Helsinki University of Technology
Telecommunications software and multimedia laboratory
Tik-111.080 Seminar on content creation
Spring 2000 Art@Science

Algorithmic Musical Composition

Hanna Järveläinen
40488w
April 7, 2000

Algorithmic musical composition

Hanna Järveläinen
HUT, Laboratory of Acoustics and Audio Signal Processing

Hanna.Jarvelainen@hut.fi

Abstract

This paper surveys some of the methods of algorithmic composition. Western classical music has been formalized in many ways throughout centuries, but the development of computers and the exponential growth of computing power have made it possible to generate music automatically. Common composing algorithms include state machines, rule-based grammars, stochastic processes, genetic algorithms, and so on. But who or what is the composer: the user or the maker of the algorithm?

1. INTRODUCTION

A certain degree of formalism is the basis of western music. Formal techniques for melody composition date back to the 11th century, when Guido d'Arezzo used a scheme that assigned a different pitch to each vowel sound in a religious text. In the 15th century, rhythmic patterns were used in a systematic way in *isorhythmic motets,* and for instance Guillaume Dufay was known to use the golden section to structure his compositions. He also derived the tempos for the different parts of a composition from the dimensions of a cathedral. Very strict rules developed for counterpoint in the renaissance and baroque era. For instance, a polyphonic composition could almost entirely consist of successions of systematic procedures such as inversions, augmentations and diminutions of a given theme.

The formal technique of generating music was adopted again in the 20th century. The *tone-row technique* was introduced by Arnold Schönberg at the beginning of the century and further developed into *serialism* by Anton Webern and his successors. In the serial technique music was factorized into parameters such as pitch, duration, and timbre that were controlled separately. A permutation was chosen from the possible values of each parameter and arranged into a row. The parameter values changed according to the row or its inversions or retrogrades [1].

In early 1950s Iannis Xenakis started to experiment with stochastic processes and used them for generating pieces by hand, without a computer. However, by that time digital computers already existed, and they were being used as a tool in the composition process. The first entirely computer-generated composition, The Illiac Suite for String Quartett, was published by Lejaren Hiller in 1956.

It became apparent that all kinds of musical processes, whether deterministic or stochastic, could be coded and implemented by computer. But as soon as music could be generated automatically, the question arose, whether the outcome of an algorithm could be called a composition. Generating music automatically within a given structural framework is not a sign of any particular creativity, furthermore, an arbitrary number of pieces could be popped out of the same algorithm. Perhaps the maker of the algorithm should rather be credited. Then again, it doesn't require significant musical or creative skills to apply a mathematical algorithm to music without considering the perceptual value of the work.

Many music generating algorithms require some input from the user of the algorithm. As long as the input and the algorithm are based on present and former musical tradition but are still original, the piece would fulfil the requirements of a composition. This approach expects some creativity from the user of the algorithm. There remains the question about a machine being creative itself. We will consider this after an overview of the common techniques of algorithmic composition. These include state machines, rule-based grammars, stochastic processes, and genetic algorithms, for instance.

2. THE PROCESS OF ALGORITHMIC COMPOSITION

Compositional algorithms can be roughly divided into deterministic and stochastic. Some algorithms behave in such a mechanistic way that is is easy to tell by listening what kind of algorithm it is. Usually this is not possible. Furthermore, several algorithms can be mixed in one piece of music, or the algorithms can be used in various dimensions of music, such as pitch, timbre, or overall structure. Composition programs can handle more details in shorter time and execute more formal rules than a human composer, who can switch the attention from details to overall strategy.

A totally automated algorithm requires little creativity from the composer. The task is to supply some seed data and execute the program. The compositional style and structure of the piece are determined by the algorithm. To interact with the procedure, the composer can modify the program. This way the composer would have the overall responsibility of the work by controlling the algorithm. Another option is to revise the output of the algorithm without changing the algorithm itself. This way the music becomes formally inconsistent, which is considered by some composers to reduce the role of algorithms and destroy the original idea of form. It is claimed that consistent rules guarantee the originality of the piece, but who can guarantee that the audience regards a piece that follows a set of artificial tules more originals than one that deviates from them.

