Automatic Music Composition
with Simple Probabilistic Generative Grammars

Horacio Alberto García Salas, Alexander Gelbukh, Hiram Calvo, and Fernando Galindo Soria

Abstract—We propose a model to generate music following a linguistic approach. Musical melodies form the training corpus where each of them is considered a phrase of a language. Implementing an unsupervised technique we infer a grammar of this language. We do not use predefined rules. Music generation is based on music knowledge represented by probabilistic matrices, which we call evolutionary matrices because they are changing constantly, even while they are generating new compositions. We show that the information coded by these matrices can be represented at any time by a probabilistic grammar; however we keep the representation of matrices because they are easier to update, while it is possible to keep separated matrices for generation of different elements of expressivity such as velocity, changes of rhythm, or timbre, adding several elements of expressiveness to the automatically generated compositions. We present the melodies generated by our model to a group of subjects and they ranked our compositions among and sometimes above human composed melodies.

Index Terms—Evolutionary systems, evolutionary matrix, generative grammars, linguistic approach, generative music, affective computing.

I. INTRODUCTION

MUSIC generation does not have a definite solution. We regard this task as the challenge to develop a system to generate a pleasant sequence of notes to human beings and also this system should be capable of generating several kinds of music while resembling human expressivity. In literature, several problems for developing models for fine arts, especially music have been noted. Some of them are: How to evaluate the results of a music generator? How to determine if what such a system produces is music or not? How to say if a music generator system is better than other? Can a machine model expressivity?

Different models have been applied for developing automatic music composers; for example, those based on neural networks [15], genetic algorithms [2, 25] and swarms [4] among other methods.

In order to generate music automatically we developed a model that describes music by means of a linguistic approach; each musical composition is considered a phrase that is used to learn the musical language by inferring its grammar. We use a learning algorithm that extracts musical features and forms probabilistic rules that afterwards are used by a note generator algorithm to compose music. We propose a method to generate linguistic rules [24] finding musical patterns on human music compositions. These patterns consist of sequences of notes that characterize a melody, an author, a style or a music genre. The likelihood of these patterns of being part of a musical work is used by our algorithm to generate a new musical composition.

To model the process of musical composition we rely on the concept of evolutionary systems [8], in the sense that systems evolve as a result of constant change caused by flow of matter, energy and information [10]. Genetic algorithms, evolutionary neural networks, evolutionary grammars, evolutionary cellular automata, evolutionary matrices, and others are examples of evolutionary systems. In this work we follow the approach of evolutionary matrices [11].

This paper is organized as follows. In Section II we present works related to automatic music composition. In Section III, we describe our model. In Section IV, we describe an algorithm to transform a matrix into a grammar. In Section V we show how we handle expressivity in our model. In Section VI, we present results of a test to evaluate generated music. Finally, in Section VII, we present some conclusions of our model and future work to improve our model.

II. RELATED WORK

A. Review Stage

An outcome of development of computational models applied to humanistic branches as fine arts like music is generative music or music generated from algorithms. Different methods have been used to develop music composition systems, for example: noise [5], cellular automata [20], grammars [13, 22], evolutionary methods [13], fractals [14, 16], genetic algorithms [1], case based reasoning [19], agents [21] and neural networks [7, 15]. Some systems are called hybrid since they combine some of these techniques. For a comprehensive study please refer to [23] and [17].

Harmonet [15] is a system based on connectionist networks, which has been trained to produce chorale style of J. S. Bach. It focuses on the essence of musical information, rather than restrictions on music structure. Eck and Shmidhuber [7] believe that music composed by recurrent neural networks lacks structure, and do not maintain memory of distant events.

Manuscript received February 10, 2011. Manuscript accepted for publication July 30, 2011.

H. A. García Salas is with the Natural Language Laboratory, Center for Computing Research, National Polytechnic Institute, CIC-IPN, 07738, DF, México (e-mail: itztzin@gmail.com).

A. Gelbukh was, at the time of submitting this paper, with the Waseda University, Tokyo, Japan, on Sabbatical leave from the Natural Language Laboratory, Center for Computing Research, National Polytechnic Institute, CIC-IPN, 07738, DF, México (e-mail: gelbukh@gelbukh.com).

