FINITE STATE MODELS FOR GENERATION OF HINDUSTANI CLASSICAL MUSIC

Dipanjan Das Monojit Choudhury
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
West Bengal, INDIA 721302
Email: {dipanjan, monojit}@mla.iitkgp.ernet.in

Abstract

We take a look at a model of computer aided generation of Hindustani Classical music. The system implemented using the model is essentially a simulation of the 'Arohana' and 'Avarohana' of a particular 'Raga'. The artificial generator creates note sequences using hand-crafted bigram finite state models which conform to the grammar of a Raga. We also describe a procedure that intends to measure the quality of the artificially generated compositions by comparing them with random sequences which have been generated from unigram models. The measure makes it evident that the computer generated sequences are better than the random ones. However, the compositions are nowhere near the quality of human created ones and therefore this leaves a lot of scope of further research in similar areas.

Keywords: computer music generation, finite state models, Hindustani Classical music, Raga.

1. Introduction

Computer music generation can be defined to be the process of creating music by artificial means, specifically with the help of computers. Artificial generation of music has many implications in Hindustani Classical Music. The structure of different *Ragas*, their specific characteristics and their effect on the human cognitive system can be explored by generation of music sequences artificially. Moreover, the structure of a *Raga* follows a fixed grammar or a fixed sequence of notes that govern the progression in some specific direction. This fixed grammar of the basic unit of Indian Classical Music enables one to model its structure computationally.

There are several applications of an artificial composer of Hindustani Classical Music. The use of a generative system would be of great help to performers who want to explore new avenues of improvisations in some particular *Raga* of interest. A computer generated composition may have radically different qualities from a human created one, in the same *Raga.* There may be interesting combinations of notes in *vistaras* or *tanas,* which probably would be difficult for a human being to think of. Such a system would thus help in finding out new possibilities in a *Raga,* which have not been explored by performers and learners. Another motivation in developing a generator using a particular model is the discovery of the characters of different *Ragas*, especially the complexity of a computational framework (grammar) that can fully specify the entire musicality of a *Raga*. For example, A *Raga* may produce the effect it normally does on an audience not only because of the structure of its *Arohana* and its *Avarohana,* but because of some standard phrases used in its rendering at times. Such *Ragas* may not be totally specified by simply modelling its Arohana and Avarohana, and might call for more detailed treatment. Many such dimensions of Hindustani Classical Music can be explored through a generative system.

In this work we take a look at generation of compositions in certain *Ragas* using hand crafted probabilistic finite-state automata. The finite state models help in deciding probable transitions from a particular note to another that conform to the grammar of the concerned *Raga.* In future such models can be automatically trained from a sufficiently large set of existing compositions in a *Raga.* We also describe a method for testing the goodness of a generated composition, which shows that the generator has outperformed a random composer, but is inferior to human composers.

The paper is organized as follows. Section 2 looks at previous work on computer music generation over the years. The mathematical model used in generating compositions is described in Section 3. Section 4 deals with the implementation details of the composer followed by a section that concludes the present work and looks at future work.

2. Previous Work

Different avenues for automatic and computer aided generation of music has been explored by several researchers over the past few decades. The first work that surfaced in the field of music generation was for practical purposes [2], where the authors claim that the use of a generative system would be of great help to composers for extending present composition techniques and to produce diversely different species of music through artificial means. Other works in this field that need mention are mathematical language based systems [3, 4, 8], generative grammar based systems [5, 9] and genetic algorithm based approaches [6, 7]. A comprehensive survey can be found in [1], where the authors summarize the various works in automated compositional research.

Computational modelling of Hindustani Classical Music, its analysis and synthesis has been explored by H. V. Sahasrabuddhe [10] for the first time from the perspective of Indian music. The approach involved the use of a finite state automaton built from given examples of performances, which can later generate reasonably good compositions in a *Raga*. Sahasrabuddhe claims that automatic as well as computer-assisted composition is possible with the above model. However a complete generative system has not been described in his work.

