Towards a Fitness Function for Musicality using LPM

Sahar Arshi, Darryl N. Davis

Department of Computer Science, University of Hull, UK.
{ S.Arshi@2014., D.N.Davis@ }hull.ac.uk

Abstract. In this article the features of Liquid Persian Music (LPM) software in employing cellular automata (CA) evolution is briefly described. Controlling synthesizer parameters by means of pattern matching over the emergent nature of CA is an important characteristic of LPM. The hypothesis is that LPM will enable us to explore new dimensions of music composition. Here, the main focus is to analyze the LPM output voices in the search for finding proper tools for enhancing them in a musical way. This paper focuses on applying Zipf’s law on various configurations of produced audio to fulfill aesthetic goals. The nature of a fitness function for measuring the pleasantness of music has been targeted as a concluding remark for future research.

Key words: Cellular Automata, Liquid Persian Music, Synthesis Toolkit, Zipf’s Law, Computer Music, Artificial Music Composition, Evolutionary Algorithms, Fitness Function.

1 Introduction

The advent of cellular automata originally dates back to 1940s, when Von Neumann was looking forward to develop a system capable of reproduction, comparable in certain respects with biological breeding [1, 2]. Cellular automata was studied as a dynamical system in 1960s [3]. Cellular Automata are discrete dynamical systems whose global intricate behaviour is determined by reciprocal influence of elementary identical individuals [3–6]. Each cell has k finite states at time t, and all the cells evolve simultaneously. The state of a cell at time t depends on its state and its neighbors’ states at time t-1 [7]. In the one dimensional elementary CA (which is the subject of this study), the permutations of each cell with its two adjacent neighbors specifies eight situations. Once allocated to binary states, the selection of one of the 256 local transition rules specify the CA evolution.

Wolfram studies on CA recognize four classes of behaviour, namely, fixed, cyclic, chaotic, and complex [3]. Li and Packard [8] subdivided the second class to three further subgroups, namely heterogeneous, periodic with intervals greater than one, and locally chaotic.

Cellular automata exhibiting myriad genres of behaviour have been targeted as a creative tool for artists. The generated patterns by CA possess self-similar characteristics known as 1/f noise or pink noise. In music composition pink noise yields
a pleasing balance between regularity and abrupt variations [9]. Various cellular automata musical systems have been designed since the advent of this field. Amongst these systems, CAMUS and Chaosynth [10, 11] have gained popularity. CAMUS exploits Game of Life and Demon Cyclic Space, and uses a Cartesian space mapping to MIDI for achieving the musical triplets. Chaosynth is a cellular automata sound producer based on the generation of sound granules.

Liquid Persian Music is a CA based toolkit for exploring various musical possibilities. This paper expands the capabilities of LPM and establishes connections with genetic algorithms. The article consists of five sections. In the second section a brief overview of LPM is given, followed by an investigation throughout the output of the software. In the third part the Zipf’s law has been applied on the output distributions of cellular automata as a musical aesthetical measurement. Zipf’s law is able to recognize the pink noise distribution in data series. The fourth section focuses on further experiments and clarifies some future research directions for LPM.

2 LPM Project Features Overview

Liquid Persian Music is an auditory software tool developed at the University of Hull. LPM explores the idea of artificial life systems in producing voices. The software takes advantage of Synthesis Toolkit (STK) [12] for implementing the physical model of a stringed musical instrument. A model of which its parameters are controlled by defined pattern matching rules. Pattern matching rules classify output from CA and update the parameters of the synthesiser to yield musical effects [13, 14]. OpenAl library is responsible for propagating the producing voices.

The elementary CA used in LPM consists of an assembly of cells arranged in a one dimensional array. In every time step of CA evolution, the pattern matcher extracts the difference between the consecutive generations. Twenty different pattern matching rules have been defined in this software as well as Dice’s coefficient, and Jaccard similarity. The obtained values from pattern matchers are then fed into synthesizer for producing sounds. Some of the synthesizer parameters include ADSR envelope, loop gain, and the musical instrument string length for defining frequency. Further information about the software can be found in [15].

An important point needed to be mentioned is that the aggregation of a CA rule and a pattern matching rule on each of the synthesizer elements does not produce a single note but a collection of notes which are referred to as voices throughout this paper.

Studying the musical behaviour derived from one-dimensional (1D) CA does not require the investigation of the 256 rules’ behaviours. The rule space can be reduced to 88 fundamental behaviours [16] by applying conjugate, reflection, and both transformations together [3], since they lead to rule sets with inherently equivalent behaviour (The interested reader is referred to [1] for formulation of conjugate and reflection transformations and how they are applied to find equivalent CA rules). The 88 1D CA rule behaviours, 7 defined synthesizer parameters, together with 20 pattern matching rules, expand the number of voices to $88 \times 20^7$.