H. Calvo is with the Natural Language Laboratory, Center for Computing Research, National Polytechnic Institute, CIC-IPN, 07738, DF, México (e-mail: hcalvo@ciic.ipn.mx).

F. Galindo Soria is with Informatics Development Network, REDI (e-mail: fgalindo@ipn.mx).
They developed a model based on LSTM (Long Short Term Memory) to represent the overall and local music structure, generating blues compositions.

Kosina [18] describes a system for automatic music genre recognition based on audio content signal, focusing on musical compositions of three music genres: classical, metal and dance. Blackburn and DeRoure [3] present a system to recognize through the contents of a music database, with the idea to make search based on music contours, i.e. in a relative changes representation in a musical composition frequencies, regardless of tone or time.

There is a number of works based on evolutionary ideas for music composition. For example, Ortega et al. [22] used generative context-free grammars for modeling the musical composition. Implementing genetic algorithms they made grammar evolve to improve the musical generation. GenJam [1] is a system based on a genetic algorithm that models a novice jazz musician learning to improvise. It depends on user feedback to improve new compositions through several generations.

Todd and Werner [25] developed a genetic algorithm based on co-evolution, learning and rules. In their music composer system there are male individuals that produce music and female critics that evaluate it to mate them. After several generations they create new musical compositions.

In our approach we focus on the following points:

– The evolutionary aspect—to keep learning while generating;
– Stressing the linguistic metaphor of musical phrases and textual phrases, words and sets of notes;
– Adding expressiveness to achieve a more human aspect;

III. MUSIC GENERATION

A musical composition is a structure of note sequences made of other structures built over time. How many times a musical note is used after another reflects patterns of sequences of notes that characterizes a genre, style or an author of a musical composition. We focus on finding patterns on monophonic music.

A. Linguistic approach

Our model is based on a linguistic approach [9]. We describe musical compositions as phrases made up of sequences of notes as lexical items that represent sounds and silences throughout time. The set of all musical compositions forms the musical language.

In the following paragraphs we define some basic concepts that we will use in the rest of this paper.

Definition 1: A note is a representation of tone and duration of musical sound.

Definition 2: The alphabet is the set of all notes: alphabet = {notes}.

Definition 3: A musical composition $m$ is an arrangement of musical notes: Musical composition = $a_1 \ a_2 \ a_3 \ ... \ a_n$ where $a_i \in \{\text{notes}\}$.

In our research we work with musical compositions $m$ of monophonic melodies, modeling two variables of notes: musical frequencies and musical tempos. We split these variables to form a sequence of symbols with each of them.

Definition 4: The Musical Language is the set of all musical compositions: Musical Language = \{musical compositions\}. For example, having the sequence of notes (frequencies) of musical composition “El cóndor pasa” (the condor passes by):

\[
b \ \ e \ \ d\# \ \ e \ \ f\# \ \ g \ \ f\# \ \ g \ \ a \ \ b \ \ 2 \ \ d \ \ 2 \ \ b \ \ 2 \ \ a \ \ g \ \ e \ \ g \ \ b \ \ e \ \ d \ \ e \ \ f \ \ g \ \ f \ \ a \ \ b \ \ d \ \ 2 \ \ b \ \ 2 \\
\]

We assume this sequence is a phrase of musical language.

B. Musical Evolutionary System

Evolutionary systems interact with their environment finding rules to describe phenomena and use functions that allow them to learn and adapt to changes. A scheme of our evolutionary model is shown in Fig. 1.

![Fig. 1. Model.](image)
Automatic Music Composition with Simple Probabilistic Generative Grammars

Polibits (44) 2011

C uses K to produce a novel musical composition times that extracts musical features and adds this information into K.

To describe our music learning module we need to define C.

For listening of the new music composition there is a function I called musical interpreter or performer that generates the sound.

For music generation is necessary to choose next note. In our model composer function C generates a melodic line based on knowledge represented by cumulative frequency distribution matrix CFM.

### Definition 5: Musical Frequency
Musical Frequency = \{musical frequencies\}
where musical frequencies (mf) are the number of vibrations per second (Hz) of notes.

### Definition 6: Musical Time
Musical Time = \{musical times\}
where musical times (mt) are durations of notes.

### Definition 7: MF is a workspace formed by two matrices. One of them is a frequency distribution matrix (FDM) and the other one is a cumulative frequency distribution matrix (CFM).