By interacting with the algorithm, various layers of the compositional structure can be accessed. The smallest scale would include individual notes or parameter. Larger scale editing could affect entire phrases, procedures or the compositional strategy. In early computer systems, interaction with the program was only possible in batch mode, i.e. the whole program had to executed before changing the code. Today online editing is possible, and it can be done on stage while the work is being performed [2].

3. STOCHASTIC MUSIC

A stochastic process depends on the laws of probability. Decisions are made according to random numbers, and it is impossible to predict the exact outcome of the process in any point in the future. Stochastic processes are easy to access, and they have been broadly used by composers either as a basis of an entire composition or as an auxiliary decision-making routine.

In most cases, a probability lookup table is used for note selection [2]. It is a table where the chance of occurrence of every possible outcome is stored. The table follows a certain statistical distribution. For instance, if there are N different outcomes (notes to be selected), the probability of each outcome would be 1/N according to a uniform distribution. However, it is usually desirable that some notes appear more often than others. Then another kind of distribution could be used. *Linear distributions* give a line of fixed, increasing or decreasing probability between two points. The uniform distribution is a special case of the linear distribution. *Exponential distributions* give an exponential probability curve. The most natural distributions are bell-shaped, such as the normal distribution which has its minima at the end points and maximum (the expectation value) in the middle point [2].

The first to use probability distributions in composition was Iannis Xenakis (born 1922). He protested against the ideals of serial music, the strict control of individual sounds and the way of quantizing all musical parameters. He considered the process of continuous change as the center of musical information. In natural events he saw plenty of such processes, such as “the collision of rain with hard surfaces, or the song of cicadas in a summer field” [3]. These hearing percepts are created by innumerable individual events. As a whole they follow stochastic laws. Xenakis used the same kind of laws in generating large masses of sounds that experienced a continuous change into something else, such as from a pizzicato to a bowed voice.

In his work Pithoprakta (1955) Xenakis used stochastic processes in many ways. Occasionally the duration of the sounds deviates from a given mean according to a probability distribution. He derived the behaviour of a group of string glissandos from the kinetic gas theory, which relates the speed of the gas molecules in a closed space to the thermodynamic conditions. The speed of each molecule follows a bell-shaped probability curve. In the music, the speed of a molecule was described by the slope of a string glissando [4].

In this example, Xenakis did not derive the whole form of a musical piece from stochastic processes but used them as a way of generating changing sound fields. The end points of the processes were determined by himself. However, in some other works he has also experimented with stochastic derivation of musical form. Xenakis processed his stochastic ideas at first by hand but started soon to experiment with computers and wrote programs for generating stochastic music.

3.1. Markov chains

Markov chains are discrete systems, in which the present outcome depends on a number of previous outcomes. In other words, the present outcome is not independent, but the process has memory for past events that affect the future. The weighted probability distribution is in fact a zeroth order Markov chain. A first order Markov chain takes the previous outcome into consideration, a second order chain two previous outcomes, and so on. Second and higher order Markov chain outputs have been found musically interesting, but lower order chains are practically never used.

An Nth-order Markov chain can be represented by a transition matrix, which corresponds to an N+1-dimensional probability table [5]. For instance, a probability table for a first order Markov process would be formed by finding the probabilities by which each note appears after any of the other notes. This could be done by examining an existing piece of music or just by setting the probabilities according to own choice. The probability matrix could also be presented by a state diagram (fig. 1), where the notes would be presented by the states and the probabilities by the transitions between states.

