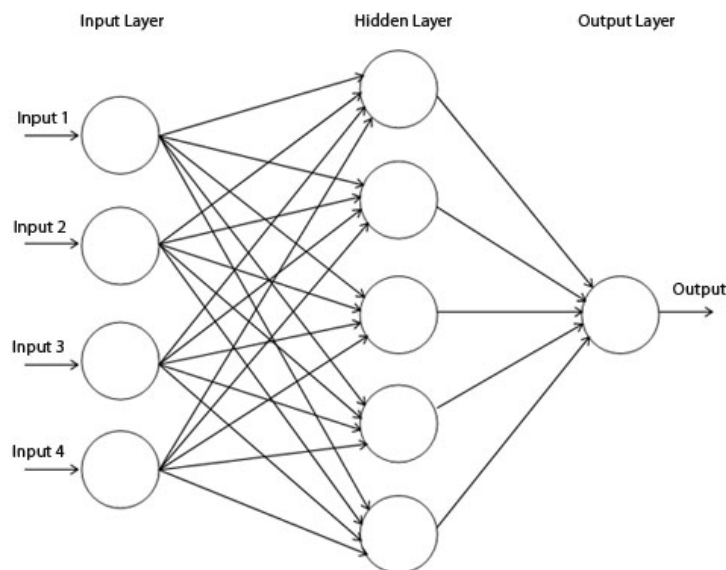


Figure 5: Three layered Feed Forward Artificial Neural Networks [1]

The structural arrangement of the neurons in the network defines the ANN type, for example, Recurrent Neural Network (RNN) and Deep Neural Network (DNN), Dynamic neural networks and Compositional pattern-producing networks. Feed Forward Neural Networks (FFNN) are the most common and simple type of ANN Figure 5 shows an architecture of FFNN. There are three layers, namely, input

layers, hidden layers, and output layers. Based on the applications and requirements, the number of nodes in a layer varies. In order to excite a neuron, an impulse is required with sufficient threshold value. The threshold value varies for each neuron in the network. Logistic functions (Equation 1) are often used in neural networks to introduce non-linearity to the model. Neural element computes a linear combination of its input signals, and applies a bounded logistic function to the result,

$$g(h) = \frac{1}{1 + e^{-\beta t}} \quad (1)$$

where  $\beta$  is a slope parameter and equation 1 is also called as log-sigmoid or sigmoid function.

To train an ANN two standard approaches are followed, namely, supervised and unsupervised learning. In *supervised learning*, both the inputs and the outputs are provided with labels. The ANN then learns (processes) the given inputs and compares its resulting outputs against the expected outputs. The difference in the comparison (bias) is then propagated back through the network, causing the system to adapt the weights of the neurons which control the network. This process continues to get a desired and satisfactory output (with minimum bias). The set of data which enables the training is called “training set”. In unsupervised training, the network is provided with inputs but not with expected outputs. The network is allowed to group the input data by selecting the best features. This is often referred to as self-organization or adaption. A corpus of musical composition is used to train (supervised) the ANN. By training ANN, we could model a system that is capable of doing recognition or generation of the musical elements or patterns. This is the basic idea behind usage of ANN in music field. Usage of ANN in music started in the 1980s for analyzing the music composition in cognition theory of music [12]. Later in 1989, Todd Lewis [23] implemented a model using three-layered recurrent network which produced a temporal sequence of monophonic melodies. Figure 6 shows the three layered recurrent neural network and its output gives an absolute pitch for monophonic melodies.

The reason behind the selection of recurrent ANN is that feed-forward network does not contain a mechanism to remember past history. This makes it unsuitable for the music generation because of repetitive rhythmic patterns and pitches are important elements in composition [5]. These characteristics of music composition makes RNN a better model for designing melody generation algorithm. Also

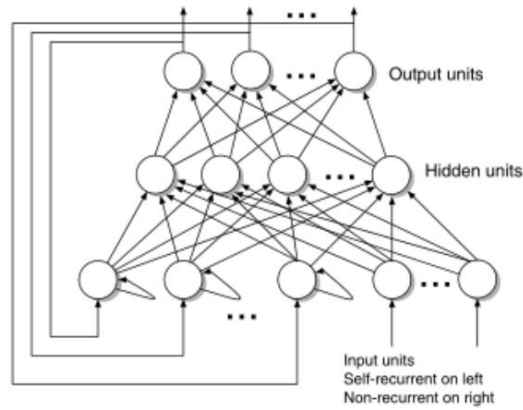


Figure 6: RNN shows the input and output relation by feed back [13]

in 1989, Duff published [7] another method of music composition using ANN. His model produced a relative pitch unlike Todd's absolute pitch [12]. In 1991 Lewis [23] developed ANN framework: *Creation By Refinement* (CBR). CBR is a neural network developed specially for the artificial creativity for music composition. A FFNN is trained with a set of patterns ranging from random to very good music, associating each pattern with a (possibly) multidimensional musicality score.

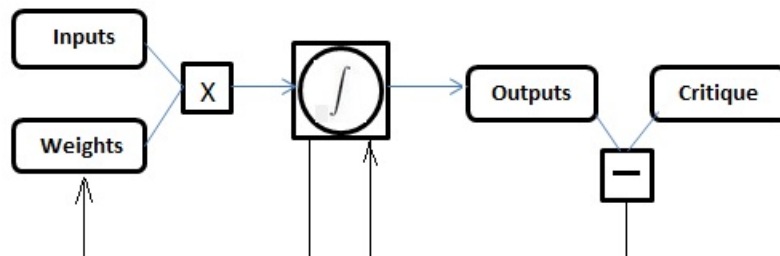


Figure 7: Creation by refinement training using gradient descent method [23]

A standard gradient descent learning (supervised) is used to train CBR, which acts as a “music critic” and then followed by a creation phase. The idea behind the CBR is perturbing the learned weights of the neurons randomly to generate novel patterns. The approach of perturbing the trained network will alter the structure of ANN but still novel patterns are produced at the expense of degrading the network structure. Figure 7 shows the simplified schematic of CBR in training. A set of musical patterns fed as input to CBR and also a set of users critique are presented as a target [23]. The weights learned from the input are compared with a target. The bias is given back to the system as an error to refine the weights. The gradient

descent mechanism plays a role in adjusting a error to match the weights suitable for the target. The critiques are expressed numerically it could be a sequence of pitches as musical (1), non musical (0), and not very good musical (0.5). The dimension of a critique may increase based on the users interest to achieve the target. Once the training phase is completed, next process is creation in CBR.

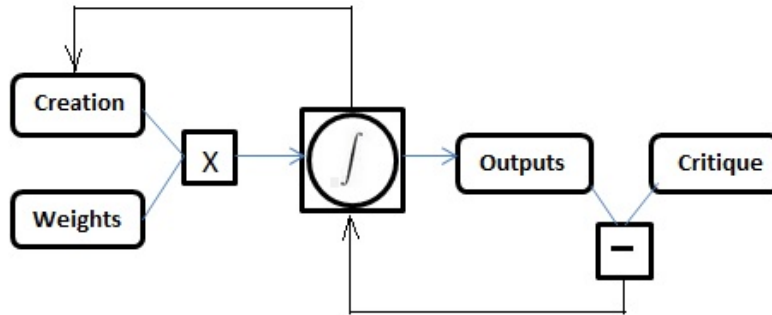


Figure 8: CBR creation using gradient descent method [23]

In creation phase, the structure used in the training phase shown in Figure 7 is similar but the input block is renamed as creation shown in Figure 8. To generate a novel pattern using a learned CBR and also to satisfy the users critique a second gradient descent approach is used. In creation phase the weights learned in the training are changed to get a novel patterns. In this process the structure of the network will get altered but it wont disturb the creation process. CBR could generate new melodies but it failed to imitate the existing style of the music [23]. So the synthesis of CBR is more novel than imitative of the learned style.

In 1994 Michael C. Mozer [17] introduced a network that compose melodies with harmonic accompaniment called CONCERT. CONCERT is an acronym for CONNECTIONIST Composer of ERrudite Tunes. It is a form of recurrent auto-predictive neural networks. CONCERT is trained by means of two ways, first by training with a set of sample melodies in which the network extracts the regularities of note and phrase progressions. The first way of training helps in defining *the melodic and stylistic constraints*. The second way of training is by representations of pitch, duration, and harmonic structure that are based on psychological studies of human perception.

The second way of training helps CONCERT defining *the psycho-acoustic constraints*



### 2.1.3 Markov Model

Markov Model (MM) is a popular stochastic technique used in finite sequence generation problems in natural language processing, statistics, genetics, speech recognition and other applications. Because of its simplicity in designing and computational efficiency, it is one of the most important choices for many sequence generation applications. “Markov chain” is another name for MM. The term *Markov chain* refers to the sequence of random variables with the Markov property defining serial dependence only between adjacent periods (as in a “chain”). The Markov property states that the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state, not on the sequence of events that preceded [24]. A process with this property is called a *Markov process* or *memory less* property of a stochastic process.

A Markov chain of order  $m$  (or a Markov chain with memory  $m$ ),

$$Pr(x_n | x_{n-1}, x_{n-2}, \dots, x_2, x_1) = Pr(x_n | x_{n-1}, x_{n-2}, \dots, x_{n-m}) \quad (2)$$

where  $m$  is finite and  $m < n$

Equation 2 refers to  $m^{th}$  order MM, where the future state only depends on the past  $m$  states. To select the first state of a sequence, MM needs an initial probability distribution which characterizes the starting state selection from the state space. From the second state onwards transitions happen between different states. The probabilities associated with various state changes are called transition probabilities. The two parameters of the MM are initial probability distribution ( $Q$ ) and transition probabilities ( $A$ ).

Let us consider the 1<sup>st</sup> order MM equation,

$$Pr(x_1, x_2, \dots, x_{n-1}, x_n) = Pr(x_1). Pr(x_2 | x_1). Pr(x_3 | x_2) \dots Pr(x_n | x_{n-1})$$

where,  $Pr(x_1)$  is the initial probability distribution or matrix.  $Pr(x_n | x_{n-1})$  is the transition probability ( $n > 1$ ). A  $m^{th}$  order MM can be represented using an  $m+1$  dimensional transition matrix ( $A$ ).

In automatic music composition MM are used as a musical object (notes, rhythm and more) generator. The simplest (but fairly common) mapping just assigns a se-

quential group of notes to each state, with the choice of just one note (instead of a larger sequence) being fairly common [12]. In 1972, Moorer, J. A.[12] proposed a method where a Markov chain captures local statistical similarities from a corpus of pre-existing compositions and regenerate it. It was observed that *low-m* Markov chains produced strange, unmusical compositions that wandered aimlessly. In the same time, training a *high-m* Markov chains is computationally expensive.

