A stochastic musical composer based on adaptive algorithms

Bruno Abrantes Basseto & João José Neto
Escola Politécnica da Universidade de São Paulo

Abstract

This paper presents an algorithmic composition technique derived from Markov chain modeling, which may be used for efficiently describing musical knowledge, providing means of hierarchical structuring and context expressing by linguistic formalisms.

1 Stochastic Music Systems

What does it mean to compose music? This is an essential question, since its answer can lead to more interesting systems of automated composition. Composing is a *creative* attitude, in which something new is created, by a specific intention of a composer, who is motivated by aesthetic or expressive needs. Such a composing act can be thought as some organization of the sonorous material. Therefore, some sort of *musical semantics* is associated to an ordering imposed to musical sounds by the composer in a procedure that is just the opposite of chaos, which in sound terms may be identified to *noise*.

From the previous definition, one must find paradoxical using computers in music composition, since these devices cannot make any creative decisions, so, it is impossible for them to perform innovative creation and they have no means of expression and are not able to manifest any kind of aesthetic reactions to anything.

Although true creation is actually impossible to computers, since they are guided by machine programs that define *a priori* what will be their behavior, an interesting alternative would be found by relying on unpredictable results, that is, those related to stochastic processes. In such processes, a kind of creative phenomenon would be associated to the production of something new, in the sense of *unexpected*.

Following this idea, one may argue that the almost unpredictable results produced by such systems are more related to noise or chaos than to music itself. Although that is true for completely random systems, one may impose organization rules into this stochastic system in order to obtain some musical structure that a human listener would interpret as music.

How can music be properly represented in computer systems? A better question is: how would musical knowledge for composing be properly represented in such systems? Considering the very nature of computing systems, it seems we should treat music as some kind of language. That subject is usually controversial, but it seems clear that using syntactic structures may be very useful as a mental model for describing musical structure. Therefore, *grammars* are pointed to be valuable for representing musical knowledge in a computer, and they constitute a very practical user interface as well.

The present work is intended to be part of a musical meta-instrument, which has the objective to work in music therapy applications, not only in contemporary music performances. In such applications, music perception and response from the performer is the most important requirement. Therefore, our starting point relies on three principles: emulating musical creativity by means of random choice; representing musical knowledge and musical structure by means of syntactic structures, like grammars; and real-time performance, allowing the presence of human performers who should interfere in composing decisions.

2 Markov chains and Languages

Looking at computer music history, we will find the early experiences in algorithmic composition at Hiller-Isaacson's *Illiac Suite* [1]. In that work, stochastic paradigms were used for choosing compositional parameters, and musical structuring (or noise filtering) is represented by a transition state matrix or a Markov network.

A Markov chain is a non-deterministic state machine, where the probability of the system being at some particular state depends only on its previous state and on the probability of a transition between those states. This could be described by a state transition table. Such a device turns out to be useful for music composing through the attachment of compositional decisions to their transitions, such as producing musical outputs as a response to the occurrence of some transition.

Even though this kind of approach obtained very poor musical results, our interest in Markov chains rely on the fact that we can establish connections between them and the logical concept of language. In fact, the sequences of symbols produced by a series of Markov chain transitions can be proved to belong to some regular language. We may start from a linear grammar for that language, generating a corresponding finite state automaton and then transforming it into a non-deterministic first-order state machine by converting its deterministic symbol-consuming transitions into probabilistic symbol-producing ones by attaching them probabilities of occurrence. The second reason for the interest in Markov chains is the great simplicity shown by such devices, which makes its real-time implementation and interface very neat.

The obvious limitation of this class of algorithms is the fact that first order Markov chains can only produce results equivalent to those of *linear* grammars. Considering that musical structure happens to be hierarchically organized, or, in other words, that there are usually nested structures in most musical works, we have to choose a less restrictive type of language, namely the *context-free* languages. It is a consensus that even this kind class of languages is not enough for representing music: one may find several examples in which a compositional decision affects future symbol substituting; this fact is related to a specific *context* allowed by some rules in the grammar. These are *context-sensitive* ones, hardly implemented on computer systems.