The musical output of LPM has been investigated through two different approaches: the decision of a human subject and the plots of pattern matcher outputs. The values of
20 pattern matchers for 10000 generations of 88 CA rules have been extracted. The initial seeds for CA are selected randomly. These values change the parameters of the synthesizers within the acceptable defined ranges for parameters.
Three sorts of behaviour have been observed in the experiments with CA and our pattern matchers in LPM (It needs to be stated that this classification into three behaviours is independent from Wolfram classes and are based on the conducted experiment). In the first group, the evolution converges to a homogeneous state, in which no differences can be measured through consecutive generations, resulting in a uniform sound. Amongst this behaviour are the first Wolfram class and first subgroup of the second class. The second behaviour consists of an ordered oscillation between two or more values. Some examples include: rules 6, 178, 26, 73 from the third subgroup of the second Wolfram class; rules 22, 30 from the third Wolfram class; and rule 110 from fourth Wolfram group. The third pattern of behaviour is observed as a disordered fluctuation between a large number of values. Among these, rule numbers 14, 28, 51 from classes 3 and 2 are notable. Figure 1 illustrates some of these behaviours for CA rule numbers 11, 22, 27, 51, 38, 110, 168 (from Wolfram classes 2, 3, 2, 2, 2, 4, 1 respectively). Subfigure (d) shows the homogeneous behaviour, while subfigures (a), (b), (c), (e), (f), (g) suggest almost an ordered oscillatory pattern within different ranges. Subfigure (h) depicts disordered fluctuation behaviour. These three obtained behaviours from our pattern matchers provide insight into the nature of the musical behaviour of our system.

Investigation on LPM output and Zipf's law

Zipf’s law characterizes the scaling attributes of many natural effects including physics, social sciences, and language processing. Events in a dataset are ranked (descending order) according to their prevalence or importance[17]. The rank and frequency of occurrence of the elements are mapped to a logarithmic scale, where linear regression is applied to the events graph. The slope and $R^2$ measurements demonstrate to what extent the elements conform to Zipf’s law. A linear regression slope of -1 indicates Zipf’s ideal. Zipf’s law can be formulated as $F \sim r^{-a}$, in which $r$ is the statistical rank of the phenomena, $F$ is the frequency of occurrence of the event, and $a$ is close to one in an ideal Zipfian distribution. The frequency of occurrence of an event is inversely proportional to its rank [17]. $P(f) = \frac{1}{f^n}$ is another way to express the Zipf’s law. $P(f)$ is the probability of occurrence of an event with rank $f$. In case of $n = 1$(Zipf’s ideal), the phenomenon is known as pink noise. The cases of $n = 0$ and $n = 2$ are called white and brown noises, respectively [17].

Voss and Clarke [18] have observed that the spectral density of audio is $1/f$ like and is inversely proportional to its frequency. They devised an algorithm which used white, pink, and brown noise sources for composing music. The results show that pink noise is more musically pleasing due to its self-similarity characteristics, the white noises are too random, and the brown noises are too correlated [17] producing a monotonous sound.
In the musical domain, Zipf’s metrics are obtained by enumerating the different musical events’ frequency of occurrence and plotting them in a log-log scale versus their rankings. The slope of Zipf’s distribution differs from \(-\infty\) to 0. The R-squared value is between 0 and 1. The decreasing of the slope to minus infinity speaks of the level of monotonicity increased. Various publications explore the utilization of Zipf’s law in musical data analysis and composition. Previous experiments [17, 19] show its successful application in capturing significant essence from musical contents. In [17] the Zipf’s metrics consist of simple and fractal metrics. The simple metrics include seventeen features of the music as well as the ranked frequency distributions of pitch, and chromatic tone. Fractal metrics gives a measurement of the self-similarity of the distribution. These metrics were later used to train neural networks to classify musical styles and composers, with an average success rate of over ninety percent; demonstrating that Zipf’s metrics extract useful information from music in addition to determining the aesthetical characteristics of music pieces [17].

Here, the values obtained from the pattern matching rules for the 88 CA are used to study the behaviour of LPM in terms of Zipfian distribution. These measurements are ranked in compliance with their redundancy. After applying linear regression on the rank and frequency of occurrence of the set values, the slope and R-squared measurements are obtained.