Let us consider a melody sequence

{D A C B A C D B B A D A C A C D B B A C B}

The MM parameters  $\langle Q, A \rangle$  are extracted from the given melody sequence are shown below,

The initial probability distribution matrix is:

$$\mathbf{Q} = \begin{matrix} & A & B & C & D \\ \begin{pmatrix} 6/21 & 6/21 & 5/21 & 4/21 \end{pmatrix} \end{matrix}$$

The transition probability matrix is:

$$\mathbf{A} = \begin{matrix} & A & B & C & D \\ \begin{pmatrix} A & 0 & 0 & 5/6 & 1/6 \\ B & 3/5 & 2/5 & 0 & 0 \\ C & 1/5 & 2/5 & 2/5 & 0 \\ D & 2/4 & 2/4 & 0 & 0 \end{pmatrix} \end{matrix}$$

Counts in each row of  $A$  are normalized so the combined probabilities for each row sum to 1,

i.e,

$$\sum_{i=1}^N P_{ij} = 1 \quad \text{where } j = 1, 2, \dots, N$$

Jones [14] introduced a simple composition technique using MM in 1981. He also shows how to make a *event relation diagrams*<sup>9</sup> from the transition matrix. Figure 11 shows an *event relation diagram* made from the transition matrix of a reference melody sequence.

---

<sup>9</sup>Event relation diagrams are visualizations that show the relationships between events and tasks and how the events affect each other.

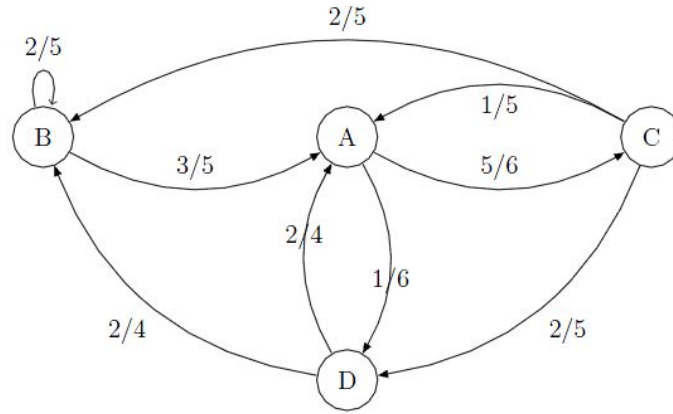


Figure 11: Event relation diagram for the reference music

A MM is not only used to generate the melodies but also to generate the rhythmic sequence of a melody. Bongjun Kim and Woon Seung Yeo [4] proposed a model in 2013 based on Markov chain to predict the rhythmic characteristics. The procedure for modeling a rhythmic MM (shown in Figure 12) is similar to the melody sequence processing (shown in Figure 11).

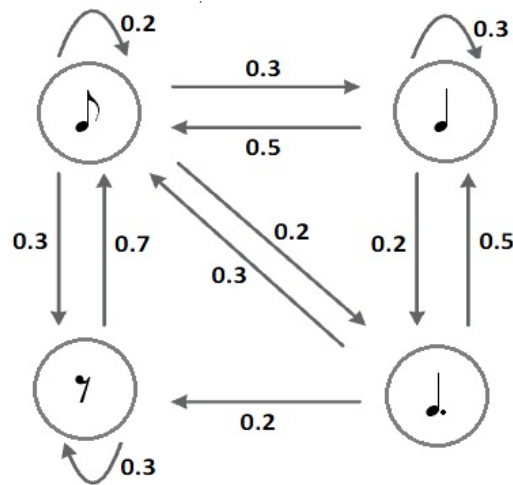


Figure 12: An example first order rhythmic MM. Source: [4]

The model shown in Figure 12 treats all the state with equal time duration (unit pulse). In practice, the rhythm symbols shown in the example model do not have equal time duration. A revised MM is used to solve this problem which uses one unit pulse for a single node (state). The example rhythmic model (Figure 12) is revised and shown in Figure 13.

To generate this model, the number of “unit pulses” in one bar needs to be defined. Here, we divide one measure into 8 beat pulses and, assuming 4/4 meter,

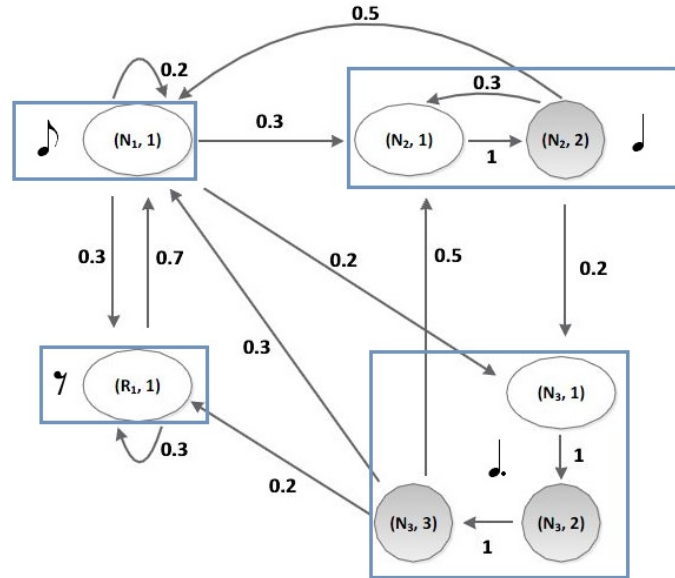


Figure 13: First order revised rhythmic MM. Source: [4]

the duration of a unit pulse corresponds to an eighth note [4]. While comparing the basic rhythmic model shown in Figure 12, the revised model's state takes more nodes when its duration is more than one unit pulse. In the revised model each state is denoted as  $(x, y)$  the first element  $x$  shows the type and duration of the event (e.g., Nd or Rd, where N: note, R: rest, and d: duration) and the second element  $y$  indicates the “counter” parameter ranging from 1 to the duration of the event [4]. For example, the quarter note has two states,  $(N2, 1)$  and  $(N2, 2)$ . The transition probabilities to newly added nodes (gray colored in Figure 13) are set to 1 because these events always occur after the first state of each event.

#### 2.1.4 Parametric Extension

Grammars can be used as music analysis tools to find the right grammar modeling a piece as well as a composition tool to generate pieces following a grammar [9]. Typically, to use grammar as a generating tool, the production rules are additionally labeled with probabilities. A grammar consists of a collection of productions and transforming non-terminal symbols into other symbols [11].

Let us consider a production rule for *crescendo*

$$A(x) : x < 64 \longrightarrow A(x + 1)$$

if the parameter  $A(64)$  is interpreted as volume then the production rule will

increases the volume of a next sequence element. Similarly grammar can be used for generating a sequence.

The following parametric grammar computes the Fibonacci series:

$$\begin{aligned}
 p1 &: A(x, y) \longrightarrow A(y, y + x) \\
 \alpha &: A(1, 1) \quad \textit{Start string (or) Axiom} \\
 &A(1, 1) \Rightarrow A(1, 2) \Rightarrow A(2, 3) \Rightarrow A(3, 5) \Rightarrow A(5, 8) \dots
 \end{aligned}$$

A parametric extension is a context-free grammar where non-terminals have parameters that take values in the set of terminals. The term parameters here refers to a small pattern of music used to make a piece by combination. The parametric grammar enables us to model some musical concepts as the notion of development which can take several short patterns as parameters, even if the actual order of patterns is not specified [9].

Mathieu Giraud and Slawek Staworko, proposed a technique to use *Bach inventions* with parametric grammar for modeling a musical structure [9]. “Bach inventions”<sup>10</sup> helps to analyze a music and to capture its pattern. The overall idea of using parametric extension is to identify the high-level structure of the score. Figure 14 shows an example measures of soprano voice used to model the Bach invention in C major.

From the reference musical score 14 the main patterns are:

*a/A*: four sixteenths in an upwards (a) or downwards (A) movement

*b/B*: four sixteenths (two successive thirds) in an upwards (b) or downwards (B) movement

*c*: two eighths, large intervals

*e*: two eighths, small intervals.

---

<sup>10</sup>The Inventions and Sinfonias is also known as the Two- and Three-Part Inventions, are a collection of thirty short keyboard compositions by Johann Sebastian Bach (1685-1750). In today’s musical world this word means this work by Bach, as it became so famous. The word originally belonged to the term in rhetoric, and here Bach used to describe the idea that could be used as a compositional theme. Thus the ‘straightforward instruction’ presents a specimen answer to the proposition as to how one can discover a good musical idea for a piece, and how to develop that into a full-scale composition. <http://www.music.qub.ac.uk/tomita/essay/inventions.html>



Figure 14: Patterns used to model musical surface of Bach invention #01 in C major, taken from the first four measures of the soprano voice. All patterns have a duration of a quarter. Source: [9].

Using the Bach Invention and parametric grammar, the reference music is decomposed into a structure shown in Figure 15.

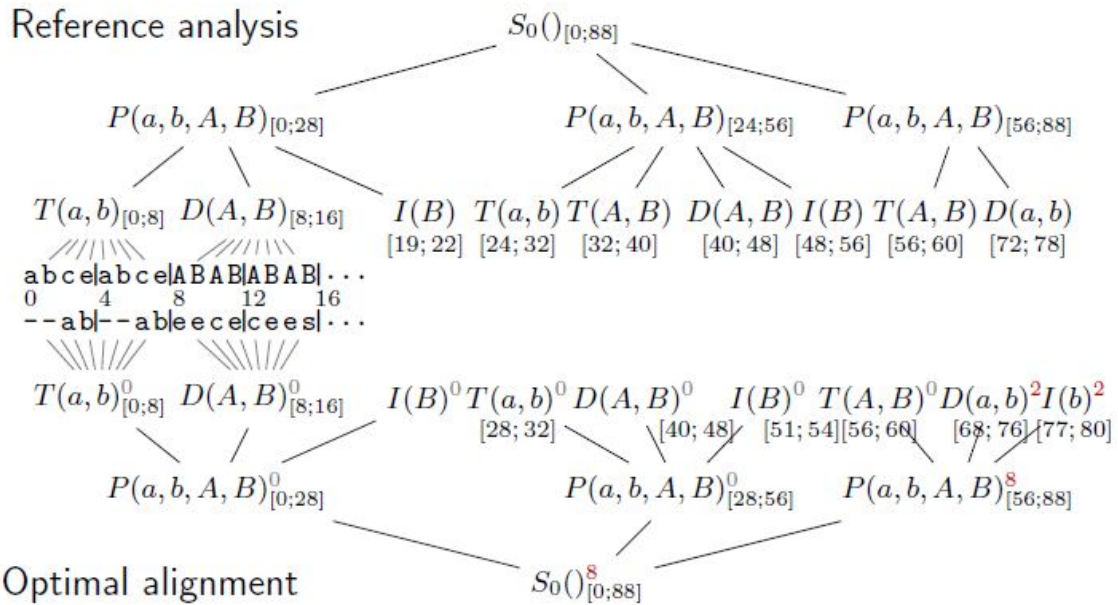


Figure 15: Decomposition of Invention #01, in C major. Each node  $n$  is displayed as  $[startA(n); endA(n)]$ , the notations used in the decomposition tree is explained in, Source: [9].