3. Mathematical Model of the Generator

The mathematical model behind the generative system is a finite state machine (FSM) which models the entire set of notes spanning three octaves. The FSM is built for a particular *Raga* and an algorithm is run on it to generate a unique music sequence. The following subsections describe this mathematical model in detail.

3.1 Basic Terms and Definitions

Before we move on to describe the mathematic model based on which the generative system has been developed, familiarization with a few terms and definitions becomes essential. The formal definitions of these terms can be found in [12]. Here we reproduce some of them.

- *Raga:* The *Raga* is the basic unit of Indian Classical Music. It is defined to be a set of notes, and the adjacency relationship of the notes is governed by the *Raga's* grammar.
- *Arohana: Arohana* is the ascending sequence of notes that the *Raga* follows. Any ascending sequence in improvised portions of the *Raga* follows the pattern defined in *Arohana* strictly.
- *Avarohana:* Similarly *Avarohana* is the descending sequence of notes that the *Raga* follows.

The range of notes we consider in the present work span three octaves. Any note sequence is a string of alphabets belonging to a finite set M, where M is the union of three sets M_{tar}, M_{madhya}, M_{mandra}, each of them consisting of the 12 basic notes. The subscripts *tar, madhya* and *mandra* are indicative of the higher, medium and lower octaves respectively [12].

3.2 The Probabilistic Finite State Machine

A probabilistic finite state machine (FSM) each for the ascending and the descending movements for a particular *Raga* is constructed for generation purposes. The finite state automaton can be described as a bigram model of note sequences, which specifies the three most frequent notes along with the corresponding probabilities that can follow a given note in a composition, while moving in a particular direction. Formally, the automaton for *Arohana* can be described as a set of 36 nodes (for the finite set *M*) each representing a note. For a node labeled *n* there are three outgoing edges e_1, e_2, e_3 with probabilities p_1, p_2, p_3, where

$$p_1 + p_2 + p_3 = 1$$
$$p_1 > p_2 > p_3$$

The edges point to some other nodes labeled n_1, n_2 and n_3 (possibly null). This is to be interpreted as follows. While in *Arohana*, after a note *n*, the probability of generating the notes n_1, n_2 and n_3 are p_1, p_2 and p_3 respectively. No other note is allowed after *n*, while in *Arohana.* The automaton for *Avarohana* can be defined similarly.

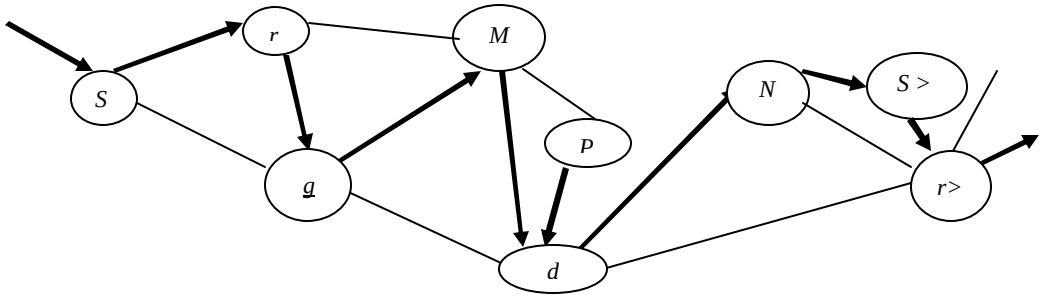

Figure 1. The part of an example transition diagram for the *Arohana* for *Raga Miyan Ki Todi*.
The range of notes is from *S* to *r*>. The bold edges display the edges that are more probable to be used.
None of the nodes here use the 3rd edge.

