A User-Centric Algorithmic Composition System

Aurélien Antoine and Eduardo Miranda

Interdisciplinary Centre for Computer Music Research
University of Plymouth,
Plymouth,
PL4 8AA, UK
aurelien.antoine@postgrad.plymouth.ac.uk

Abstract. Inspiration and compositional processes are eclectic and different for each composer. For example, some draw inspiration from literature or nature to inform their musical creativity while others use algorithms and computer programs. This paper introduces a computer-aided algorithmic composition system implemented in *OpenMusic*. We start the paper with brief background information about algorithmic composition and automatic composition systems, followed by the description of our system. Then, we present some examples to illustrate the abilities of the system. The paper concludes with final remarks.

Keywords. Algorithmic Composition, Computer-Aided Composition, Computer Music, Generative Composition, Markov chain.

1 Introduction

Finding inspiration to compose a piece of music is different for each composer. Some observe nature, use literature or even interpret scientific phenomena to expend their musical creativity. Others prefer to use rules in their compositional processes. This practice resulted in the creation of the algorithmic composition field.

The term algorithm originally referred to performing arithmetic [15]. Nowadays, an algorithm can be defined as "*a sequence of instructions carried out to perform a task or to solve a problem*" [12].

To the non-expert reader, it would be easy to assume that the field of algorithmic composition is relatively new and resulted from the development of technological devices during the last 60 years. However, generating music using algorithmic composition is a long-standing compositional method. This technique can be dated as far as the ancient Greeks, where the Pythagoreans believed that numbers and mathematical properties could be used in the process of composing music, hence they believed that the field of mathematics and music were linked [18]. It is also worth noting that Pythagoreans identified an arithmetical relationship between harmonic intervals, which was an important addition to music theory [3], [10]. Guido d'Arezzo, widely considered the father of western music theory, applied rules for some of his compositions [16]. A famous composer who also used algorithmic composition techniques is

Mozart. In his famous piece *Musikalisches Würfelspiel*, he used a dice to sequence musical fragments he composed beforehand [13].
Technological advances, and more specifically computers, have expanded the scope of algorithmic composition. Research and techniques in algorithmic composition have increased largely during the last 60 years. Composers are able to implement algorithms in real-time and in live performance [12] whereas before the computer, composers had to process everything by hand, which was time consuming. One of the first musical pieces composed with computer-based algorithms was Hiller and Isaacson's *Illiac Suite*, written in 1956 [7]. One of the first computer systems for automatic composition was MUSICOMP (short for MUsic Simulator Interpreter for COmpositional Procedures) developed by Baker and Hiller in the early 1960s [6]. Since, numerous systems and approaches have been implemented for algorithmic composition [14], such as Cope's EMI [4] or CAMUS [12] by the second author, to name but two.
In this paper we introduce a computer-aided composition system using a Markov chain algorithm. Sections onward describe and explain the different steps of the generative system and present some examples. Then, we discuss some further developments and the paper concludes with final remarks.

2 Description of the Computer-Aided Composition System

To implement our computer-aided composition system, we decided to use the object-oriented visual programming software named OpenMusic[1] [1]. This environment, based on the programing language Common Lisp, is developed by the IRCAM[2] Musical Representations research group. We also use various OpenMusic objects from two external libraries, *Morphologie* [2] and *SOAL* [5].
We decided to work with MIDI files as data for our system due to it being a standard communication and control protocol, which also contains musical information that are easily retrievable. Furthermore, several OpenMusic objects are designed to work with MIDI files, which make it simple to manipulate musical data and export the resulted compositions. The system processes the information in three steps as detailed below.

Step 1: Information Retrieval

The first step is the extraction of musical characteristics from the monophonic MIDI file input by the user. Such a step is necessary due to the algorithm not being random: results are based on the original musical piece. The first feature the system identifies is the MIDI note, which is then multiplied by 100 to obtain its respective midicent. Next, the system retrieves the duration of each note and stores its millisecond value. The other characteristics are extracted in each bar of the musical input. The melody direction represents the motionless, descending or ascending movement of the melody

Fig. 1. First 4 bars of Bach's *Gavotte en rondeau, Partita for Violin No.3*.