The decomposition obtained using parametric grammar is mostly descriptive rather generative. This representation could be used for music generation applications [9].

### 2.1.5 Hierarchical Grammars

It has been observed that almost all forms of music involve repetition, either of individual sequences of notes or at some higher levels of structural grouping [11]. These repetitions might happen with some changes in different levels to form a pattern or to describe a theme of the music. This property of music supports the use of *Hierarchical Grammars* for representation and generation. Hierarchical grammars have a similar specification as a DOL-System grammar (explained in Section 2.1.1), except that on the successor side of a production, one or more of the successor symbols may be an entire grammar in itself [11]. Let us consider an example rule in a hierarchical grammar for string reproduction shown below,

$$\begin{aligned}
 A = \{ & \\
 & B(x) = \{ \\
 & \quad pb1 : a(y) \longrightarrow a(y + 1)c(y/2) \\
 & \quad \alpha : a(x) \\
 & \} \\
 & pal : B(x) \longrightarrow a(x) [B(2x)a(x)] \\
 & \alpha : B(1) \\
 & \}
 \end{aligned}$$

The rule shown above has two rules parameter  $A$  and  $B$ . The rule set  $A$  works along the horizontal left to right direction and  $B$  works on the vertical down direction.

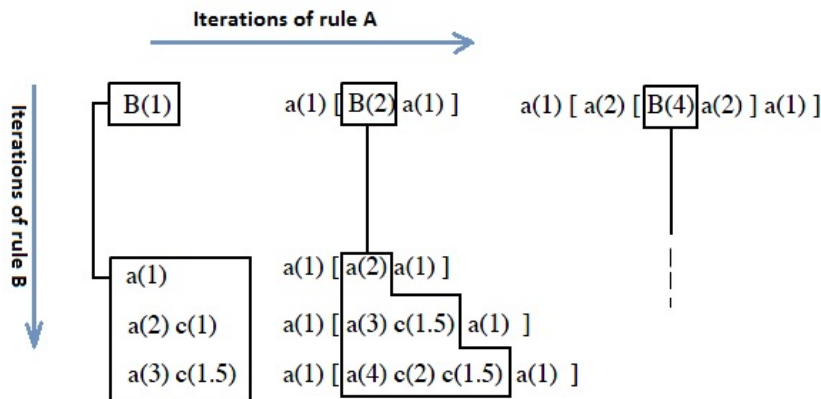


Figure 16: Strings produced by hierarchical grammars. Source: [11]

For every iteration the rule set produces a new string sequence. Figure 16 shows

the sequence generated by the simple hierarchical grammars.

Generating a sequence using a hierarchical grammars is a simple process. On the other hand finding the rules from a reference music sequence is a complicated process. In 1997, Craig G. Nevill-Manning and Ian H. Witten [18] described an algorithm, called *SEQUITUR* that identifies hierarchical structure in sequences of discrete symbols. *SEQUITUR* produces a grammar based on repeated phrases in the input sequence (in our case it is music). Each repetition gives rise to a rule in the grammar, and is replaced by a non-terminal symbol, producing a more concise representation of the sequence [18]. Figure 17 shows an example music sequence matches within and between two chorales using *SEQUITUR* algorithm.

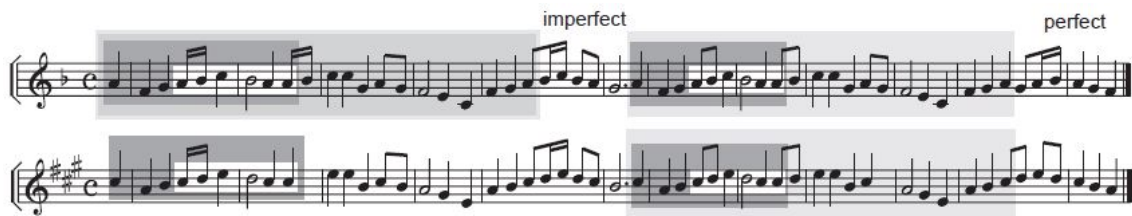


Figure 17: Illustration of matches within and between two chorales for chorales *O Welt, sieh hier dein leben* and *O Welt, Ich muss Dich lassen* by J.S. Bach. Source: [18]

Very complex grammars (such as those with many levels of hierarchy, or productions that cause exponential growth in the produced string) need an efficient computing to handle the production of symbols.

### 3 Proposed Method

Algorithmic music composition can be done by numerous methods as explained in section 2.1. We would like to combine two existing techniques to generate a melody content. Finite-Length Markov Processes with Constraints [20] is used to generate the melodies and Probabilistic Prediction of Rhythmic Characteristics in Markov Chain-based Melodic Sequences [4] (with modifications explained in section 3.3) used to generate the rhythm. These two techniques will handle the melody and rhythm separately and finally both are combined to create a musical score. Along with a generation of melodies using an existing technique we would like to include more constraints other than constraints explained in [20]. Section 3.2.3 explains about the constraints which are going to be included in the melody generation. A new model for algorithmic music composition using a Hidden Markov Model (HMM) with Categorical Distributions (CD) is introduced in section 3.4. This model will generate both melody and rhythm unlike the previous method. Section 3.4 describes the model training and generation process. The HMM with CD model for melody generation is a new technique that has been made as a part of this thesis work.

#### 3.1 Data

The proposed method for melody generation comes under the category of “Learning systems”. Learning systems are systems that do not have prior knowledge of the domain, but instead it will learn from examples (Training data). In the problem of music generation the production rules and constraints are the knowledge. To get knowledge about the style of the music, generation model needs a training data to learn. In AI, the process of getting knowledge from an training data is commonly called *Training phase*. Training data will provide statistical information to the algorithm for generation (output). In other words the style of the training data (melodies) are learned by the algorithm. The database shown in Table 2 includes a diversity among the collection of solos. It will help us to have an idea over the algorithm's ability to imitate a different set of music.

A pentatonic scale is a musical scale or pattern with five notes per octave. The pentatonic scale has a unique character and is complete in terms of tonality. These five note scale is the basis for countless folk songs around the world. One construction takes five consecutive pitches by omitting the fourth and the seventh notes [6]. For example, if we start from C major, the pentatonic scale is C D E G A. Figure 18

Genre/Style	Number of Melodies
Folk	53
Lullaby:	1
Spiritual:	27
Children Song:	8
Shanty:	1
Swing:	1
Guitar Licks:	8
Modern classical music:	1
Jazz standard:	1
Exercising songs:	2

Table 2: Melodies collection in database

shows an example of a famous pentatonic scale composed by David De Loach <sup>11</sup>

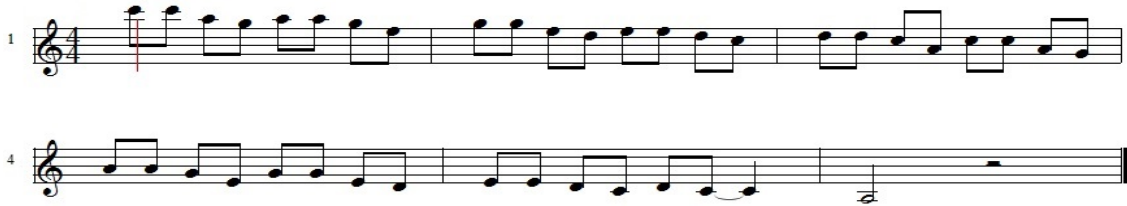


Figure 18: Pentatonic melody by David De Loach

The database contains 103 different pentatonic pattern MIDI files.

### 3.1.1 MIDI Features Extraction

All the melodies were collected in MIDI file format. The MIDI format will give separate sequences of pitch, time duration, velocity and note-on/off status, tempo and time signature information. In general, MIDI file format handles an occurrence of a rest in musical score by holding a silence (depends on the rest duration). For our convenience, the sequence obtained from the MIDI file is further processed to add information about a rest in the note sequence.

Let us consider an example music sequence shown in Figure 19,

<sup>11</sup>David De Loach is an experienced guitarist and author of a popular college level guitar textbook.



is defined as a triple  $\langle X, D, C \rangle$ :

$$\mathbf{X} = \{X_1, \dots, X_n\}$$

$$\mathbf{D} = \{D_1, \dots, D_n\}$$

$$\mathbf{C} = \{C_1, \dots, C_m\}$$

where  $\mathbf{X}$  is a set of variables, each variable  $X_i$  has a nonempty domain  $D_i$ . The domain  $D_i$  is a collection of all the possible values and  $C$  is a set of constraints. Each constraint  $C_i$  involves some subset of the variables and specifies the allowable combinations of values for that subset. A solution to a CSP is a complete assignment that satisfies all constraints.

The problem formation of CSPs could be well understood with an example, let us consider a known map coloring problem. Figure 20 is showing the states and territories of Australia. Now the problem is to color each state using three different colors (*red, green and blue*) with the condition that no neighboring states should have the same color [15].



Figure 20: States and territories of Australia

The first step is to formulate the given problem in-terms of CSP parameters. Consider the states (*WA, NT, Q, NSW, V, SA, and T*) as variables  $\mathbf{X}$ , and the colors (*red, green and blue*) as domain  $\mathbf{D}$ . The condition that neighboring states are not allowed to have same color is considered as a constraint  $\mathbf{C}$ . For better understanding visualize the map as constraint graph shown in Figure 21.

In a constraint graph representation the nodes represent the variables (states) and

link between the nodes show the connection of neighboring states.

$$X = \{WA, NT, Q, NSW, V, SA, T\}$$

$$D = \{red, green, blue\}$$

$$C = \{WA \neq NT, WA \neq SA, NT \neq Q, NT \neq SA, Q \neq SA, Q \neq NSW, \dots V \neq T\}$$

Inequality in constraint  $C$  represent variables (states) are not supposed to take similar colors from the domain  $D$ .

The problem can start from assigning colors to any variable available in  $X$ . The first variable has the freedom of taking any of three colors as there are no previous conditions. From the second variable onwards the possible colors for each state are assigned by obeying a constraint  $C$ . Figure 22 shows the process of assigning colors to the state starting from  $WA$ . By following this step until all the variables get a color the process continues. Finally we will end up with many possible solutions as a result to this problem. One of the possible solutions is,

$$\{WA = red; NT = green; Q = red; NSW = green; V = red; SA = blue; T = green\}$$

The CSP is used along with MM in the melody generation process to include the control constraints (explained in section 3.2.3). The next section explains, how constraints can be compiled into a non-homogeneous MM, using CSP and renormalization techniques.

