PSEMA: An algorithm for pattern stimulated evolution of music

A.N. Mavrogianni*, D.S. Vlachos† and G. Harvalias*

**Athens School of Fine Arts, 42 Patision St.,106 82 Athens, Greece*
†*University of Peloponnese, 22100 Tripoli, Greece*

Abstract. An algorithm for pattern stimulating evolution of music is presented in this work (PSEMA). The system combines a pattern with a genetic algorithm for automatic music composition in order to create a musical phrase uniquely characterizing the pattern. As an example a musical portrait is presented. The initialization of the musical phrases is done with a Markov Chain process. The evolution is dominated by an arbitrary correspondence between the pattern (feature extraction of the pattern may be used in this step) and the esthetic result of the musical phrase.

Keywords: genetic algorithm
PACS: 45.10.Db

INTRODUCTION

Algorithmic composition, sometimes also referred to as "automated composition," basically refers to "the process of using some formal process to make music with minimal human intervention" [1]. Such "formal processes," as we will see, have been familiar to music since ancient times. The title itself, however, is relatively new; the term "algorithm" having been adopted from the fields of computer science and information science around the halfway mark of the 20th century [2]. Computers have given composers new opportunities to automate the compositional process. Furthermore, several different methods of doing so have developed in the last forty years or so, although even Mozart too, used automated composition techniques in his Musikalisches Wurfelspiel ("Dice Music"), a musical game which "involved assembling a number of small musical fragments, and combining them by chance, piecing together a new piece from randomly chosen parts" [1].

The earliest instance of computer generated composition is that of Lejaren Hiller and Leonard Isaacson at the University of Illinois in 1955-56. Using the Illiac high-speed digital computer, they succeeded in programming basic material and stylistic parameters which resulted in the Illiac Suite (1957). The score of the piece was composed by the computer and then transposed into traditional musical notation for performance by a string quartet. What Hiller and Isaacson had done in the Illiac Suite was to (a.) generate certain "raw materials" with the computer, (b.) modify these musical materials according to various functions, and then (c.) select the best results from these modifications according to various rules (Alpern, 1995). This "generator/modifier/selector" paradigm was also later applied to MUSICOMP, one of the first computer systems for automated composition, written in the late 1950s and early 1960s by Hiller and Robert Baker, which realized Computer Cantata: "Since [MUSICOMP] was written as a library of subroutines, it made the process of writing composition programs much easier, as the programmer/composer could use the routines within a larger program that suited his or her own style" [1]. This idea of building small, well-defined compositional functionsŮi.e. "subroutines", and assembling them together would prove efficient and allow the system a degree of flexibility and generality [1], which has made this approach a popular one in many algorithmic composition systems even into the present day.

Another pioneering use of the computer in algorithmic composition is that of Iannis Xenakis , who created a program that would produce data for his "stochastic" compositions, which he had written about in great detail in his book Formalized Music [4]. Xenakis used the computer's high-speed computations to calculate various probability theories to aid in compositions like Atrees (1962) and Morsima-Amorsima (1962). The program would "deduce" a score from a "list of note densities and probabilistic weights supplied by the programmer, leaving specific decisions to a random number generator" [1].

With Xenakis, it should be noted, however, "the computer has not actually produced the resultant sound; it has only aided the composer by virtue of its high-speed computations" [3]: in essence, what the computer was outputting was not the composition itself but material with which Xenakis could compose. In contrast, the work of Hiller and

Isaacson attempted to simulate the compositional process itself entirely, completely delegating creative decisions to the computer.

Already in these first two examples (Xenakis and Hiller) we find two different methodologies that exist in computer-generated algorithmic composition: (1.) "stochastic" vs. (2.) "rule-based" systems. As we will see, there is also a third category, (3.) which we can label AI, or artificial intelligence systems.

ARTIFICIAL INTELLIGENT SYSTEMS FOR MUSIC COMPOSITION

One unique approach for algorithmic composition using the computer is that of artificial intelligence (AI) systems. These systems are like rule-based systems in that they are programs, or systems of programs, based on some pre-defined grammar; however, AI systems have the further capacity of defining their own grammar, or, in essence, a capacity to "learn".

An interesting branch of AI techniques is that of "genetic programming," a very recent technique in the field of computer science for "automatic programming" of computers [1]. Rather than basing its grammar on scores input to the computer, genetic programming generates its own musical materials as well as form its own grammar. The composer must also program a "critic" function, therefore, which then listens to the numerous automatically produced outputs at various stages of the processing to decide which are "fit" or suitable for final output (the composer having final say, then, as to which of these to discard and which to save). Below is a more in-depth description of the different processes involved in genetic programming methods:

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 evaluated 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. The 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 [5]. 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 melodies 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.

PATTERN STIMULATED EVOLUTION

The musical phrases that consist the initial population are constructed using a Marcov chain process. For each member of the population:

1. generate a pitch randomly
2. generate duration of pitch randomly (following Poisson statistics)
3. generate next pitch following a Gaussian distribution with mean value the last pitch
4. got to step 2

Fitting of each member of the population is evaluated using a predefined pattern. The pattern is a twelve dimension vector, with values ranging from 0 to 1. Each value corresponds to each of one of the major scales and the according minor one. Using miditoolbox by [6], the l;evel of dominance of each scale in each member is calculated and the fitting function calculates the inner product between the pattern vector and the level of dominance of each scale in the member.

CONCLUSIONS

PSEMA is a system for pattern stimulated evolution of music. The pattern can be arbitrary making the system capable of performing audio effects on installations. As an example of the capabilities of the system, a musical portrait is presented.

REFERENCES

1. A. Alpern, Techniques for algorithmic composition of music. http://alum.hampshire.edu/ adaF92/algocomp/algocomp95.html. Hampshire College (1995).
2. H.K. Burns, Algorithmic composition, a definition. http://music.dartmouth.edu/ wowem/hardware/algorithmdefinition.html. Florida International University (1997).
3. D. Cope, New Directions in Music. 4th ed. W. C. Brown: Dubuqe, Iowa. 259 (1984).
4. Xenakis, Iannis. Formalized Music: Thought and Mathematics in Composition. Indiana University Press. 387.
5. A.R. Burton and T. Vladimirova, Generation of musical sequencies with genetic techniques, Computer Music Journal, **23**(4),59Ű73 (1999).
6. T.Eerola and P. Toiviainen, MIDI Toolbox: MATLAB Tools for Music Research, University of Jyvaskyla, Finland (http://www.jyu.fi/musica/miditoolbox/) (2004).