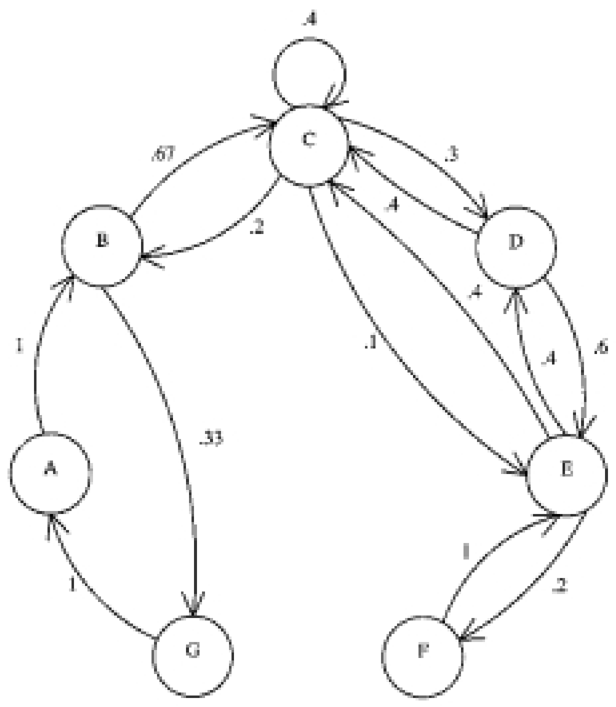

Figure 1: A state diagram for the Markov presentation of “Yankee Doodle”. The circles represent states (=occurence of particular pitches), and the arcs represent possible transitions with probabilities indicated by the numbers (Moore 1998).

One of the problems with Markov chains is that unless the probabilities in the state diagram are generated randomly, they have to be derived from existing music. By such method the process generates an output of the same musical style as the input. That kind of music has hardly any compositional value. Another restriction of the method is that it operates on discrete pitch values. Maybe the notes could be replaced by the changing speed of the pitch in the probability table, but even if the continuity of pitch would be gained, the slope parameter would still be quantized.

Markov chains can mainly be used for generating a melody. One possibility might be to extend the probability analysis to durations, harmonic relations, and the overall structure of the piece. But then we would be even more dependent on existing music for input, or a whole new musical style should be generated to get an interesting output. Furthermore, there would positively exist some interconnections between the different characteristics, and another model would be needed to control those. To account for all possible variations would also require a large probability matrix.

4. CELLULAR AUTOMATA

An automaton (a state machine) is a procedure whose output depends on its internal state and its input. In musical applications an automaton models the behavior of the musical parameters (pitch, timbre, amplitude, and so on). The task can be divided between several automata so that each controls one parameter. An automaton is defined through a number of states and the transitions between states that depend on the input. The output of the automaton depends on both current state and the input. This can be presented through a state diagram in similar manner as for Markov processes. Certain inputs and outputs are associated with each of the arcs representing the stage transitions. If several automata are linked, they constitute a system that transmits the previous output of one or more automata to the input of the others. The effect of the linkages can be either direct or inverse, and each linked automaton may have a different weight factor. After the initial input the system runs autonomously. Its evolution is determined by internal analysis that changes the sources and the weights.

Cellular automata (CA) are linked systems, where the behavior of each automaton (cell) is the same. The interconnection of the cells is a symmetric structure that does not change with time. All cells execute a single rule synchronously. An evolution rule is applied to the basic elements (cells) who develop from the starting configuration with time according to the rules. The evolution rules are only considered between the local neighbours of a cell. Global trends are not coded beforehand. Although the function of each individual cell is simple, complex global behavior is created by their interaction. If the system doesn't die out, it usually ens up either executing periodic behavior, chaotic or irregular cyclic patterns. A well-known CA application is the *game of life* that models the evolution of simple organisms [2].

The CAMUS 3D system is an example of a musical application of cellular automata [6]. As the composition progresses, the themes of the music experience transformations such as repetition, inversion, transposition, and augmentation. Thus the temporal development is due to the evolution rules and is not specified in advance. It uses two cellular automata structures, the Game of Life and the Demon Cyclic Space. The cells of the former can either be dead or alive, and the latter is based on higher states dominating lower ones. At each time step, the coordinates of the live cells of the game of life structure are analyzed and used to determine a four-note chord. The instrumentation is determined by the corresponding cells of the cyclic space structure.

The progression is seen in figure 2. The cell (5,5,2) of the game of life is active, and so a chord is chosen whose pitches are 0, 5, $5+5$, and $5+5+2$ semitones above the chosen fundamental pitch. The corresponding cell of the cyclic structure is in state four, which means that the chord would be played by the instrument number four. The fundamental pitch of the chord can be chosen either manually or according to some other algorithms, for instance by a stochastic selection routine. To avoid a piece that consists entirely of block chords, also the starting times and the order of the notes are controlled stochastically.

5. FLOW CONTROL, GRAMMARS, ETC.

The traditional music notation itsef includes a hierarchical structure (such as notes, phrases, and movements) and musical flow control procedures such as repetitions and jumps (for instance *da capo al fine*). Musical control structures and formal grammars are based on modeling these characteristics. One of the main applications of control flow algorithms as well as formal grammars are interactive performance systems, where the structure and order of the musical events are decided by musical external events. In addition to these, many other compositional algorithms are based on applying rules. These include for instance generate-and-test methods, constraints and expert systems [2].