Each time a musical composition \( m_i \) arrives, L upgrades FDM. Then it recalculates CFM, as follows:

### Definition 8: Let FrequencyNotes be an array in which are stored the numbers corresponding to a musical composition notes.

### Definition 9: Let \( n \) be the number of notes recognized by the system, \( n \in \mathbb{N} \).

### Definition 10: Frequency Distribution Matrix (FDM) is a matrix with \( n \) rows and \( n \) columns.

Given the musical composition \( m = f_1 f_2 f_3 \ldots f_n \) where \( f_i \in \text{FrequencyNotes} \). The learning algorithm of L to generate the frequency distribution matrix FDM is:

\[
\forall i \in [1,n], \exists n \in \mathbb{N}, FDM_{i,j} = \text{FDM}_{i,j+1}, \quad \text{where } FDM_{i,j+1} \in \text{FDM}.
\]

### Definition 11: Cumulative Frequency Distribution Matrix
CFM is a matrix with \( n \) rows and \( n \) columns.

The algorithm of L to generate cumulative frequency distribution matrix CFM is:

\[
\forall i \in [1,n], \exists j \in [1,n], \exists n \in \mathbb{N}, \quad \text{CFM}_{i,j} = \sum_{k=1}^{j} \text{FDM}_{i,k}
\]

### Definition 12: \( T_i \) and \( T \).
Let \( T_i \) to be an element where it is store the total of cumulative frequency sum of each \( i \) row of FDM.
\[ \forall i \in [1,n], \exists a \in \mathbb{N}, \ T_i = \sum_{k=1}^{a} FDM_{i,k} \]

Let \( T \) be a column with \( n \) elements where it is store the total of cumulative frequency sum of \( FDM \).

Note generation algorithm:

```plaintext
while (not end) {
    \( p = \text{random}(T_i) \)
    while (CFM_{i,j} < p)
        \( j = j + 1 \)
    \( \text{next note} = j \)
    \( i = j \)
}
```

E. Example

Let us take the sequence of frequencies of musical composition “El cóndor pasa”:

\[
\begin{array}{cccccccc}
    b&d&c&f&g&f&g&a \\
    3&2&2&3&1&1&4&2 \\
    d&3&2&3&1&1&4&2 \\
    e&2&2&3&1&1&4&2 \\
    f&6&2&2&1&1&4&2 \\
    g&6&2&2&1&1&4&2 \\
    a&3&2&3&1&1&4&2 \\
    b&3&2&3&1&1&4&2 \\
    c&6&6&6&6&6&6&6 \\
    e&6&6&6&6&6&6&6 \\
    b&3&2&3&1&1&4&2 \\
\end{array}
\]

Fig. 2. Frequency distribution matrix FDM.

We apply the algorithm of L to calculate cumulative frequency distribution matrix CFM of Fig. 3 from frequency distribution matrix FDM of Fig. 2. Then we calculate each \( T_i \) of \( T \) column.

For generation of a musical composition we use note generator algorithm. Music generation begins by choosing the first composition note. \( S \) row of matrix of Fig. 3 contains all possible beginning notes. In our example only the \( b \) note can be chosen. Then \( b \) is the first note and the \( i \) row of \( CFM \) which we use to determine second note. Only the \( e \) note can be chosen after the first note \( b \).

So the first two notes of this new musical melody are \( m_{i=1} = \{ b, e \} \). Applying note generator algorithm to determine third note: We take the value of column \( T_e = 9 \). A \( p \) random number between zero and 9 is generated, suppose \( p = 6 \). To find next note we compare \( p \) random number with each non-zero value of \( e \) row until one greater than or equal to this number is found. Then column \( g \) is the next note since \( M_{e,g} = 9 \) is greater than \( p = 6 \). The column \( j = g \) is where it is stored this number that indicates the following composition note and the following \( i \) row to be processed. The third note of musical composition \( m_{i=1} \) is \( g \). So \( m_{i=1} = \{ b, e, g \} \). Then to determine the fourth note we must apply the note generator algorithm to \( i = g \) row.