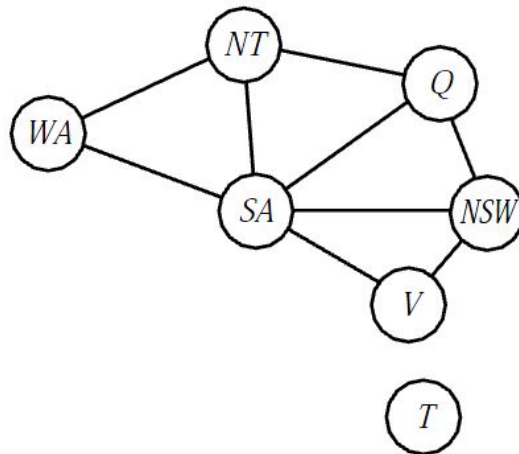


Figure 21: Map transformed as constraint graph [15]

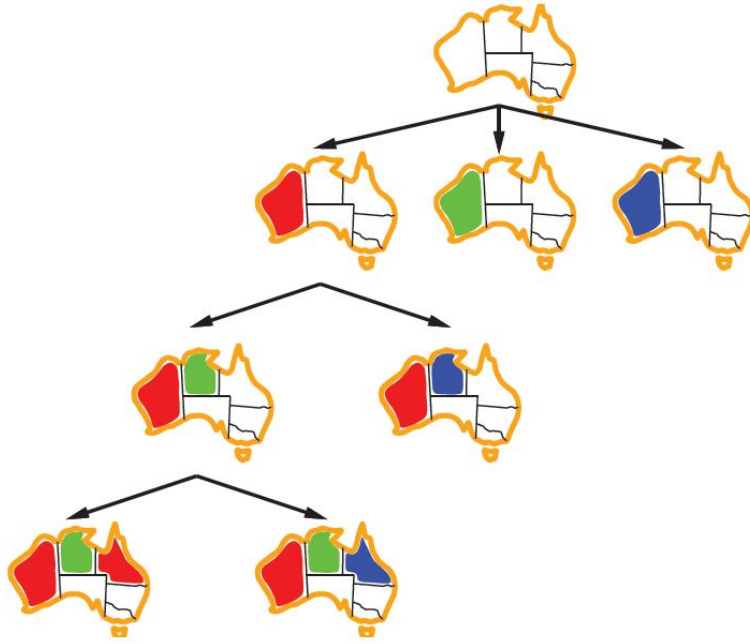


Figure 22: Color assigning through CSP. Source: [15].

### 3.2.2 MM with CSPs

The MM is going to generate melodies and the CSP will make sure all control constraints are included in the melody. To combine a MM with CSP, first we need to train the MM with the training set. Once the MM is trained, we need the parameters such as length of the final sequence (melody length) and control constraints to be satisfied.

Lets us consider an example reference melody show in Figure 23 and requirement is output should be of length 4 and the control constraints are second note should be either  $C$  or  $D$  and the last note has to be  $C$ .



Figure 23: Reference musical score

From the given melody, the parameters of MM are calculated as below

Initial probability matrix is:

$$\begin{matrix} & C & D & E \\ \begin{pmatrix} 4/7 & 3/14 & 3/14 \end{pmatrix} \end{matrix}$$

Transition probability matrix is:

$$\begin{array}{c} C \quad D \quad E \\ C \begin{pmatrix} 2/7 & 3/7 & 2/7 \end{pmatrix} \\ D \begin{pmatrix} 2/3 & 0 & 1/3 \end{pmatrix} \\ E \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \end{array}$$

The above two parameters are sufficient to generate a sequence of melody statistically similar to the reference melodies. But, the given constraints may or may not be satisfied. So, we need to formulate the given constraints in terms of CSP problem structure [20].

The variables are  $X_1, X_2, X_3, X_4$  (number of  $X$  denotes the required output length of notes) and  $C_1, C_2, C_3, C_4$  (each  $C_i$  corresponds to output variable  $X_i$ ) are the constraints with domain values  $C, D, E$  (possible notes).

With respect to the constraints, the MM parameters are re-normalized as below, There are no conditions to apply on the first variable  $X_1$  of the output sequence  $X$ . So the initial probability matrix of MM is used without any changes.

$$C_1 = \begin{array}{c} C \quad D \quad E \\ \begin{pmatrix} 4/7 & 3/14 & 3/14 \end{pmatrix} \end{array}$$

The second variable  $X_2$  should be either note  $C$  or  $D$ . In general, this kind of constraint can be achieved by making the column of notes into zero in a transition matrix other than possible (allowed) notes. Hence, make the column corresponds to note  $E$  into zero. After eliminating the column of note  $E$ , re-normalize each row of the transition matrix to make the sum is equal to one. This modification of a transition matrix will make the variable  $X_2$  to get note  $D$  or  $E$ .

$$C_2 = \begin{array}{c} C \quad D \quad E \\ C \begin{pmatrix} 2/5 & 3/5 & 0 \end{pmatrix} \\ D \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \\ E \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \end{array}$$

There is no condition given for the third variable  $X_3$ , this makes the variable independent and the derived transition matrix from the reference is used without any

changes.

$$C3 = \begin{array}{c} C \\ D \\ E \end{array} \begin{array}{ccc} C & D & E \\ \left( \begin{array}{ccc} 2/7 & 3/7 & 2/7 \\ 2/3 & 0 & 1/3 \\ 1 & 0 & 0 \end{array} \right) \end{array}$$

The last variable  $X_4$  should end in note  $C$ , so the constraint  $C_4$  needs a modification similar to  $C_2$ . Making the columns of notes  $D$  and  $E$  into zero and re-normalize the resulting transition matrix.

$$C4 = \begin{array}{c} C \\ D \\ E \end{array} \begin{array}{ccc} C & D & E \\ \left( \begin{array}{ccc} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{array} \right) \end{array}$$

The final sequence structure with a combination of MM (as constraints) and CSP is shown below,

$$\xrightarrow{C_1} X_1 \in \{C, D, E\} \quad \xrightarrow{C_2} X_2 \in \{C, D\} \quad \xrightarrow{C_3} X_3 \in \{C, D, E\} \quad \xrightarrow{C_4} X_4 \in \{C\}$$

Applying the constraint (modified matrix) over the generation of a note at correct position (variable), will lead to a constraint satisfied melody sequence [20].

### 3.2.3 Melody Constraints

In the work Finite-Length Markov Processes with Constraints [20], the control constraints are included in three different themes, they are:

*Continuation:* Input is continued to produce a sequence of the same size. A constraint is posted on the last note to ensure that it is terminal, i.e., occurred at the end of an input melody, to produce a coherent ending [20].

*Variation:* Is generated by adding two unary constraints that the first and last notes should be the same, respectively, as the first and last notes of the input [20].

*Answer:* Is like a Continuation, but the last note should be the same as the first input note. This creates a phrase that resolves to the beginning, producing a sense of closure [20].

In this thesis for melody generation process, 4 types of constraints are included

other than constraints which are explained above. They are:

1. *Include one note*
2. *Include several notes*
3. *Use only a set of notes*
4. *Include a given pattern of notes*

**1. *Include one note:*** This constraint is to add a single note in the output melody sequence. From the given input (sequence length  $n$  and the note to be included) the trained MM and CSP will formulate a problem as explained in section 3.2.1. The requirement is to include one note in a sequence, so the first step is the selection of one output variable between  $X_1$  to  $X_n$ . It is done by choosing a random number between 1 to  $n$ . Once the variable ( $X_i$ ) is selected, the next step is to modify the constraint  $C_i$  corresponds to the selected variable  $X_i$ . The modification of the transition probability to include a note is well explained in section 3.2.1.

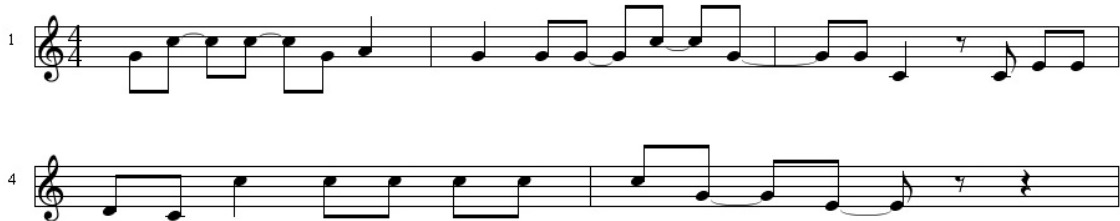


Figure 24: Output melody sequence with  $D4$  (note)

Figure 24 shows a melody generated by the algorithm which satisfies the constraint of including the note  $D4$ . The output is in the style of the training set and also includes the given note.

**2. *Include several notes:*** This is an extended version of the previous constraint, which includes more than one note in the output sequence. The input for this constraint is a set of notes and the length of the output sequence. For the given set of notes we select a position for each note randomly. Additionally, the MM parameters should be modified with respect to the given notes. The modified MM (constraints  $C_i$ ) gives a guarantee that the output contains the given set of notes.



Figure 25: Output melody sequence with  $A4$  and  $B4$

Figure 25 shows a melody generated by the algorithm that satisfies the constraint (Include notes  $A4$  and  $B4$ ) and remains in the reference style.

**3. Use only a set of notes:** The idea is to generate an output melody sequence only with a combination of a given set of notes. This is a strong constraint compared to the above two constraints. To implement this constraint, we use MM and L-Systems instead of CSP. The trained MM will generate a melody sequence of a given length. This generation is completely random with no constraints involved, in other words it is a simple MM sequence generation. The generated melody sequence is post-processed by a L-Systems with a set of grammar rules (explained in section L-Systems).

The grammar rules rewrite the generated melody to satisfy the constraint. The generated melody sequence may contain the notes which are not allowed to present. The grammar rules will replace those notes (not allowed to present) with the note from the given set of notes. By rewriting, we can ensure that the output sequence contains only the given set of notes.

The melody sequences in the training set contains 12 semitones in different combinations. The L-Systems (explained in the section 2.1.1) parameters are:

$\Sigma$ :  $C C\# D D\# E F F\# G G\# A A\# B$  (Semitones)

$\alpha$  (axiom or start string): Generated melody (MM output sequence without constraint)

Production rules:

$C\# \rightarrow D$

$D \rightarrow E$   
 $D\# \rightarrow E$   
 $E \rightarrow C$   
 $F \rightarrow E$   
 $F\# \rightarrow G$   
 $G \rightarrow A$   
 $G\# \rightarrow A$   
 $A \rightarrow C$   
 $A\# \rightarrow B$   
 $B \rightarrow C$

Based on the requirement, the production rules are selected from the list shown above. The selected production rules are applied over the generated melody sequence ( $\alpha$ ). The resulting sequence will contain only the given set of notes.