Figure 1 shows an example finite state machine for the *Arohana* of *Raga Miyan Ki Todi.*

3.3 Designing the FSMs for a *Raga*

The FSM for the ascending and the descending movements for a particular *Raga* is hand crafted by an expert in Hindustani Classical music. In the present design of the FSM, the probabilities p_1, p_2 and p_3 for the three different edges e_1, e_2 and e_3 for a particular node n are fixed for all *Ragas*. The probabilities used are $p_1 = 0.70$, p2 = 0.20 and p3 = 0.10. The following steps are used while constructing the FSMs for the *Arohana* and the *Avarohana* of a particular *Raga:*

1. For a particular *Raga,* each note *w* of all the octaves M_{mandra}, M_{madhya} and M_{tar} is taken up one by one for both the *Arohana* and *Avarohana.* If the note *w* is forbidden in the concerned *Raga,* then no edges emanate from the node corresponding to *w*. Otherwise, the note *x* which is most frequently used after *w* is the note transition to which is most probable. The probability p_1 is assigned to the edge e_1 that connects the node corresponding to *w* to the one corresponding to *x*.
2. Next, the note *y* is considered that is the next frequent note (less frequent than *x*) among the notes that occur after *w*. The probability p_2 is assigned to the edge e_2 that connects the node corresponding to *w* and the node corresponding to *y*. Such a note *y* may not exist at all, in which case this edge is not present.
3. If a note such as *y* exists, then we search for a third frequent note *z* and similarly the edge e_3 is constructed which is assigned a probability p_3. Such a note *z* may not exist at all.

After the FSMs for a particular *Raga* are constructed, a note sequence generation algorithm is used on them to get a composition. The following subsection describes the algorithm used in the present work.

3.4 Generation Algorithm

The basic algorithm for generating a sequence of notes takes as input the two FSMs for the *Arohana* and the *Avarohana* for a particular *Raga.* The number of notes that the sequence will have and the start note of the sequence act as inputs as well. Let the sequence of notes be denoted as:

$$S = \{s_0, s_1, s_2 \ldots\ldots\ldots s_{n-1}, s_n\},$$

where s_0 is the start note and serves as input. Let the FSM corresponding to the ascending movement be FSM_a and the one corresponding to the descending movement be FSM_d. The basic generation algorithm is as follows:

1. For the current note s_i, a random number $0 = rand_1 < 1$ is generated. If $0 = rand_1 < 0.5$, then we choose FSM_a as the FSM with which we will work. Otherwise we choose FSM_d.
2. The node $n_{current}$ corresponding to s_i is chosen in the selected FSM. Now, one more random number $0 = rand_2 < 1$ is generated.
3. If $0 = rand_2 < p_1$, then the next node n_{next} that we consider is n_1, i.e. the node that can be reached from $n_{current}$ via the edge e_1. If such an edge e_1 does not exist then this process is abandoned, the other FSM is selected and the process continues from Step 2 once more.
4. Otherwise if $p_1 = rand_2 < (1 - p_2)$, then the next node n_{next} that we consider is n_2, i.e. the node that can be reached from n via the edge e_2. If such an edge e_2 does not exist, then the edge e_1 is selected and n_{next} is assigned to n_1.
5. If both the conditions of Step 3 and 4 are not satisfied, then we choose the edge e_3, and the next node n_{next} in consideration is n_3. If such an edge e_3 does not exist, then the edge e_2 is selected and n_{next} is assigned to n_2.
6. s_{i+1} is now assigned to the note corresponding to n_{next}.
7. The whole process repeats from Step 2 if the total number of notes n to be generated is not reached.

At Step 2 of the algorithm, the selection of either FSMa or FSMd may produce two scenarios:

1. If the same FSM is selected in numerous consecutive steps, then notes generated will be in one particular direction for a long stretch of the sequence.
2. On the other hand if in nearby steps, alternate FSMs are selected, then there will be a lot of to and fro motion in the total sequence of notes.