Since each non-zero value of \( i \) row represents notes that used to follow \( i \) note, then we will generate patterns according to probabilities learned from musical compositions examples.

```
<table>
<thead>
<tr>
<th>b</th>
<th>d</th>
<th>c</th>
<th>f</th>
<th>g</th>
<th>a</th>
<th>b</th>
<th>e</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
```

Fig. 3. Cumulative frequency distribution matrix CFM.

IV. MATRICES AND GRAMMAR

Our work is based on a linguistic approach and we have used a workspace represented by matrices to manipulate music information. Now we show that this information representation is equivalent to a probabilistic generative grammar.

There are different ways to obtain a generative grammar \( G \). From frequency distribution matrix FDM and total column \( T \), it is possible to construct a probabilistic generative grammar.

Definition 13: MG is a workspace formed by FDM and a probabilistic grammar \( G \).

To generate a grammar first we generate a probability matrix \( PM \) determined from frequency distribution matrix FDM.

Definition 14: Probability Matrix (PM) is a matrix with \( n \) rows and \( n \) columns.

The algorithm to generate probability matrix \( PM \) is:

\[ \forall i \in [1,n], \forall j \in [1,n], \exists FDM_{i,j} \neq 0 \Rightarrow PM_{i,j} = FDM_{i,j} / T_i \]

There is a probabilistic generative grammar \( G \{ V_n, V_t, S, P, Pr \} \) such that \( G \) can be generated from \( PM \). \( V_n \) is the set of nonterminals symbols, \( V_t \) is the set of all terminal symbols or alphabet which represents musical composition notes. \( S \) is the axiom or initial symbol, \( P \) is the set of rules generated and \( Pr \) is the set of rules probabilities represented by values of matrix \( PM \).

For transforming the PM matrix in a grammar we use the following algorithm:
Automatic Music Composition with Simple Probabilistic Generative Grammars

1. Build the auxiliary matrix AM from PM:
   a. substitute each row i tag of PM with a nonterminal symbol \( X_i \) except S row which is copied as it is
   b. substitute each column j tag by its note \( f_j \) and a nonterminal symbol \( X_j \)
   c. copy all values of cells of matrix PM into corresponding cells of matrix AM

2. For each row \( i \) and each column \( j \) such that \( AM_{ij} \neq 0 \)
   a. \( i \) row corresponds to grammar rule \( X_i \)
   b. \( j \) column corresponds to a terminal symbol \( f_j \) and a nonterminal symbol \( X_j \) with probability \( p_{ij} \)

Then rules of grammar \( G \) are of the form \( X_i \rightarrow f_j \ X_j \) (\( p_{ij} \)). This is a grammatical representation of our model. For each music composition \( m \), a MG, the workspace formed by FDM and grammar \( G \), can be recursively generated.

\[ L(m_1...L(m_2, L(m_1, L(m_0, MG_0)))) = MG_{i+1} \]

A. Example

From frequency distribution matrix FDM of Fig. 2 it is generated probability matrix PM of Fig. 4.

\[ \begin{array}{cccccccccc}
\text{b} & \text{d} & \text{e} & \text{f} & \text{g} & \text{a} & \text{b} & \text{d} & \text{e} & \text{g} \\
\text{S} & 1 &  &  &  &  &  &  &  &  \\
\text{b} &  & 1 &  &  &  &  &  &  &  \\
\text{d} &  &  & 1 &  &  &  &  &  &  \\
\text{e} & 19 & 29 & 29 & 39 & 19 &  &  &  &  \\
\text{f} &  &  &  &  &  & 1 &  &  &  \\
\text{g} &  &  &  &  &  &  & 611 & 211 & 211 \\
\text{a} &  &  &  &  &  &  &  & 35 & 25 \\
\text{b} &  &  &  &  &  &  & 19 & 39 & 29 \\
\text{d} &  &  &  &  &  &  & 612 & 612 &  \\
\text{e} &  &  &  &  &  &  &  &  &  \\
\text{g} &  &  &  &  &  &  &  &  & 1 \\
\end{array} \]

Fig. 4. Probability matrix PM.