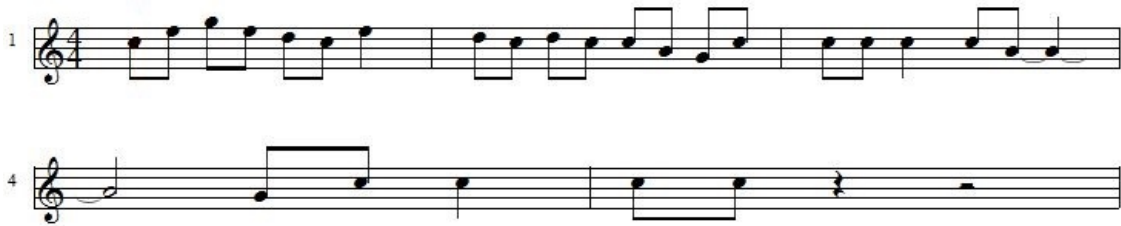


Figure 26: Algorithm generated melody sequence ( $\alpha$ ) with no constraints

Figure 26 shows the output melody sequence ( $\alpha$ ) generated by the trained MM with no constraints.

Let us consider that the generated melody sequence contains the notes  $A4$  and  $A5$ , which are not a part of the given set of notes ( $C4 G4 C5 D5 E5 G5$ ). By string rewriting technique, the generated melody (randomly by MM) is modified to make sure the output contains only  $C4$ ,  $G4$ ,  $C5$ ,  $D5$ ,  $E5$  and  $G5$  notes. So the grammar rules involved in this rewriting are,

$\Sigma$ :  $A4 C4 G4 C5 D5 E5 G5 A5$

$\alpha$ : Generated note sequence (MM output with no constraints)

P: (Production rules)

$A4 \rightarrow C4$

$A5 \rightarrow C5$ .

Figure 27 shows the output melody sequence rewritten by the grammar rules to make sure it contains only  $C4$ ,  $G4$ ,  $C5$ ,  $D5$ ,  $E5$  and  $G5$  notes.

**4. Include a given pattern of notes:** This constraint is about including a pattern of notes in a melody sequence. The pattern specifies both notes and their duration. The method for including a duration for the given pattern is explained in section 3.3. This constraint is more similar to *Including note* but instead of a note here it is a pattern of notes. The selection of position should make sure the required length of the output should not be exceeded while adding the pattern inside. The position range is decided by the method below:

Position range: 1 to  $(n - l)$ ,

where  $n$  is the required length of the output sequence (number of notes) and  $l$  is the given pattern length (number of notes in given pattern)

By selecting a random number in the position range (to include the given pattern) will make sure the output length does not exceed the required length. For example if the required output sequence length is 20 ( $n$ ) and the given pattern length is 4 ( $l$ ). The position range is 1 to 16 and a random number will be chosen in this range to include the given pattern.

Including a pattern can be achieved by using CSP, but L-Systems make the work much easier compared to CSP. The trained MM will generate a output sequence without any constraints for the required length. Once the sequence is generated, the given pattern of notes are rewritten<sup>12</sup> in the selected position without consid-



Figure 27: Output melody sequence only with ( $C4$   $G4$   $C5$   $D5$   $E5$   $G5$ )

<sup>12</sup>The process of rewriting the generated notes with the given pattern of notes tend to change the reference style. The effect of rewriting the notes can be reduced by choosing a pattern that improvises the given reference style.

ering the notes already present. The position is stored for further processing, as mentioned above pattern includes both notes and their duration. The stored position is used in rhythm generation to make sure the pattern gets its right duration. The rhythm generation for this constraint is explained in Section 3.3.



Figure 28: Given pattern of notes (G4 F5 E5 D5)

Lets us consider a pattern  $G4 F5 E5 D5$ , shown in Figure 28 and it should be included in the output melody sequence. Figure 29 shows the output sequence which includes the pattern and also remains in the style of the training set. In Figure 29 the group of notes inside the green box is the given pattern.

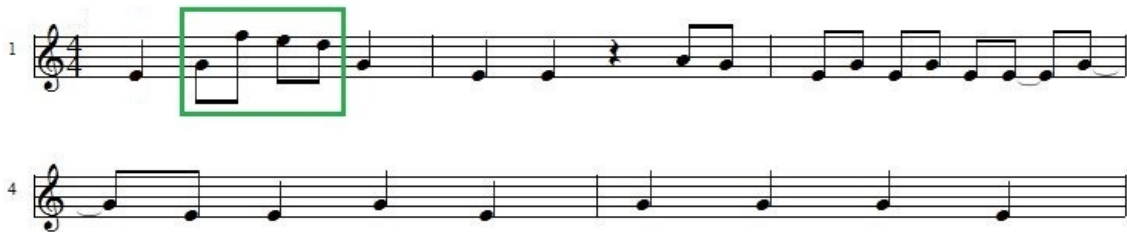


Figure 29: Output melody sequence with given pattern (G4 F5 E5 D5)

Another example for including a pattern of notes is shown in Figure 30.

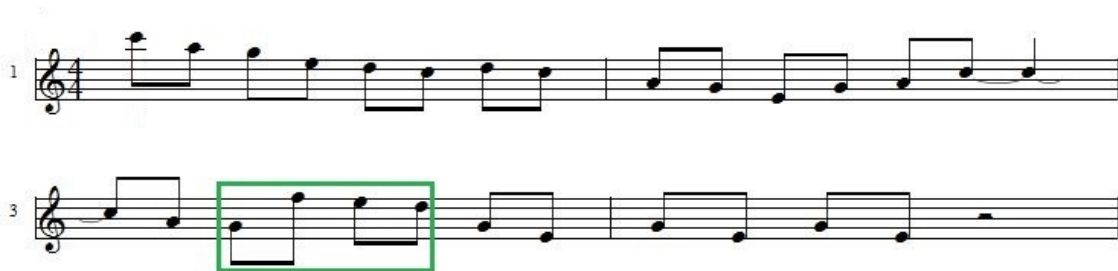


Figure 30: Output melody sequence with given pattern (G4 F5 E5 D5)



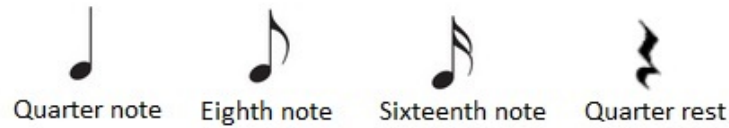


Figure 32: Unique rhythm symbols in the reference musical score

(nodes) in the MM and the probability is also taken from the rhythm sequence. Once the required parameters for the MM are gathered the rhythm sequence can be generated for the required number of notes. To generate a rhythm sequence, we need a melody sequence which is generated using the techniques explained in Section 3.2. The duration information for each note present in the melody sequence is generated using the trained MM.

For example, the melody sequence length is 25 (number of notes), duration information generated by the MM is shown below,

Given melodies,

$$\{G5, G5, E5, E5, E5, G5, G5, B5, B5, A5, G5, E5, E5, D5, G5, E5, E5, D5, G5, 'R', E5, D5, G5, G5, A5\}$$

Generated rhythm,

$$\{0.5, 0.5, 1.0, 1.0, 0.5, 1.0, 0.25, 1.0, 0.25, 0.25, 0.25, 1.0, 0.25, 1.0, 1.0, 1.0, 0.25, 0.25, 0.25, 1.0, 0.5, 1.0, 1.0, 0.5, 1.0\}$$

The melody and rhythm sequence are combined to form a final musical score shown in Figure 33.

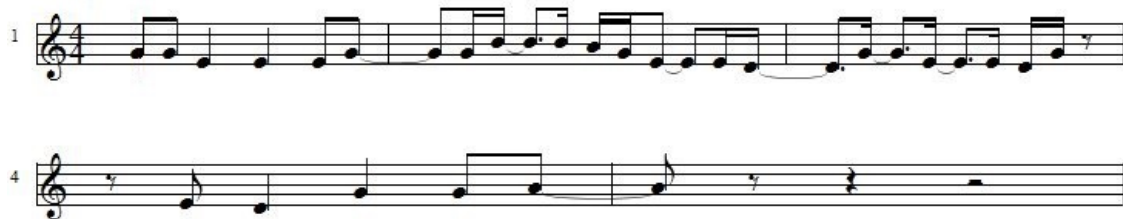
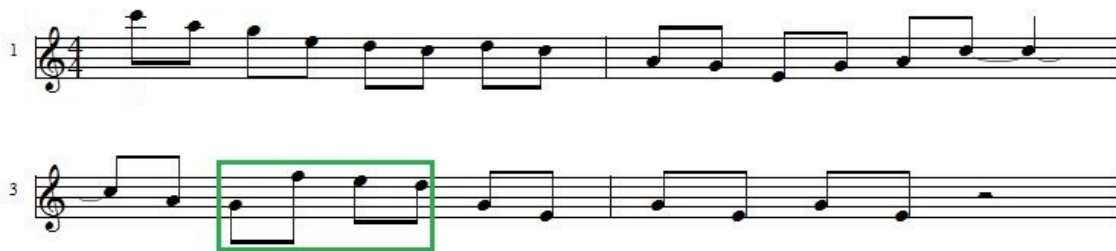


Figure 33: Musical score formed by melody combined with generated rhythm

As mentioned in section 3.2.3 *Include pattern of notes*, rhythm generation cannot be a random generated sequence. The duration information given in terms of pattern should be included in the rhythm sequence in the location that corresponds to the pattern of notes. Using string rewriting grammar (L-Systems) the rhythm sequence can be post-processed to include the given pattern of rhythm. Once the random rhythm sequence is generated, the length (number of notes and duration) of both rhythm and melody should be the same. Now we need to use the location (stored while processing the melody) where the pattern of notes is inserted. The given duration information of the pattern is rewritten over a random generated rhythm without considering the actual value present. Now the pattern of notes and duration takes a correct position inside the final musical score. Let us consider an example duration in pattern of notes ( $G4 F5 E5 D5$ ) is 0.5, 0.5, 0.5, 0.5 (all are eighth notes). Figure 34 shows the final musical score with given pattern of notes and its corresponding duration.



### 3.4 Melody and Rhythm Generation using HMM

In general, the musicians handle both melody and rhythm together in composition. But the *Algorithm 1* handles them separately and then combine to make a final sequence. Even though the melody and rhythm are taken from the same training set. The generation happens in two individual models. In order to overcome this limitation *Algorithm 2* is designed using HMM with CD which handles both melody and rhythm together.

A well known generative model HMM is combined with CD to generate both melody and rhythm in a single process. The HMM is fragmented into two major parts, one as the MM and the other as *Output Distribution* (in our case it is CD). The MM is going to handle notes and *Output Distribution* is for rhythm.