For both of the above situations, it has been found that the quality of the note sequence generated is not of acceptable quality. For optimum quality, there has to be some measure that would determine which FSM to decide at Step 2 of the algorithm, that avoids both of the scenarios enumerated above. A solution is:

1. We initialize a variable *inertia* to some value which is between 0 and 0.5 (say 0.22).
2. A variable *flag* that toggles between two values 1 and -1 is also taken. The *flag* is assigned to -1 when at one step FSM_a is chosen and it is assigned to 1 when FSM_d is chosen.
3. At Step 2 of the basic generation algorithm, we check whether $0 = rand_1 < (0.5 + flag*inertia)$. If this condition is satisfied, we choose FSM_a. Otherwise, we check if $0 = rand_1 < (0.5 - flag*inertia)$. If this condition is satisfied, the other automaton FSM_d is selected.
4. If both of the above conditions are not satisfied, then rand1 is assigned a value of $0.5 + flag*0.5$ which forces the direction of note generation to reverse.

The use of the variable *inertia* prevents the occurrence of the two disadvantageous scenarios.

There are a few boundary conditions for the generative system as well. Whenever the current note reaches the end note of either limit, then the generator does not find any valid transition at the corresponding node and it is forced to change direction and come within the limits. The above algorithm thus forms a module that can generate a note sequence of a given length using the two hand-built FSMs.

4. Implementation of the Generative System

The system can generate compositions in ten *Ragas* viz. *Yaman, Yaman Kalyan, Miyan Ki Todi, Puriya Kalyan, Puriya Dhanasree, Bhairavi, Bhairav, Khamaj, Bilawal* and *Bihag*. The overall structure

of a composition is fixed as described below. Let there be n beats in a cycle. There is a *Sthayee* and an *Antara* consisting of two sequences of n notes. The composition repeats the first line of the *Sthayee* twice. Then the second line of the Sthayee and the Antara are repeated for a fixed number of times. Four n-beat *tanas* follow next with the first line of the *Sthayee* in between each of them. The composition ends in a *tehayi.*

A composition generator uses the basic note sequence generation module (Section 3.4) to get the n-beat sequences. The start note for each of the sequences is provided as input by generating these notes randomly. Preferably, the lines of the *Sthayee* start with a comparatively lower frequency note and the lines of the *Antara* start from a higher frequency note.

The *tanas* that are generated are of n-beat each but of twice the tempo of the actual composition. In the present implementation, only four types of beat sequences are used – *teentala*, *ektala*, *jhaptala* and *keharwa*. The generator has been implemented using Java 1.4, and a Java-based text to MIDI generator JFugue [12] is used for the realization of the compositions in audio. JFugue takes text notes as input to generate MIDI files of a composition. The different *talas* have also been realized using this software. The overall tempo of the song can also be varied as and when required. Jfugue provides a choice of a series of instruments and a composition can be generated using any one of these.

The implemented system has a friendly GUI that lets the user select the required *Raga*, the type of beat sequence he or she wants the composition to be generated in and the type of instrument from a limited set.

5. Experiments and Observations

In order to test the efficiency of the generator, an experiment was designed and conducted on a set of subjects as described below.

5.1 The Random Model

A random model for a particular *Raga* is defined to be any sequence of notes that belong to the *Raga*. But the consequent notes created by the random model may not conform to the grammar of the concerned *Raga*. These types of random sequences are included in the test suite as a placebo.

5.2 Experimental Setup

An experimental setup was constructed for judging the quality of the music generated by the computer as follows:

1. A set of five compositions in *Ragas Yaman, Khamaj and Todi* was picked up from Pandit Bhatkhande's book [13]. The *tanas* were composed by an expert and included inside the compositions. The structure of each composition was same as the one created by the generator. The instruments, beat sequences etc. were used from Jfugue [11] itself to negate any possibility of improvement in audio quality over the artificial compositions. A set of five compositions in the same three *Ragas* was composed using the random model which has the same overall structure as well. Five compositions using the artificial composer and five using the random model were generated to complete the test suite. The total of 15 compositions are mixed up and presented to a subject.
5. The subject was asked to judge each composition on a 10-point scale. He/she was also requested to identify the *Raga* of the composition and to give any comment on the composition, if any.
6. This testing was done on 10 different people among which five have had formal training in Indian Classical Music, and the others without any training but possessing interest in music.