[1] http://forumnet.ircam.fr/product/openmusic-en
[2] Institut de Recherche et Coordination Acoustique/Musique

TABLE I: MUSICAL FEATURES OF BAR 1 AND BAR 2 OF EXAMPLE IN FIG. 1.

	1	**2**	**3**	**4**	**5**	**6**	**7**
Midicent	7100	8000	8000	7800	7600	7800	8100
Duration (ms)	400	400	400	200	200	400	400
Melody direction	0	+1	0	-1	-1	+1	+1
Melody interval	0	9	0	2	2	2	3

stored as 0, -1 or +1, respectively. Note that the melody direction's value for the first note of each bar is 0, due to the absence of a previous note. The last musical features analysed is the melody interval. Here, for each bar, the current MIDI note value is compared with the previous one and it returns the distance between the two notes. Again, the value of the first note of each bar will be 0, due to there being no previous note. We also store the number of notes contained in each bar of the musical piece. Table I shows an example of the musical features extracted by the system, using the musical example in Fig. 1.

Step 2: Generative Process

The second step of the system is to generate musical sequences using the information retrieved from the original file. We decided to implement an algorithm based on a Markov chain [8]. Our rationale is the relative ease of implementation and its proven ability to generate interesting results for music [19], [11], [14]. We chose to use a second-order Markov chain for our approach. A first-order generates results close to randomness while a second-order adds more constraints and gives results more similar to the original data. An increase of the order requires more data to train the algorithm and it tends to limit the possibilities and results with sequences very similar to the original. However, we still want a part of pseudo-randomness and chance in the generative process of our system. Randomness is a compositional element often incorporated by practitioners in the field of algorithmic composition [13], [14]. We use the lists of midicents (notes), durations and number of notes per bar as the data for our second-order Markov chain algorithm. We apply the algorithm on the midicents and then on the durations. We use the number of notes to define the length of the dataset to generate. We also use an OpenMusic object to quantify the durations to get a rhythm structure for the musical sequence.

Step 3: Transformative Phase

We decided to not only use the sequences generated by the Markov chain, but also to give the user the ability to transform these musical sequences. The first transformation available is a pitch inversion. The maximum midicent value being 12700, to make a pitch inversion we need to subtract the current midicent value to 12700. The next transformation reverses the pitches in each bar. For instance, a list of pitches (A B C A) is transformed to (A C B A).

The third transformation combines data from the melody interval and melody direction and is calculated as follows:

$$(MI \times MD \times 100) + Midicent \quad (1)$$

where *MI* is the melody interval value and *MD* is the melody direction value. For example, if the list of melody interval values is (0 2 4 1), the list of melody direction values is (0 -1 1 -1) and the list of midicents is (7000 7300 7100 7900) the result of the transformation is (7000 7100 7500 7800). It is worth noting that the aforementioned transformations are bare resemblance to serial music techniques [17].

The last transformation integrates a random walk algorithm [9]. We constrained the algorithm to generate values comprised between the lowest and the highest midicent values contained in the musical sequence and not completely random values. We also use melody interval values to define the maximum step of the random walk.

All these transformations can be applied to the results generated by the second-order Markov chain algorithm.

User Control

We decided to give the user the ability to select which transformation they want to apply on the musical sequences generated. At the current stage of development, the user can input their selection using a slider object. The range of the slider indicates how different from the original the user wants the results to be. We simply use a ten-point scale with 0 representing small amount of difference, which is the result of the Markov chain generation, and 9 representing a large amount of difference. For the highest index, we apply all the transformations available to the result of the generative algorithm.

Furthermore, the user has the ability to edit the musical sequence directly in the program, before they export the result. This is feasible using the OpenMusic *voice* object to display the generated sequence. The user can also listen to the musical sequence, using a MIDI player integrated in OpenMusic.