#### 3.4.1 Categorical Distribution

CD is a generalized Bernoulli distribution and it describes the result of a random event that can take on one of  $K$  possible outcomes, with the probability of each outcome separately specified.

The parameters of CD are,

$K$  - Number of possible outcomes or categories ( $K > 0$ )

$p_1, p_2 \dots p_K$  where  $p_i$  is the probability of category with  $\sum p_i = 1$

In our problem of music generation, the CD handles the rhythm generation process. Here, we need a separate CD for all the unique pitches (all the rests are treated as one symbol, 'R') present in the reference music. For every CD, the parameters are extracted from the given training set (rhythm). This approach of using separate CD for every note/rest restricts the algorithm to create a new duration value in the generation process. The CD is calculated as follows:

First select the note (any one of the available note in the training melody sequence) and count the number of occurrence in the training melody sequence. For example, if the note  $D$  is selected and it occurs 5 times in the melody sequence.

Once the occurrence (for the selected note) is counted, group the similar durations and count it separately. For example, note  $D$  occurs 3 times with 8th note duration and 2 times with half note duration.



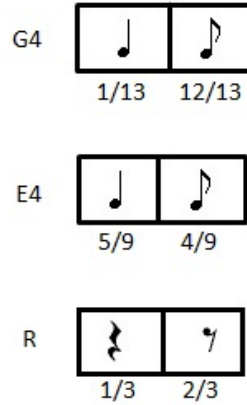


Figure 36: CD for the given reference melody

3. Pick a uniformly distributed number between 0 and 1.
4. Locate the greatest number in the CDF whose value is less than or equal to the number just chosen.
5. Return the category corresponding to this CDF value.

By following the above steps a rhythm for the note/rest is chosen. The CD is combined with trained MM (over melody) to make a complete model to generate music.

### 3.4.2 HMM with Categorical Distribution in Generation

For making the HMM model work, the given reference pitches is analyzed by the algorithm to get three parameters they are,

$$\langle \mathbf{Q}, \mathbf{A}, \mathbf{B} \rangle$$

Where

$Q$  - Initial probability matrix

$A$  - Transition probability matrix

$B$  - Output distribution matrix (CD)

The parameters  $Q$  and  $A$  are extracted from the pitches of a training set.  $B$  is taken from the rhythm of the training set as explained in Section 3.4.1. Once these three parameters are learned from the reference, the model is ready to generate both melody and rhythm.



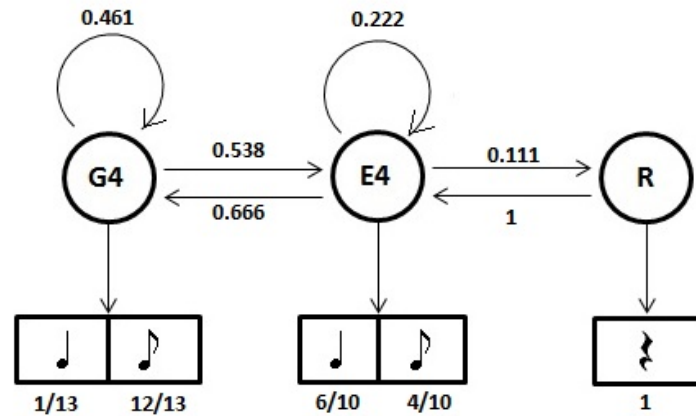
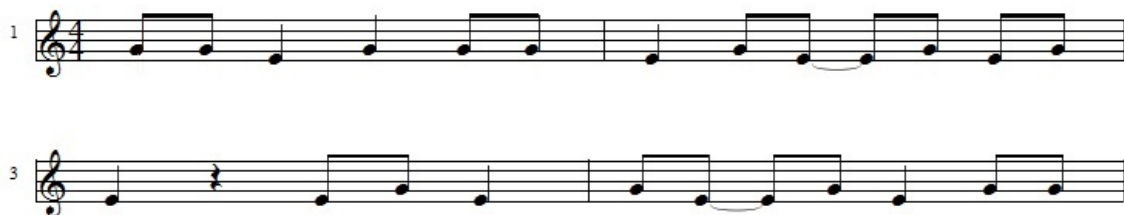


Figure 38: HMM with CD model for the reference music

The model shown above is capable of imitating the given style of the reference melody. The user has to specify a length of the output sequence (number of notes) that the model will generate. This model is designed in such a way that the number of notes in the output sequence can be controlled but not the time duration. The HMM output for the reference melody *Bee Bee* folk song is shown in the Figure 39. It clearly shows that the presence of rest (quarter rest) in the generated sequence may or may not be there in further generation. But there is no chances of getting a rest (half or eighth note rest) other than *quarter rest*.

Figure 39: *Bee Bee* folk song style imitated by HMM with CD model

An example reference music *Green Mountain*-Taiwanese folk song and the style imitated by HMM with CD model is shown in Appendix 6.5.

The HMM with CD model is evaluated without including any musical constraints. It is possible to include musical constraints similar the algorithmic generation explained in Section 3.2.3.

## 4 Results

To evaluate the performance of the composed melodies, a listening test was designed. Since music is subjective, the listening test was conducted to evaluate the algorithmic composition in two different subjective measures. The proposed algorithms involve a variety of musical constraints (explained in section 3). Every constraint is capable of generating a melody and it needs to be evaluated. Hence the listening test is split into two phases as *Test 1* and *Test 2*. Inevitably human judgment carries with it a degree of uncertainty and variability, and it is the goal of most experimenters to reduce this to a minimum by using experienced listeners who are carefully trained to recognize and evaluate the question [8]. We have chosen 20 listeners<sup>16</sup> from *Semantic Music Technologies Team* at Fraunhofer IDMT to conduct *Test 1* and *Test 2*.

*Test 1*: The aim of this test was to evaluate the algorithm composed melodies against the human composition. A set of human and algorithm composition along with its labels (Human composed melody and algorithm composed melody) were given to the listeners (Training). Once the listener gets an idea over the difference in two different compositions, an unlabeled melody was presented with the question -‘*Test melody composed by human or algorithm ?*’’. Based on their perception and subjective measures the test melody was evaluated. In order to avoid the random guess over the test melodies, another category ‘*Not sure*’ along with ‘*Human composition*’ and ‘*Algorithm composition*’. The ‘*Not sure*’ option helps the listener in case there is no confidence on the test melody or the listener gets confused to identify. For *Test 1*, 15 samples (each of maximum 6 seconds) of melody have been used (5-Human composed and 10-Algorithm composed). The results (normalized) are shown as a confusion matrix in Table 3.

		Predicted		
		Human Composed	Algorithm Composed	Not sure
Original	Human Composed	73%	19%	8%
	Algorithm Composed	22.5%	69%	8.5%

Table 3: Confusion matrix for the *Test 1*

<sup>16</sup>The listeners have different expertise in music.

Based on *Test 1* result, 22.5% of the algorithm composed melodies are predicted as human composed. This misprediction shows that the algorithm composed melody sounds more similar to the human composition. On the contrary, 69% of the algorithm composed melodies were predicted correctly (as algorithm composed). This shows that the algorithm compositions lack in getting the feel as human composed melodies. On the other hand, this (*Test 1*) subjective evaluation answered the question ‘*Test melody composed by human or algorithm ?*’ but not about the *pleasantness* of the melody. The melodies generated by AI techniques still sound pleasant similar to the human composition for example, the melodies composed by “Lamus”<sup>17</sup>. In the *Test 1* result, 8.5% of the algorithm composed melodies were categorized in ‘*not sure*’. This could be interpreted as the melodies were not artificial enough to be classify as ‘*Algorithmic composition*’ as well as not too natural to be classified as ‘*Human composition*’.

In *Test 2* , the *pleasantness* of the algorithm composed melodies with constraints were evaluated. The listeners were asked to listen to the test melodies (5 melodies per constraint category) to avoid the surprise in the evaluation. The categories for evaluation were:

- *Include one note (Algorithm 1)*
- *Include several notes (Algorithm 1)*
- *Include a given pattern of notes (Algorithm 1)*
- *Use only a set of notes (Algorithm 1)*
- *No constraint (Algorithm 1)*
- *No constraint (Algorithm 2)*

where,

*Algorithm 1*: Melody and rhythm generation separately (Section 3.2.3 and 3.3)

*Algorithm 2*: Melody and rhythm generation using HMM with CD model (Section 3.4)

From the 6 categories, 5 samples of melody per category were generated (total 30 samples). When the listeners listen to 30 test melodies, the question - ‘*How pleasing is the algorithm generated melody ?*’ was asked for every test sample. As a response

---

<sup>17</sup>Lamus is a computer composer specialized in contemporary classical music. It learn rules to compose music similar to humans.

to this question, they were asked to rate the test melodies on the below scale:

*Excellent* (score 100–80)

*Good* (score 80–60)

*Fair* (score 60–40)

*Poor* (score 40–20)

*Bad* (score 20–0).

The scaling criteria was designed by following the recommendations of MUSHRA<sup>18</sup> ITU TU-R BS.1534-2 [10].

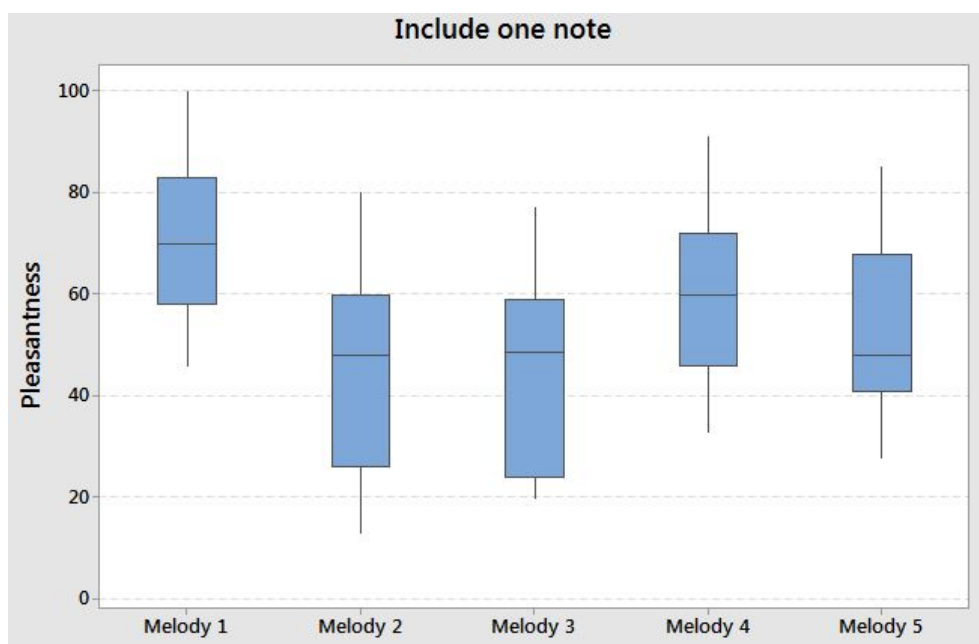


Figure 40: Pleasantness score for Algorithm 1 melody samples with constraint *Include one note*.