5.3 Results

The results of the tests conducted on the subjects make evident the fact that the human composed pieces are of the best quality; precisely they have received a score of 8-9 on a scale of 10. The randomly generated sequences on the other hand have been marked lowest in the scale of 10. The sequences generated in *Raga Miyan Ki Todi* by the computer have received 7/8 from most subjects. But for the other two *Ragas* the scores are just a bit more the random sequences.

6. Conclusion and Future Work

In this paper we have discussed a system that can generate Hindustani Classical compositions for a limited set of *Ragas*. It can be concluded from the conducted tests that the artificially generated music pieces are superior in musical quality than random music samples, but yet quality of human created compositions is far above the ones created by the computer. It is interesting to note that for some *Ragas* like *Miyan Ki Todi*, the compositions are quite good because of the *Raga's* structure. For *Ragas* like *Khamaj* on the other hand the essence has not been captured at all. Since the mathematical model behind the system is uniform for all *Ragas* and is only a bigram model of the notes that the *Raga* allows, an expert in Hindustani Classical music can clearly explain the failure of the system in capturing the structure of complex *Ragas*. This leaves us a lot of scope of improvement and future work.

One aspect of the system may be improved by increasing the bigram nature of the finite state models to n-gram models where $n > 2$. This definitely would help us capture the essence of more complex *Ragas*. Often, the characteristics of a particular *Raga* makes is expressed because of the use of a few standard note phrases repeatedly. Incorporating these specific note sequences in the generative system can improve the quality of the artificial compositions as well. The present system generates compositions in MIDI format. In such a system it is not possible to include *meends* and *gamakas* that are central to Hindustani Classical pieces, and therefore the actual beauty of a human performance cannot be captured in its entirety.

References

[1] Pearce, M., Meredith, D. & Wiggins, G. (2002). Motivations and Methodologies for Automation of the Compositional Process. In *Musicae Scientiae*, 6(2), 119-147.

[2] Hiller, L. & Isaacson, L. (1959). *Experimental Music*. New York: McGraw–Hill.

[3] Xenakis, I. (1971). *Formalised Music*. Bloomington, Indiana: Indiana University Press.

[4] Ames, C. & Domino, M. (1992). Cybernetic Composer: an overview. In M. Balaban, K. Ebcioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* pp. 186–205. Cambridge, MA: MIT Press.

[5] Sundberg, J. & Lindblom, B. (1976). Generative Theories in Language and Music Descriptions. In *Cognition, 4*, 99–122.

[6] Wiggins, G. , Papadopoulos, G. , Phon-Amnuaisuk, S. & Tuson A. Evolutionary Methods for Musical Composition (1999) *International Journal of Computing Anticipatory Systems*, 1999.

[7] Phon Amnuaisuk, S. & Wiggins, G. (1999). The Four-Part Harmonisation Problem: A comparison between Genetic Algorithms and a Rule-based System. In *Proc of AISB 99*, Edinburgh, Scotland.

[8] Ames, C. (1987). Automated composition in retrospect: 1956–1986. *Leonardo*, *20*(2), 169 –185.

[9] Steedman, M. (1984). A generative grammar for jazz chord sequences. *Music Perception*, *2*(1), 52–77.

[10] Sahasrabuddhe, H. V. (1992). Analysis and Synthesis of Hindustani Classical Music. http://www.it.iitb.ac.in/~hvs/paper_1992.html

[11] www.jfugue.org

[12] Choudhury M. and Ray P.R. (2003) Measuring Similarities Across Musical Compositions: An Approach Based on the Raga Paradigm. Proceedings of International Workshop on Frontiers of Research in Speech and Music, Kanpur, 2003. (pp 25 - 34).

[13] Bhatkhande V.N. (1993) *Hindustani Sangeet Paddhati (Vol 1-4):*Hatras: Sangeet Karyalaya, 1993.