A quick analysis of compositions made by a first-order Markov chain will lead one to the conclusion that the system was only barely able to structure musical material as the composition enlarges, since its organization will be strictly local. The first choice for overcoming this difficulty was using higher order Markov chains, that is, a state transition probabilistic system where the next decision depends not only on the present state, but also on a number of past states. This does not lead to a practical solution since the original simplicity vanishes suddenly as the order of the system rises and the musical result is still poor. The use of higher order Markov chains usually conduces to non-linear behaviors, like the presence of cycles, and its relationship with formal grammars becoming less immediate.

Our problem is then representing probabilistic systems which are able to model hierarchical musical structure (and not only a linear one), trying to find a particular means of introducing musical context-sensitive decisions, without discarding the simple idea of Markov networks.

3 Adaptive formalisms for music composition

The idea behind adaptive schemes is that, although the basic structure of the automata is maintained, it can be self modified in response to the occurrence of a transition. Therefore, an *adaptive* Markov network is defined by its initial set of states, initial state transition matrix *and* a set of transition *functions*, which are invoked whenever the associated transition is executed. Those functions may create or remove states or transitions in the

automaton, what means that they are able to dynamically change the graph topology, that is, the algorithm itself. Adaptive formalisms are efficiently employed in compiler construction and natural language processing. Their use in non-deterministic schemes is new, and it showed to be a good contribution to computer music composition.

It seems intuitive that such a mechanism is able to attach new processing steps to the original algorithm, associated to decisions that would be taken later *in the future*. That is, at a moment we may be concerned with choices that had a foundation in the past, what is equivalent of context sensitiveness. This new flexibility is introduced to make it possible to handle more sophisticated types of languages, leaving the original simplicity of the finite automata formalism untouched.

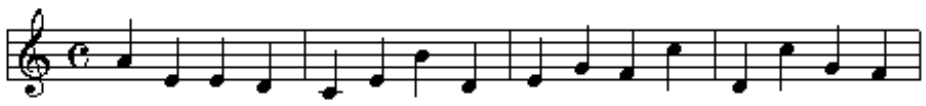
We can extend this concept to a *set* of adaptive state machines, where all state machines may be executed independently, but the set of transition functions can make transformations not only local to a particular machine, but in any of the machines belonging to the set. These relationships between any pair of adaptive Markov chains can be of two types: the first, both state machines interfere with each other, by adapting its own behavior to the other. The second kind of interaction is when one machine controls the other, by imposing restrictions or constraints it must meet; such inter-relation is called hierarchical.

Presented this way, one may use adaptive sets of probabilistic state machines to make a compositional framework with a much smaller number of states, employing transition functions to change contexts or interacting with another machine at the same level or vertically structured. One composer transcribing musical knowledge to this framework, can treat any isolated detail, modeled through a particular state machine, while having control over the entire structure, expressed by machine interactions. With this idea, we kept simple the model, by making it hierarchically structured and by adding to it the capacity of taking contexts into account.

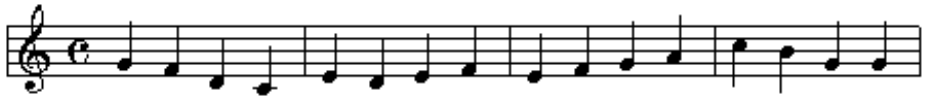
4 A PRACTICAL EXAMPLE

In this example, we report a small experience, in which we generate increasingly complex melodies by imposing successive restrictions to the chosen musical notes.