Each melody sample in the graph is an average of pleasantness score given by the 20 listeners. Figure 40 shows that including one note in the output sequence of the algorithmic composition ranged between *Good* and *Fair* to the listeners.

*Including several notes* in the output sequence ranged between *Fair* and *Poor* to the listeners shown in Figure 41. More notes are included randomly in the sequence. There are lot of chances that the note which is being included in the sequence may

<sup>18</sup>MUSHRA stands for **M**Ultiple **S**timuli with **H**idden **R**eference and **A**nchor and is a methodology for subjective evaluation of audio quality, to evaluate the perceived quality of the output from lossy audio compression algorithms.

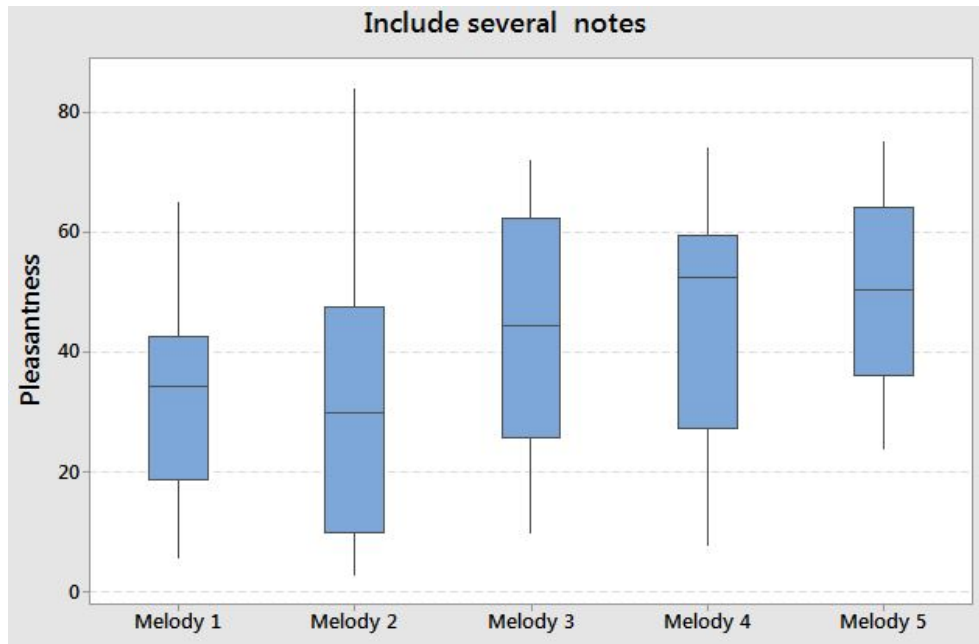


Figure 41: Pleasantness score for Algorithm 1 melody samples with constraint *Include several notes*

not adapt with the actual style of the melody sequence.

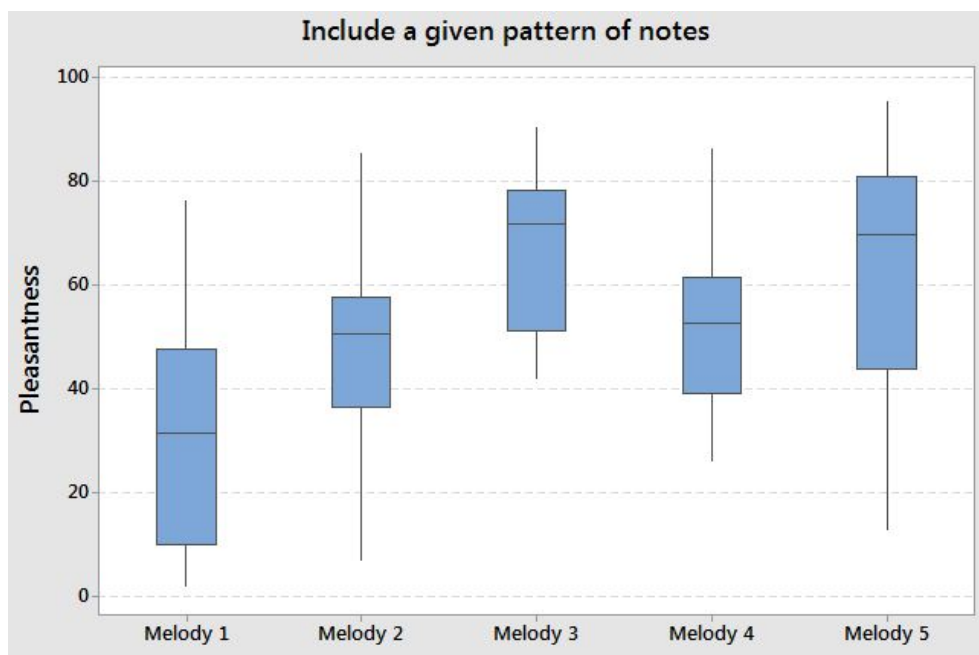


Figure 42: Pleasantness score for Algorithm 1 melody samples with constraint *Include a given pattern of notes*

By *Including a given pattern of notes* (notes and duration) in the output sequence, the results ranged between *Good* and *Poor* as shown in Figure 42.

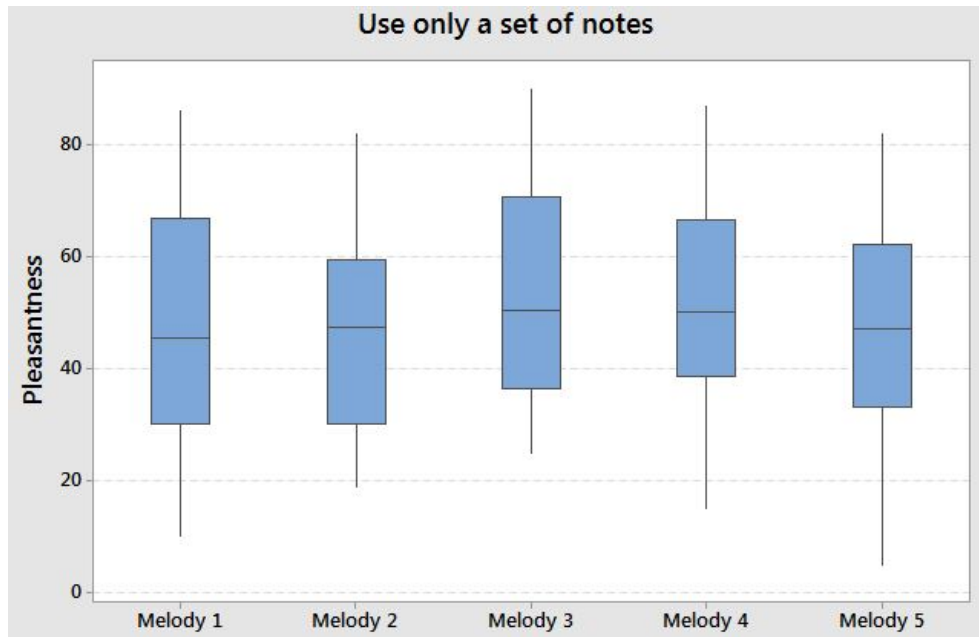


Figure 43: Pleasantness score for Algorithm 1 melody samples with constraint *Use only a set of notes*.

Making an output sequence with the constraint - *Use only a set of notes*, is the strongest constraint compared to others. Figure 43 shows that all the samples in the constraint *specific notes* sounds *Fair* to the listeners.

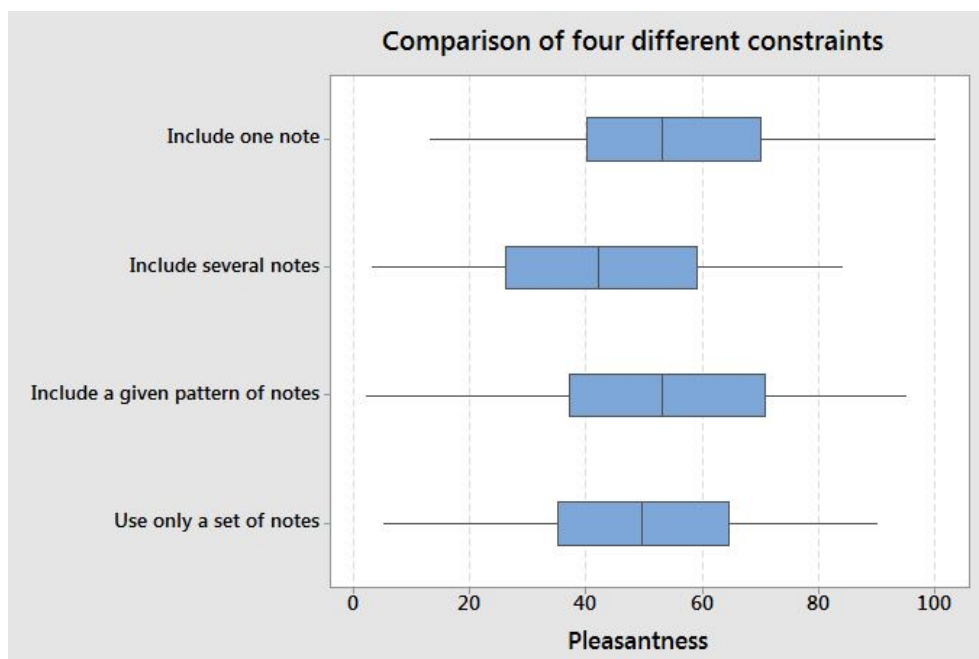


Figure 44: Pleasantness score for 4 categories

Figure 44 shows a comparison of the 4 categories of melody generation. From the comparison chart it is clearly evident that all the melody samples sound *Fair* to the listeners. The standard deviation of the comparison chart shows that the listeners were scattered on the *pleasantness* scale but the median lies between the range 40-60 (*Fair*).