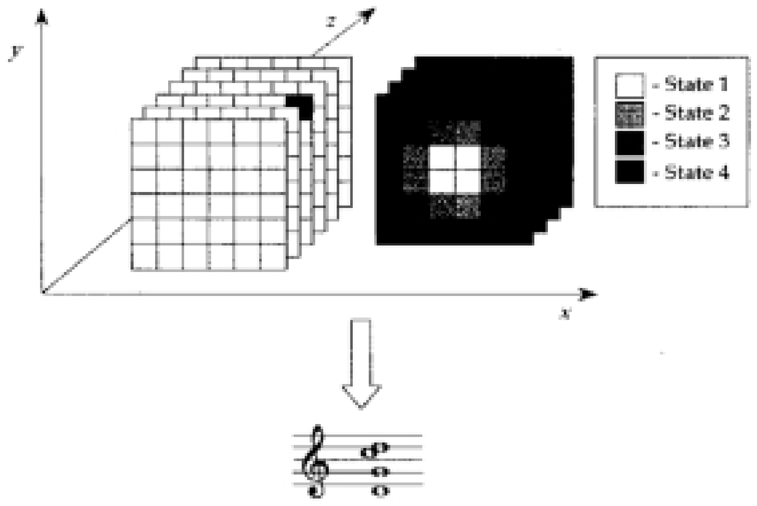

Figure 2: A typical time step of the CAMUS 3D system. A musical chord is chosen based on the game of life automaton, and the instrument is chosen by the state of the corresponding cell of the cyclic space automaton (McAlpine, Miranda, and Hoggar 1999).

5.1. Models of musical control structure

Music can be seen as sequences of events that are controlled by directives such as jumps, procedure calls, and loops. Repetitions and jumps are the most obvious flow control statements, but for instance the variations of a theme could correspond to a procedure call to decorate a melody. The biggest problem is that several events happen at the same time and interact with one another.

Petri nets are suitable for controlling interactive and concurrent processes [7]. They consist of *places* that hold tokens as a description of the state of the particular place, and *transitions*, which are active components unlike the ones in automata. The places are connected to transitions by input arcs, and the transitions are connected to other places by output arcs. A transition can *fire*, if it is enabled by a sufficient number of tokens at its input places. Tokens are moved to other places only through transitions. The firing of transitions may also be controlled by different kinds of objects such as counters or delays. The most basic net modules are for instance sequences and forkings. Petri nets are an efficient process control structure. For instance, they can model the different voices of a polyphonic piece.

5.2. Grammars

In most music a degree of hierarchy exists naturally. We have individual notes that connect to each other to form phrases, sections, movements, and whole pieces. The problem with many compositional algorithms is the lack of macrostructure. Music that consists of only one level of hierarchy (for instance successive short melodies) is hardly of any interest. A formal grammar is a way to present hierarchical structures. A grammar can be applied to music as well as natural languages, for instance.

The composer defines a musical grammar. A grammar is a set of rules that expands high-level symbols (such as “second movement”) into a detailed lowest-level description.

The composer can now choose a suitable macrostructure, which is then further elaborated by the computer according to the grammar. Of course, the grammar defines a set of compositions, not just one. By varying the macrostructure, the composer can try out an arbitrary number of candidates for the final composition [2].

A grammar consists of variable-level *tokens*. Non-terminal tokens present macrostructure, and terminal tokens are the lowest-level units, such as notes. The hierarchy of the different tokens is controlled by a set of *rewrite rules* that specify which lower-level tokens can replace a higher-level one. Some grammars generate only a single token at a time, while others can produce a mixture of terminal and non-terminal tokens. Some rules are context-sensitive. This means that the rewrite rules depend on the neighbours of the higher-level token.

5.3. Other rule dependent systems

The *generate-and-test* (GAT) method is one of the earliest ways of algorithmic composition [2]. It is based on generating at first a random parameter value and subjecting it to a series of tests. If the value passes all the tests, it will be chosen as output, otherwise a new random value will go through the tests and so on. The result of the test may be absolute acceptance or rejection of the value, or merely guidelines that accept many values within a region. Some systems provide a “degree of success” without rejecting any value. A way to make the GAT system more efficient is for instance to settle for a close enough value and adjust it according to the rules. If the task is for instance to generate triads, one with two suitable pitches could be chosen and the third one could be adjusted. GAT methods have been used by Lejaren Hiller among others.