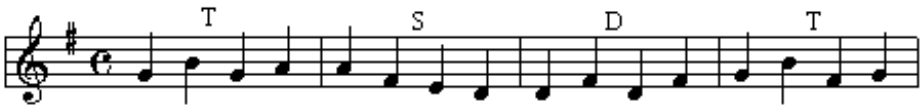
Let's start with a simple eight-state non-adaptive Markov chain to create a single melody line. For every transition arriving to each state we associate the generation of a musical note, in the scale of c major. The simpler machine has all transitions allowed between all states, with equal probabilities of occurrence (1/8). Hence, the probabilities of the system being in each of its states are the same, then the generated music will have the notes uniformly distributed among the possible notes. For instance, the system led to the following:



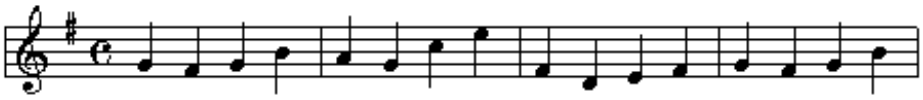
This result shows a melody line without any structuring parameter, and even considering the fact that it generates only notes from a known scale, its randomness gives us the sensation of noise. We would add some sort of redundancy in order to reduce such a noise effect. The first aspect one can note from the analysis of this melody is the fact that its melodic moves do not follow any principle of connectivity. We can improve that by making adaptive this probabilistic machine and by writing transition functions that increase the probabilities of transitions leading to non-distant states, which generate notes at intervals of seconds, thirds and fourths, in this order of preference. The octave jump is also allowed. With these restrictions, we can obtain the following:



We can observe that this particular result has much better melodic consistence, and its smaller degree of local randomness has reduced the noise sensation detected in the preceding case. The music generated still lacks any musical structure. The first level of structuring we have chosen to include refers to harmonic consistence. In order to achieve that, it would be convenient to use another Markov chain, which will be responsible for choosing the current harmonic function (tonic, dominant, subdominant), always following the correct evolution between two harmonic functions. That behavior is defined by a simple grammar. Each state of this machine will force preferential transitions to the earlier basic machine (which will correspond to preferential notes at the melody), increasing their probabilities at downbeats, like in:

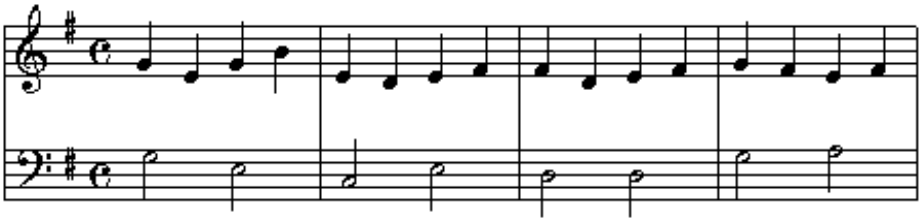


where the key was turned into g major. Capital letters over the measures represent the current state of the harmony Markov chain; the generated melodic line closely follows the chosen harmonic functions. One may easily identify the stronger musical character of this fragment. We may now aggregate to the music- generating framework some thematic- structuring feature, by adding yet another machine - a deterministic one - which will induce directions of melodic movement by increasing the probabilities of the transitions that correspond to the desired movement. Such a change may give us the following result:



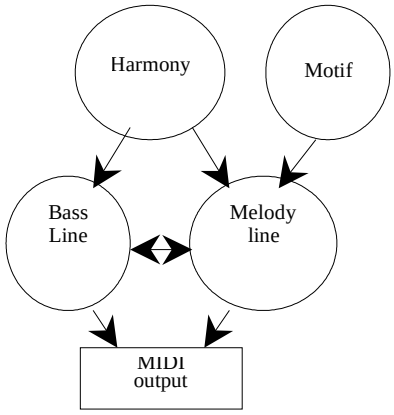
This composition is more interesting. A motive that last one measure is clearly identifiable, which is repeated through the entire period. We can easily have phrase- structuring if we extend this idea by aggregating sections with different thematic material, allowing nesting phrase structures by means of adding hierarchically influencing adaptive Markov chains. Again, the syntactic structure of the musical phrase to be created is defined by a grammar that will define the behavior of the top- most machine.

In the following example, we show the result obtained by adding one more adaptive state machine interacting with the previous melody- generator, generating another melody line (the bass line), which will imply counterpoint restrictions over the former.