\[ \begin{array}{cccccccccc}
\text{b}_{X_1} & \text{d}_{X_1} & \text{e}_{X_1} & \text{f}_{X_1} & \text{g}_{X_1} & \text{a}_{X_1} & \text{b}_{X_1} & \text{d}_{X_1} & \text{e}_{X_1} & \text{g}_{X_1} \\
\text{S} & 1 &  &  &  &  &  &  &  &  \\
\text{X}_1 & 1 &  &  &  &  &  &  &  &  \\
\text{X}_2 & 1 &  &  &  &  &  &  &  &  \\
\text{X}_3 & 19 & 29 & 29 & 39 & 19 &  &  &  &  \\
\text{X}_4 &  &  &  & 1 &  &  &  &  &  \\
\text{X}_5 & 611 & 211 & 211 &  &  &  &  & 111 \\
\text{X}_6 &  &  &  &  & 35 & 25 &  &  &  \\
\text{X}_7 & 19 & 39 & 29 & 39 &  &  &  &  \\
\text{X}_8 & 612 & 612 &  &  &  &  &  &  \\
\text{X}_9 &  &  &  &  &  &  & 1012 & 2012 &  \\
\text{X}_{10} &  &  &  &  &  &  &  &  & 1 \\
\end{array} \]

Fig. 5. Auxiliary matrix AM.

From matrix PM of Fig. 4 the auxiliary matrix AM of Fig. 5 is generated. From given AM matrix of Fig. 5 We can generate grammar \( G\{V_n, V_t, S, P, Pr\} \) Where \( V_n=\{S, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}\} \) is the set of non-terminals symbols. \( V_t=\{b, d_s, e, f_s, g, a, b_2, d_2, e_2, g_2\} \) is the set of all terminal symbols or alphabet. S is the axiom or initial symbol. \( Pr \) is the set of rules probabilities represented by values of matrix AM. Rules P are listed in Fig. 6.

\[ \begin{align*}
S & \rightarrow b \ X_1 (1) \\
X_1 & \rightarrow c \ X_1 (1) \\
X_1 & \rightarrow c \ X_1 (1) \\
X_1 & \rightarrow b \ X_1 (1/9) | d_s \ X_2 (2/9) | f_s \ X_4 (2/9) | g \ X_5 (3/9) | b_2 \ X_6 (1/9) \\
X_1 & \rightarrow g \ X_1 (1) \\
X_1 & \rightarrow c \ X_6 (6/11) | f_s \ X_4 (1/11) | a \ X_2 (2/11) | c_2 \ X_8 (1/11) \\
X_6 & \rightarrow g \ X_3 (5/11) | b_2 \ X_5 (2/5) \\
X_7 & \rightarrow g \ X_3 (1/9) | a \ X_2 (3/9) | d_s \ X_6 (2/9) | c_2 \ X_8 (3/9) \\
X_8 & \rightarrow b_2 \ X_6 (6/12) | g_2 \ X_2 (6/12) \\
X_9 & \rightarrow d_2 \ X_8 (10/12) | g_2 \ X_2 (2/12) \\
X_{10} & \rightarrow c_2 \ X_8 (1) \\
\end{align*} \]

Fig. 6. Probabilistic generative grammar.

V. EXPRESSIVITY

Expressivity can be regarded as a mechanism that displays transmission and interpretation vividness of feelings and emotions. For example fear in front of a threat. Physical factors interfere like cardiac rhythm, changes in respiratory system, in endocrine system, in muscular system, in circulatory system, secretion of neurotransmitters, etc. Another important factor is empathy which is the capacity of feelings and emotions recognition in others [6]. It is out of our research to explain how these physical changes are made or how empathy takes place among living beings. We just simulate expressivity in music generation.

A. Expressivity within our Model

Music can be broken down into different functions that characterize it like frequency, time and intensity. So each note of a melody is a symbol with several features or semantic descriptors that give the meaning of a long or short sound, low, high, intense, soft, of a guitar or of a piano.

With our model is possible to represent each of these variables using matrices or grammars that reflect their probabilistic behavior. In this paper we have presented how to model frequency and time. We can build an intensity matrix the same way. With more variables more expressivity the generated music will reflect.

Using our model we can characterize different kinds of music based on its expressivity, for example in happy music or sad music. Besides we have the possibility of mixing features of distinct kinds of music, for example frequency functions of happy music with time functions of sad music. Also we can combine different genres like classic times with rock frequencies. So in addition of generating music we can invent new genres and music styles.