Expert systems use knowledge and inference to choose the next sequence. Besides facts and rules, the knowledge base of an expert system consists of heuristics. Facts are direct statements about some parameter values, while rules are conditionals. Heuristics call for inference on present knowledge of parameter values. For instance, the next most probable pitch could be infered on five previous ones, etc. Expert systems are most suitable for generating music whose style can be codified by facts, rules, and heuristics, such as the musical style of Bach’s keyboard pieces. D. Cope developed a system that imitates the personal musical style of several composers of the tonal era. The biggest constraint of expert systems is that new musical styles are not that well-defined or have not developed far enough to be codified extensively. In that case the system should be able to learn from material presented to it and complete the knowledge base by itself.

Neural networks have been applied for melodic interpolation and extrapolation based on a learning set of desirable melodies without constructing a knowledge base at all. But the problem is the same: A large enough set of compositions have to be available before the composing algorithm can work. That kind of system might rather be useful in analysis purposes instead of composition.

6. FRACTALS

As Benoit Mandelbrot explained, a certain kind of spectral density indicates fractional behaviour of a sound. If the playback speed of sound is doubled, the quality of the sound is usually changed. However, when we listen to white noise at double speed, it still sounds like white noise. White noise has a completely flat spectral density, it is random and uncorrelated. Another kind of noise process is Brownian noise, which is random but highly correlated and has a spectral density of $1/f^2$. Fractional noises have a spectral density of $1/f^g$, where $0 \leq g \leq 2$ [2].

Richard F. Voss and John Clarke studied the spectral density of the audio power of musical signals in the mid-1970's. They found that the audio power of many different kinds of musical sounds showed a fractional $1/f$ behavior. This kind of process lies between the totally uncorrelated white noise and the highly correlated Brownian noise (see figure 3). An important property of pure $1/f$ noise is that it correlates logarithmically with its past values. Thus the last ten values have as much influence as the last hundred and the last thousand.

Another feature of $1/f$ noise is *self-similarity*, which means that the smaller scale details match the ones higher in hierarchy, but on a different scale. The self-similarity of musical structures has fascinated composers long before fractal mathematics was known. J.S. Bach sometimes used in his fugues the same pitch contour in the slowly-varying cantus firmus as well as the detailes of the surface structure. Even earlier, it was typical to derive the key for each movement of a work from a short melodic motive. Mandelbrot suggests that music exhibits fractal behaviour for the reason that much of it is hierarchic and even self-similar of structure. Pieces are broken down into movements, sections, phrases, and notes.

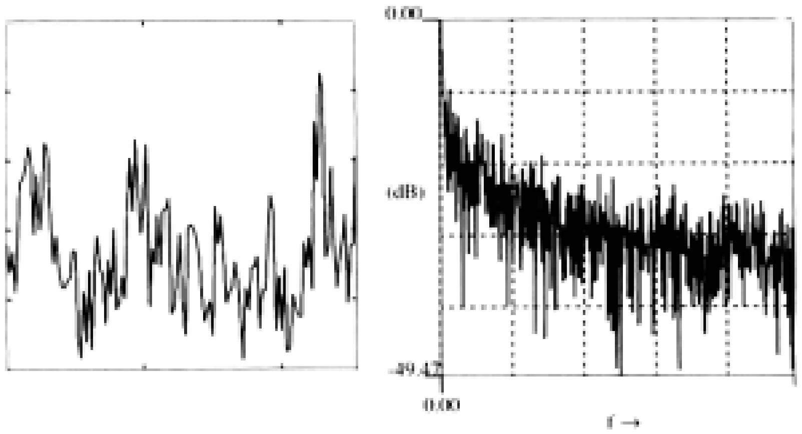

Figure 3: Waveform and power spectrum of 1/f noise. The waveform on the left was generated by summing together a number of random noise generators in a certain way. The 1/f power spectrum on the right drops off 3 dB per octave (Moore 1998).