The values of the 20 pattern matching rules have been extracted from over 10000 generations of CA evolution. Figure 2 shows linear regression lines fitted to Zipfian data distribution of LPM outputs for specific CA rule numbers. Table 1 depicts the slopes obtained from applying Zipf’s law on the dataset after the five hundredth generation to ten thousandth generations for these rules. The reason for this time delay is to let the CA to reach stability in its evolution after the initial state. The coloured cells in the figures are obtained after a scrutinized comparison between the output graphs and Zipfian data. Most of the parameters attained in the table for Zipfian slopes are as expected according to the output graphs from the previous section. The brown cells indicate the conditions where the distribution follows Zipf’s law. The yellow cells show the distributions with minus infinity slopes. The dark green cells illustrate monotonous outputs. The light green cells demonstrate the situations in which the author would expect Zipfian ideal parameters, despite the obtained parameters being far from ideal. The light blue cells depict the cases where the parameters have nearly ideal Zipfian distribution; however, the graphs in the previous section (Fig.1) show the contrary. In this case the Zipf’s distribution is not sufficient for showing the musicality of the data distribution, due to the limited diversity of events. The remaining cells (dark purple) show slopes near to zero, with the graphs in Fig. 1 suggesting tedious outputs.
Fig. 2. Zipfian distributions for (a) rule 168: pattern 4, (b) rule 38: pattern 5, (c) rule 38: pattern 16, (d) rule 110: pattern 5, (e) rule 11: pattern 13, (f) rule 22: pattern 1, (g) rule 27: pattern 2, (h) rule 51: pattern 13.

The remainder of this section presents an analysis of the colour coding given for table 1. A confusion matrix allows good visualisations over the performance of a classifying algorithm. It shows how the predicted and actual classes overlap each other due to performance of the algorithm [20].

The target classes are obtained by human decision. The human labelling was performed according to the studies on the graphs as in figure 1 and also by interactive auditory tests from LPM outputs. The predicted classes are gained by labelling them in terms of Zipf’s law musicality; the slopes which are mostly near the Zipf’s ideal (-1) are categorized as being musical. Here, it has been decided that Zipf slopes between -2.1 and -0.6 are expected to be musical.

The confusion matrix in table 2 is defined as follows, with the related colour coding and observed behaviour in Wolfram classes and Li and Packard subclasses shown in table 3:

- TP (True Positive) items are expected to be musical and their Zipf slopes show they have musical attributes (brown cells).
- FP (False Positive) indicate items not expected to be musical but their Zipf’s graphs show they have musical attributes (light blue cells).
- TN (True Negative) are items not labelled as musical and are correctly classified outside of the musical group (yellow, dark green, and purple cells).
- FN (False Negative) are expected to have a musical output, however, Zipf’s metric is not showing that (light green cells).
Table 1. Zipfian slopes for some examples, the first column on left depicts CA rules and the first row stand for pattern matching rules (please refer to table 3 for colour coding).

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
</tr>
</thead>
<tbody>
<tr>
<td>168</td>
<td>-2.53</td>
<td>-4.43</td>
<td>-3.12</td>
<td>-2.87</td>
<td>-2.53</td>
<td>-2.53</td>
<td>1.1f</td>
<td>-4.43</td>
<td>-0.61</td>
</tr>
<tr>
<td>11</td>
<td>1.33</td>
<td>-1.21</td>
<td>-2.52</td>
<td>-2.75</td>
<td>-1.94</td>
<td>-2.19</td>
<td>-1.96</td>
<td>-2.00</td>
<td>-0.75</td>
</tr>
<tr>
<td>27</td>
<td>-1.54</td>
<td>-3.37</td>
<td>-1.57</td>
<td>-3.15</td>
<td>-1.60</td>
<td>-3.30</td>
<td>-1.61</td>
<td>-3.13</td>
<td>-1.99</td>
</tr>
<tr>
<td>38</td>
<td>-3e-4</td>
<td>-1.18</td>
<td>-0.11</td>
<td>-1.74</td>
<td>-1.22</td>
<td>-0.93</td>
<td>-1.36</td>
<td>-1.19</td>
<td>-3e-4</td>
</tr>
<tr>
<td>51</td>
<td>-3.30</td>
<td>-3.18</td>
<td>-3.19</td>
<td>-2.84</td>
<td>-3.28</td>
<td>-3.28</td>
<td>-3.34</td>
<td>-3.00</td>
<td>-1.95</td>
</tr>
<tr>
<td>22</td>
<td>-2.60</td>
<td>-2.86</td>
<td>-2.96</td>
<td>-3.07</td>
<td>-3.26</td>
<td>-2.94</td>
<td>-3.44</td>
<td>-3.29</td>
<td>-1.64</td>
</tr>
<tr>
<td>110</td>
<td>-3e-4</td>
<td>-3e-4</td>
<td>-3e-4</td>
<td>-3e-4</td>
<td>-Inf</td>
<td>-Inf</td>
<td>-Inf</td>
<td>-Inf</td>
<td>-3e-4</td>
</tr>
</tbody>
</table>