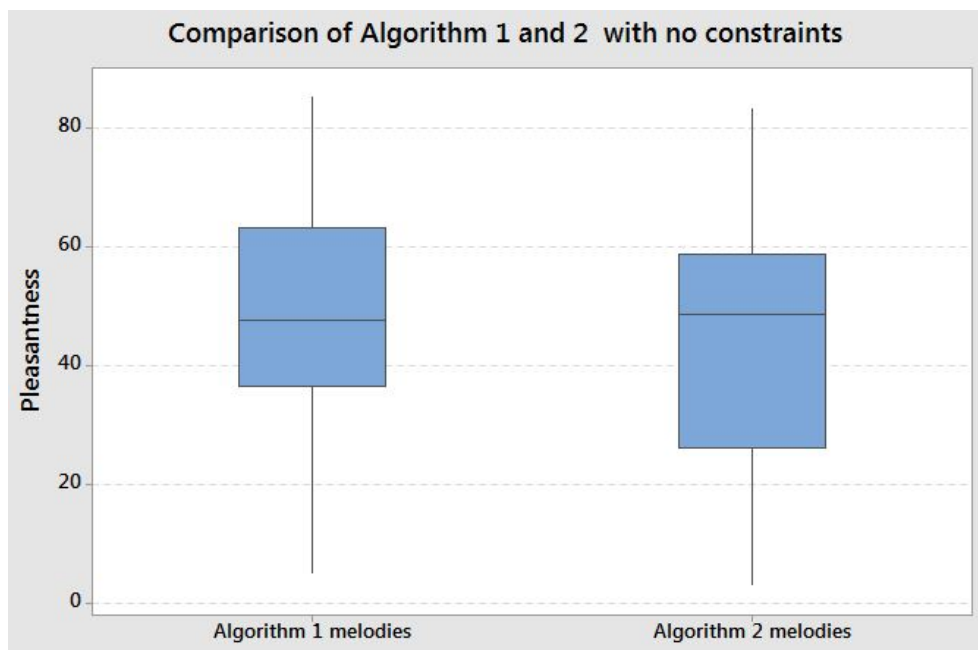


Figure 45: Pleasantness score comparison Algorithm 1 and Algorithm 2 melody samples with *No constraints*

As we explained in section 3.4, *Algorithm 2* is modeled with no constraints. To have a better comparison for the melodies generated by this model, *Algorithm 1* was made to generate melodies with no constraints. Figure 45 shows the comparison of *Algorithm 1* and *Algorithm 2* with no constraints. Interestingly, the median for *Algorithm 2* (Median is 48.5) is higher than *Algorithm 1* (Median is 47.5). But the upper quartile<sup>19</sup> for *Algorithm 2* (Upper quartile is 58.5) is less than *Algorithm 1* (Upper quartile is 63). The *pleasantness* measure for both the algorithms were ranged between 40-60 which falls in the category *Fair*.

<sup>19</sup>The third quartile (designated Q3) also called the upper quartile or the 75th percentile (splits off the highest 25% of data from the lowest 75%)

## 5 Conclusion

The objective of designing an algorithm to compose a melody with user constraints is achieved. The first method of generating a melody and rhythm separately is working well. All the compositions done by the first method sounds *Fair* to the listeners. In specific, the composition with *Including one note* constraint sounds more pleasant than other constraints. On the other hand, *Including several notes* constraint reduces the *pleasantness* of a composition compared to other methods. Adding notes in a random position changes the actual style learned from a reference melody than including a single note. In the remaining categories of constraint such as *Including a given pattern of notes* and *Use only a set of notes* were ranged between 40-60 (*Fair*) in the scale. The performance of a new model HMM with CD shows an interesting result (Figure 45).

The automatic melody generation has demonstrated how AI can be used to generate melodies with constraints. Although it is far from a complete solution to generating *Good* and *Excellent* results, its a good starting point for exploring algorithm composition. It is easy to see that there are endless possibilities for future work on this project.

For the future work I propose:

- In this thesis, I have evaluated 4 musical constraints in *Algorithm 1* (section 3.2.3). By including more constraints it is possible to check how the pleasantness of a composition varies.
- Instead of including notes in a random position(*Include note* and *Include notes*) one can analyze the given notes semantically with the previous notes in the sequence before adding it. This will help in increasing the pleasantness of the composition.
- Similar to *Algorithm 1*, CSP can be used in the *Algorithm 2* (HMM with CD model) to include constraints.
- A MM is playing a key role in *Algorithm 1* and *Algorithm 2*. A second-order Markov chain could also be introduced by considering the current state and also one or more previous states. Higher, *n*th-order chains tend to generate results with a sense of phrasal structure rather than the aimless wandering produced by a first-order system [20].

## References

- [1] Synaptic Architecture-free Neural Network Library. <https://github.com/cazala/synaptic>.
- [2] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. *Advances in Neural Information Processing Systems 26*, pages 2643–2651, 2013.
- [3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag New York Inc, 1 edition, 2006.
- [4] Bongjun Kim and Woon Seung Yeo. Probabilistic prediction of rhythmic characteristics in markov chain based melodic sequences. *2013 International Cryptographic Module Conference (ICMC)*, pages 429–432, 2013.
- [5] Chun-Chi J. Chen and Risto Miikkulainen, editors. *Creating Melodies with Evolving Recurrent Neural Networks*, 2001.
- [6] Jeremy Day-O’Connell. *Pentatonicism from the eighteenth century to Debussy*, volume [v. 46] of *Eastman studies in music*. University of Rochester Press, Rochester, NY, 2007.
- [7] M. O. Duff. Backpropagation and Bach’s 5th cello suite (sarabande). *International Joint Conference on Neural Networks IJCNN*, pages 575–578, 18 Jun 1989.
- [8] Francis Rumsey and Tim McCormick. *Sound and Recording*. Focal Press (Elsevier Ltd.), sixth edition, 2009.
- [9] M. Giraud and S. Staworko. Modeling musical structure with parametric grammars. *International Conference on Mathematics and Computation in Music*, pages 1–12, 2015.
- [10] International Telecommunication Union. Method for the subjective assessment of intermediate quality level of audio systems ITU-R BS.1534-2, 2014.
- [11] Jon McCormack. Grammar based music composition: Complex systems 96: From local interactions to global phenomena. pages 320–336, 1996.
- [12] Jose David Fernández and Francisco Vico. AI methods in algorithmic composition: A comprehensive survey. pages 513–582, 2013.

- [13] Judy A. Franklin. Recurrent neural networks and pitch representations for music tasks. 2004.
- [14] Kevin Jones. *Compositional Applications of Stochastic Processes*, volume 5 of *Computer Music Journal*. The MIT Press, 1981.
- [15] Lars Schmidt-Thieme. Artificial intelligence: Constraint satisfaction problems, summer term 2007.
- [16] Markus Wagner and Tomasz Oliwa. *Composing Music with Neural Networks and Probabilistic Finite-State Machines*, volume 4974 of *Lecture Notes in Computer Science*. Springer, 2008.
- [17] Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing. *Connection Science*, 6:247–280, 1994.
- [18] C. G. Nevill-Manning and I. H. Witten. Compression and explanation using hierarchical grammars. *Computer Journal*, pages 103–116, January 1997.
- [19] G. Nierhaus. *Algorithmic Composition: Paradigms of Automated Music Generation*. Mathematics and Statistics. Springer Science & Business Media, 2009.
- [20] François Pachet, Gabriele Barbieri, and Pierre Roy. Finite-length markov processes with constraints. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, Volume One:635–642, 2011.
- [21] Przemyslaw Prusinkiewicz. Score generation with L–systems. *Proceedings of the 1986 International Computer Music*, pages 455–457, 1986.
- [22] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants: The Virtual Laboratory*. Springer Science-Business Media, 2012.
- [23] Peter M. Todd and D. Gareth Loy. *Music and connectionism*. MIT Press, Cambridge, Mass., 1991.
- [24] William Feller. *An Introduction to Probability Theory and Its Applications*. Wiley 2nd edition (1971).

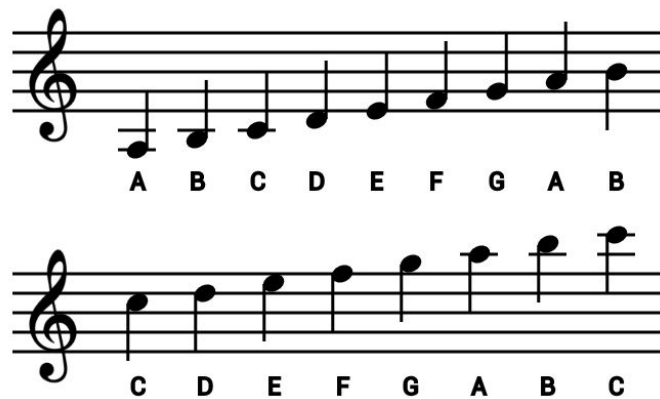
## 6 Appendix

### 6.1 MIDI ASCII Note Table







The chart shows the MIDI numbers which correspond with all the notes from C0 (the lowest) to G9 (the highest). C4 is middle C.

Note	Octave										
	-1	0	1	2	3	4	5	6	7	8	9
C	0	12	24	36	48	60	72	84	96	108	120
C#	1	13	25	37	49	61	73	85	97	109	121
D	2	14	26	38	50	62	74	86	98	110	122
D#	3	15	27	39	51	63	75	87	99	111	123
E	4	16	28	40	52	64	76	88	100	112	124
F	5	17	29	41	53	65	77	89	101	113	125
F#	6	18	30	42	54	66	78	90	102	114	126
G	7	19	31	43	55	67	79	91	103	115	127
G#	8	20	32	44	56	68	80	92	104	116	
A	9	21	33	45	57	69	81	93	105	117	
A#	10	22	34	46	58	70	82	94	106	118	
B	11	23	35	47	59	71	83	95	107	119	













### 6.2 Notes Name and Position in the Staff



### 6.3 Music Symbols and its Property

 Quarter Note	$\frac{4}{4}$ Four-Four Time
 Half Note	$\frac{3}{4}$ Three-Four Time
 Whole Note	$\frac{2}{4}$ Two-Four Time
 Eighth Note	$\frac{6}{8}$ Six-Eight Time
 Beamed Eighth Notes	 Common Time

 Quarter Note	 Treble Clef	 Triplet
 Half Note	 Bass Clef	 Dotted Half
 Whole Note	 Sharp	 Sixteenth Note
 Eighth Note	 Flat	 Repeat sign

### 6.4 Musical Score of *Bee Bee* Folk Song

1 

7 

### 6.5 Melody Generated by HMM with CD Model in the Style of *Green Mountain* (Taiwanese Folk Song)

Three staves of musical notation in 4/4 time, representing the reference score for the Taiwanese folk song 'Green Mountain'. The first staff (labeled '1') contains the melody, starting with a red vertical line. The second staff (labeled '5') and third staff (labeled '8') show accompaniment parts.

Reference Musical Score: *Green Mountain* (Taiwanese folk song)

Three staves of musical notation in 4/4 time, representing the melody generated by HMM with CD model in the style of 'Green Mountain'. The first staff (labeled '1') contains the generated melody, starting with a red vertical line. The second staff (labeled '4') and third staff (labeled '7') show accompaniment parts.

*Green Mountain* (Taiwanese folk song) Style imitated by HMM with CD model

TRITA TRITA-EE 2015:98  
ISSN 1653-5146

[www.kth.se](http://www.kth.se)