Voss and Clarke applied the $1/f$ principle on music composition. They used a $1/f$ noise generator as a pitch selection unit. The random numbers from the noise process were rounded and scaled to produce pitch values in a range of two octaves. They generated music in the same manner using also white noise and Brownian noise and compared the melodies produced by using each noise process. The melodies generated by using white noise for pitch selection were too random, while the Brownian noise produced dull and too correlated melodies. The ones produced by the pure $1/f$ noise sounded musically most natural [5].

Most methods of relating fractal mathematics to musical composition are based on selecting pitches according to fractal objects. In addition to $1/f$ noise processes, many other fractal forms are used. For instance, Robert Greenhouse's The Well-tempered Fractal system [8] is based on selecting at first a fractal and a musical scale onto which the output of the fractal function is mapped. The control of musical events through fractals has been expanded from pitch selection to onset times, dynamics, number of MIDI channel, or the number of parallel notes by for instance Gary Lee Nelson [9], who used parallel fractal algorithms in his works.

One of the Finnish pioneers in fractal-related music is Ville Pulkki, who took another kind of viewpoint. His work “Kuusi” is not based on pitch selection by fractal algorithms, but making the self-similar structure of fractals clearly audible. He used the slightly modified Koch island fractal (in visual form the snowflake-like image) to define the overall structure of the work, while the musical intervals and the root pitch were selected by the composer. This lead to meaningful harmonic passages due to slowly varying base notes. The endless resolution of details in fractals was also well captured by the surface structure [10].

7. GENETIC ALGORITHMS

Genetic algorithms (GA) imitate the natural selection by producing new generations of candidates to solve a specific problem. A population is at first initiated randomly in a manner that depends on the problem to be solved. It is then evaulated for fitness. A number of individuals are selected by a procedure that most probably picks the best fitting candidates. A mating pool is formed of the chosen individuals. The *crossover* operation is applied in the mating pool by pairwise swapping part of the the bits between two individuals. A *mutation* may sometimes occur. A single bit is changed in the population. Tha algorithm stops when a chosen termination condition is fulfilled.

To apply genetic algorithms to musical composition, a certain environment must be established for the process [11]. At first, the number of possible combinations of the musical objects represented by the population must be limited to the meaningful ones. Otherwise, the “search space” for different combinations of notes in time, rhythm, and harmony would be unmanageable. Another important factor is rule presentation. A set of rules is used to define the possible ways of evolution of the composition with successive iterations of the algorithm. The third critical area is fitness evaluation as a measure for how well the rules are obeyed. The fitness function decides whether an individual survives to the next generation or not.

Genetic compositional algorithms can be classified by their different fitness-evaluation methods. Many of them were previously introduced as independent compositional algorithms. Now they are used as a part of a GA system. The evaluation methods can be deterministic, formalistic, user-determined, or neural.

Deterministic evaluation methods use a mathematical function to give the fitness of the individual musical event. The function could be for instance the number of common elements. Deterministic methods could be used to melodic development by applying specific rules to a phrase of music to alter it in some way. Formalistic rules make use of stylistic features of existing music that are codified into a set of rules. These kinds of methods have been used for instance to harmonize mlodies according to chosen musical styles. The fitness measure indicates how well genetically reproduced chords follow the traditional rules of the style. Another possibility would be to see the rules as constraints upon the search space.

User-determined evaluation methods call for subjective input from the user of the algorithm. The user is required to supply the fitness measures for individuals or to stipulate the style of the output. In another application by Biles, the evaluation was based on the average opinion of the audience. Biles also created a jazz improvization system called GenJam. The performance of the system is supervised by the user who acts as mentor to the algorithm, encourageing when the improvization is good. The neural fitness functions are in a way related to the user-defined ones. But now the neural network is to recognize the good individuals according to its training data. In addition to individually functioning evaluation methods, neural networks are used to make the human decision-making more efficient. For instance, a neural network might pre-select the candidates that are presented to a human mentor. Other applications of neural fitness functions include generating short rhythmic and melodic patterns.

7.1. But what about "genetic music"?

Also another kind of biologically inspired compositional method is said to produce "genetic music". This is in no way related to genetic algorithms, but to the structure of human DNA. Similarities have been found between genetic and musical structures. Both consist of a linear sequence of elements that form complex combinations. The DNA signals seem to exhibit the same kind of $1/f$ power spectra [12]. Genetic music, or DNA music, is created by exploring different characteristics of the four subunits of DNA (thymine, cytosine, adenine and guanine). For instance, Susan Alexjander used the light absorption spectra of the DNA base molecules to derive the pitches. John Dunn uses amino acid characteristics, such as molecular weights and volumes, to select the values of pitch and other musical parameters.