Table 2. Confusion Matrix

<table>
<thead>
<tr>
<th>Musicality = True (Positive)</th>
<th>Musicality = False (Negative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP (254 items)</td>
<td>FP (30 items)</td>
</tr>
<tr>
<td>FN (207 items)</td>
<td>TN (1269 items)</td>
</tr>
</tbody>
</table>

Table 3. Colour coding interpretations in terms of confusion matrix.
The Accuracy ($\frac{TP+TN}{TP+FP+TN+FN}$), sensitivity ($\frac{TP}{TP+FN}$), and specificity ($\frac{TN}{TN+FP}$) [20] of the classifier are calculated as 87, 55, 98% respectively. High accuracy suggests Zipf classifier is likely to predict the musical and non-musical samples correctly. Middle range rate for sensitivity indicates its average ability for identifying musical elements. The specificity shows its success rate in correctly excluding non-musical individuals.

4 Applying Zipf’s Law on a Crafted Sequence of Voices

Regarding the outputs of previous stage, it was assumed that juxtaposing a collection of voices yield Zipfian slopes. On this account a 100 sequences, consisting of a random selection of voices, were defined in Matlab. As a first stage, the length of voices were randomly chosen amongst the total number of CA generations. The obtained Zipfian slopes range from -3.06 to -1.44 with 0.81 and 0.66 as their respective r-squared values. Figure 3 illustrate the Zipfian distributions for these cases and for a third case with an ideal fit (graph c).
In the consequent stages the random selection of whole generations was performed with a smaller sample size (max 20). By decreasing the length of voices, the monotonicity of those characteristic elements will be lowered, which can produce acceptable range of numbers for the Zipfian distribution. In this last experiment the min and max achieved Zipfian slopes are -1.56 and -0.89; and the r-squared values are 0.92 and 0.91. Some examples are depicted in figure 4, with the case of maximum r-squared value in graph c. The results imply that crafting voices with the appropriate characteristic beside each other give musically pleasant outputs in LPM. Tailoring the best musical combinations from the possible space of emerging voices is a task which is appropriate for the use of evolutionary algorithms.
Fig. 4. Zipfian distributions for the first experiment with voices of random length up to 20 for (a) $s = -0.89, r^2 = 0.91$, (b) $s = -1.56, r^2 = 0.96$, (c) $s = -1.36, r^2 = 0.97$.

If Pattern Matching over CA defines musical structures, then the application of Genetic Algorithms (GA) could tailor the musical sequences to make them aesthetically acceptable to audience and conform to musical rules. The search to find optimal solutions is guided by assigning higher fitness to competent individuals. Applying Genetic Algorithms on search and optimization of musical sequences has special requirements. For example, defining the search space; specifying the musical knowledge and rule representation; and the choice of an appropriate fitness function [21]. Since there are infinite possibilities for producing music; it is necessary to define suitable constraints to limit the search space. An idea could be to keep those melodies which conform to ideal Zipfian slopes. The second issue would be to clarify how music composition evolution is effected by GA progression by means of rule representation.

5 Conclusion and Future Directions

In this paper, pattern matching over CA evolution was employed as a controller for the parameters of a synthesizer. The outputs of the software have been explored through graphs and auditory tests. The output distributions have been investigated regarding
their compliance with Zipf’s law. The results are categorized according to the expectations from studying the behaviours. In the third experiment collections of voices are sequenced; some with pleasing Zipfian slopes. Some of the CA and rules do not contribute to musical outputs by themselves, but, experiments with crafted pieces have shown that the proper combination of those elements can give in acceptable musical results.

The measured Zipfian slopes characterize the global features [17] of the produced music. Attention was kept on one dimension of the synthesizer (frequency) and on global measurement of aesthetics throughout this study, for simplicity. Although, Zipf’s law can be considered a good approach for investigating the pleasantness of the output melody, there are other approaches which can be taken into account in our research. Other pattern matching rules are under investigation to assist us in attaining some other aspects of visual aesthetics from cellular automata evolution [22]. By extending the dimension of the produced sound, our future direction will investigate different possibilities of ordering those elements horizontally and vertically for yielding melody and harmony regarding aesthetical measurements.

In the next step the author is to investigate the feasibility of different approaches for crafting an appropriate fitness function and whether it will be automatic or interactive [9]. In interactive mode, users train evaluators which are applied as fitness functions in the evolutionary system. Fitness bottleneck is a challenge in this type of evaluation [9]. There are various automatic fitness functions for the application of GA in music composition given a set of musical training sets. These include a model based on Zipf’s law [19], Markov model, or artificial neural networks. Fitness functions based on the distance from an original melody [23] are other approaches; as are using constraint satisfaction with genotypes conforming to music theory [24], and the weighted sum of statistical compositional features [25].

References