VI. RESULTS

In order to evaluate whether our algorithm is generating music or not, we decided to conduct a Turing-like test. Participants of this test had to tell us if they like music generated by our model, without knowing that it was automatically music generated. This way we sought the answer to two questions: whether or not we are doing music and whether or not our music is pleasant.
We compiled 10 melodies, 5 of them generated by our model and another 5 by human composers and we asked human subjects to rank melodies according to whether they liked them or not, with numbers between 1 and 10 being number 1 the most they liked. None of subjects knew about the order of music compositions. These 10 melodies were presented as in Table I.

<table>
<thead>
<tr>
<th>ID</th>
<th>Melody</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Zanya</td>
<td>(generated)</td>
</tr>
<tr>
<td>B</td>
<td>Fell</td>
<td>Nathan Fake</td>
</tr>
<tr>
<td>C</td>
<td>Atucin</td>
<td>(generated)</td>
</tr>
<tr>
<td>D</td>
<td>Idiot</td>
<td>James Holden</td>
</tr>
<tr>
<td>E</td>
<td>Ciclos</td>
<td>(generated)</td>
</tr>
<tr>
<td>F</td>
<td>Dali</td>
<td>Astrix</td>
</tr>
<tr>
<td>G</td>
<td>Ritual Cibernético</td>
<td>(generated)</td>
</tr>
<tr>
<td>H</td>
<td>Feelin’ Electro</td>
<td>Rob Mooney</td>
</tr>
<tr>
<td>I</td>
<td>Infinito</td>
<td>(generated)</td>
</tr>
<tr>
<td>J</td>
<td>Lost Town</td>
<td>Kraftwerk</td>
</tr>
</tbody>
</table>

We presented this test to more than 30 participants in different places and events. We sought that the characteristics of these participants were as varied as possible (age, gender and education), however most of them come from a related IT background. Test results were encouraging, since automatically generated melodies were ranked at 3rd and 4th place above human compositions. Table II shows the ranking of melodies as a result of the Turing-like test we developed.

<table>
<thead>
<tr>
<th>ID</th>
<th>Ranking</th>
<th>Melody</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>Fell</td>
<td>Nathan Fake</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>Idiot</td>
<td>James Holden</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>Atucin</td>
<td>(generated)</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>Zanya</td>
<td>(generated)</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>Dali</td>
<td>Astrix</td>
</tr>
<tr>
<td>H</td>
<td>6</td>
<td>Feelin’ Electro</td>
<td>Rob Mooney</td>
</tr>
<tr>
<td>J</td>
<td>7</td>
<td>Lost Town</td>
<td>Kraftwerk</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
<td>Ciclos</td>
<td>(generated)</td>
</tr>
<tr>
<td>G</td>
<td>9</td>
<td>Ritual Cibernético</td>
<td>(generated)</td>
</tr>
<tr>
<td>I</td>
<td>10</td>
<td>Infinito</td>
<td>(generated)</td>
</tr>
</tbody>
</table>

VII. CONCLUSIONS AND FUTURE WORK

We proposed an evolutionary model based on evolutionary matrices for musical composition. Our model is learning constantly, increasing its knowledge for generating music while more data is presented. It does not need any predefined rules. It generates them from phrases of the seen language (musical compositions) in an unsupervised way.

As we shown, our matrices can be expressed as probabilistic grammar rules, so that we can say that our systems extracts grammar rules dynamically from musical compositions. These rules generate a musical language based on the compositions presented to the system. These rules can be used to generate different musical phrases, meaning new musical compositions. Because the probabilistic grammars learned can generalize a language beyond the seen examples of it, our model has what can be called innovation, which is what we are looking for music creation, while keeping the patterns learned from human music.

As a short-term future work we plan to characterize different kinds of music, from sad to happy, or from classic to electronic in order to find functions for generating this kind of music. We are also developing the use of other matrices to consider more variables involved in a musical work, such as velocity, fine-graded tempo changes, etc., thus adding more expressivity to the music created by our model.

ACKNOWLEDGEMENTS

The work was done under partial support of Mexican Government (CONACyT 50206-H, SIP-IPN 20113295, COFAA-IPN, PIFI-IPN, SNI).

REFERENCES