8. DISCUSSION

We saw some of the techniques of algorithmic composition. Of course, many others exist. Stochastic methods are based on selecting musical parameters and structures by random numbers. The stochastic distribution can be correlated or uncorrelated. Uncorrelated distributions can be seen as zeroth order Markov chains. Higher order Markov processes correlate the current value of the output with its past values. Automata, musical flow control structures and grammars generate musical structure by applying different kinds of rules. Rule-based algorithmic composition is encouraged by the hierarchical structure of most music. A widely used tool for algorithmic composition is also the *sonification* of other natural processes that are in some ways similar to music. For instance, fractional noise or even our DNA system exhibit same kind of behaviour as music.

The relation between human and algorithmic composition is not well-defined. Some composers use algorithmic methods as building blocks while they control the output of the process and remain responsible for the final result. Others feel that a logically generated structure is all that matters and try to avoid interaction with the composing algorithm that might reduce the degree of formality. Which ever the standpoint, the algorithms always require a certain amount of input from the user. No algorithm can generate meaningful music from scratch. Knowledge of the desired musical style, or a set of artificial rules, or a training set of existing music is at least needed.

Evaluation of algorithmic compositions is problematic. Usually algorithms are evaluated by their computational efficiency, but it has no meaning in compositional applications unless a system needs to run in real time. The degree of human interaction could be one measure, but it can be either desirable or undesirable. The ability of the algorithm to obey the given rules and generate music of the specified style could also be taken as a basis for evaluation. But then we would not be judging the compositional value of the work but the ability of the algorithm to imitate. In my opinion, the only way to judge is by listening. A completely formalized structure adds no value to the work, if it is not perceived and appreciated as such. It may be impossible to tell the degree of formality by listening.

A composition is a creative piece of work. But who is responsible for the creativity, the algorithm itself or the maker of it? We can argue that human composers also base their creativity on existing musical knowledge and don't have to start from scratch. But human composers never have a complete set of rules, because the present musical style may still be developing or the composer's individual style cannot be codified. Human composers develop the style as they work. This makes their work unique and creative.

I see the value of algorithmic composition elsewhere. While the music itself may not be that brilliant, real-time operation makes it possible to apply input to the system online. This way music can interact with other art forms such as dance or visual arts, and it may become the interpreter of external musical sources of creativity.

9. BIBLIOGRAPHY

[1] Donald Jay Grout. *History of Western Music*. W.W. Norton & Company, 5 edition, 1996.

[2] Curtis Roads. *The Computer Music Tutorial*, chapter 18-19. MIT Press, 1996.

[3] Iannis Xenakis. *Formalized Music*. Indiana University Press, 1971.

[4] Nouritza Matossian. *Xenakis*. Kahn & Averill, London, 1986.

[5] F. Richard Moore. *Elements of Computer Music*, chapter 5. Prentice Hall, 1998.

[6] Kenneth McAlpine, Eduardo Miranda, and Stuart Hoggar. Making music with algorithms: A case-study system. *Computer Music Journal*, 23(2):19–30, 1999.

[7] D. L. Kimbler. Petri nets. URL: `http://taylor.ces.clemson.edu/ie340/ files/ 340-16.htm`, 1997.

[8] Robert Greenhouse. URL: `http://www-ks.rus.uni-stuttgart.de/people/schulz/ fmusic/wtf/` , 1995.

[9] Gary Lee Nelson. Real time transformation of musical material with fractal algorithms. URL: `http://www.timara.oberlin.edu/people/%7Egnelson/gnelson.htm`, 1993.

[10] Ville Pulkki. Musical presentation of fractals. Unpublished manuscript. Submitted to the ICMC2000 Conference, August 27th - September 1st 2000, Berlin, Germany, 2000.

[11] Anthony R. Burton and Tanya Vladimirova. Generation of musical sequencies with genetic techniques. *Computer Music Journal*, 23(4):59–73, 1999.

[12] Wentian Li. Protein and dna music. URL: `http://linkage.rockefeller.edu/wli/dna_corr/music.html`, 1998.