The last example is particularly interesting, since it contains all the features presented behind, its acoustical result carry several intrinsic musical properties, like harmonic,

thematic and voice structuring. We may describe the machine relationships with the following diagram that also represents the actual implementation, where circles denote Markov chains and the arrows represent interactions between a pair of them:



5 Conclusions

There are two kinds of users of the implemented system. The first is the *composer*, who will introduce machine definitions corresponding to the musical knowledge actually modeled, by means of a simple language that is interpreted for constructing individual state transition matrices for the desired machines. The second is the *performer*, who interact to the system while it is composing and performing a new piece according to the behavior programmed by the composer. While the program is being executed, transitions fire attached adaptive functions, according to the specification in the source code, which modify transitions within machine structures, and change their associated probabilities. A run-time MIDI output is produced at each interaction and the system offers several primitives for controlling the MIDI channel. The performer is allowed to control several parameters like tempo, dynamics and articulation, which are kept by a timing controller. Several other higher-level compositional parameters can be set and altered in real-time, serving as external parameters for the adaptive machines. They may be used for forcing some musical decision by increasing its associated probability of occurrence, or simply for decision through conditional if-then-else clauses.

By now the performer interacts to the system through a set of windows-like controllers like scrollers and buttons, but this interface shall be converted into a group of sensors and cameras for interacting with the performer into a non-computational environment. The system works as a musical meta-instrument, where the performer make changes in high-level musical language structures, like harmony, sentence structure, form, row and so on, and not only on low-level note events.

The practical results we had obtained have demonstrated the power of the adaptive formalism in representing musical knowledge. The model kept simple our implementation, which allowed efficient real-time performances. We found that the most important feature displayed has been the ability of formally describing musical knowledge.

One can extend the shown examples to higher orders of musical material organization, like entire sections of musical pieces. For example, another Markov chain that controls the harmony generator behavior may be used for making the system to modulate to other keys inside a musical composition. The same way, the motif machine may be subordinated to a higher-level structure-generating machine that would make the musical piece to exhibit some given structure, e.g., a ternary ABA structure. The last machines may interact to each other to make, for example, the internal section ‘B’ to be in the key of the dominant, returning to the original key by the end of the composition. Note that all of these

enrichments are extremely simple to implement, without changing any part of the rest of the state machines.

The use of the adaptive formalism is also useful for implementing some musical language characteristics that are usually difficult to be done with other approaches. One particular and interesting example is the production of *thematic variations*. This is a situation where there is a context change; it can be easily implemented by allowing the theme-inducing machine to be affected by the current notes chosen by the melody generator. Therefore, such a memory of past variations works like an algorithm alteration over the original theme programmed into the system.

Although the examples shown were related to tonal music, the present system is perfectly applicable to many other forms of musical organizations, from serialism to microtonal music. The system would be used to control any low-level musical parameter and not only MIDI notes on and off, which may be very interesting for creating new forms of musical expression.

6 Bibliography

[1] HILLER, L., ISAACSON, **Experimental Music**, 1ed, New York, McGraw-Hill, 1959.

[2] NETO, J. J., **Contribuições à Metodologia de Construção de Compiladores**, Tese de Livre Docência, São Paulo, Escola Politécnica da Universidade de São Paulo, 1993.

[3] HOWARD, R. A. **Dynamic Probabilistic Systems**, v. 1 e 2, 1ed., New York, John Wiley & Sons, Inc., 1971.

[4] LERDAHL, F., JACKENDOFF, R. **A Generative Theory of Tonal Music**, 1ed., Cambridge, The MIT Press, 1983.

[5] POLANSKY, L. Live Interactive Computer Music in HMSL, 1984-1992. **Computer Music Journal** v.18 n.2 pp.59-77, 1994.

[6] ROADS, C. Grammars as Representations for Music. **Computer Music Journal** v.3 n.1 pp.48-55, 1979.

[7] SCHOENBERG, A. **Fundamentos da Composição Musical**, 2ed, São Paulo, Editora da Universidade de São Paulo, 1993.

[8] SMAILL, A., WIGGINS, G., HARRIS, M. Hierarquical Music Representation for Composition and Analysis. **Computer and the Humanities** n.27 pp.7-17, 1993.