Export Results

Once the user found an interesting result, it is possible to export it in order to use it in notation software or to play it using an appropriate device or instrument. We use two built-in OpenMusic objects to enable the exportation in two different file types. The first option is to export the musical sequence as a MusicXML file. This format encapsulates musical information that can then be imported in music notation software to arrange the musical score. The second option available is the exportation as a MIDI file. This is a practical format that gives the user the ability to play the sequence using MIDI instruments, to import the file in DAWs (Digital Audio Workstation) or also in music notation software for further edits.

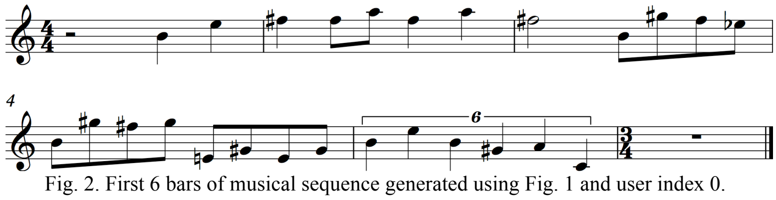

Fig. 2. First 6 bars of musical sequence generated using Fig. 1 and user index 0.

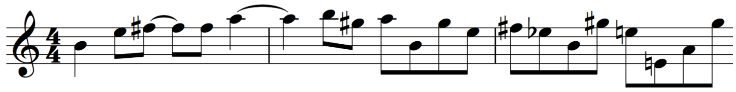

Fig. 3. First 3 bars of musical sequence generated using Fig. 1 and user index 4.

Fig. 4. First 4 bars of musical sequence generated using Fig. 1 and user index 9.

3 System Testing and Results

In this section, we present some examples to illustrate the operation of our computer-aided composition system. We used the MIDI file of Bach's musical piece *Gavotte en rondeau*, *Partita for Violin No. 3* as musical input for the system. Fig. 1 shows its first four bars. We generated musical sequences for all user index possibilities. Due to space constraints, we only present three examples. We decided to take results for user index 0, 4 and 9. We did not edit the results and we imported them in music notation software to generate the musical scores we include in this paper.

Fig. 2 shows the first six bars of a musical sequence generated with an index 0, which represents a desire of a little amount of difference from the original source. Hence, the system applies the Markov chain algorithm only on the midicents and keeps the original rhythm structure. No further transformation is applied. Fig. 3 represents a generation using index 4. The system generates pitches and rhythm using the Markov chain algorithm. Then, the program transforms the musical sequence using the combination of the melody direction and melody interval values. Finally, Fig. 4 shows the first four bars of a musical sequence generated using index 9. Here, the system generates pitches and rhythm using Markov chain. Then, it applies all the transformations, namely pitch inversion, reverse pitch, random walk on pitch and the combination of melody direction and melody interval data.

4 Final Remarks

In this paper, we presented a computer-aided composition system developed at the Interdisciplinary Centre for Computer Music Research (ICCMR). This system, designed to expand composers' musical creativity, uses an original MIDI file input by the user. After retrieving various musical information, as described in section 2, the system generates musical sequences using a second-order Markov chain algorithm. To go further than only generating compositions from a Markov chain, we developed a number of methods to transform the musical sequences. We decided to give the user the ability to choose the transformations, instead of applying them arbitrarily. This system has been used to aid composing symphonic pieces, notably *Symphony of Minds Listening* and *Shockwaves* composed by the second author.

At the current stage of development, we use a slider to determine the user index. This could be an area to explore for further developments. A possibility could be to use an EEG (electroencephalogram) device or other types of sensors to determine the user index. Another area of development could be the implementation of more transformations, maybe involving sensors in their process, to expand the musical abilities of this computer-aided algorithmic composition system.

References

[1] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue. "Computer-assisted composition at IRCAM: From PatchWork to OpenMusic." *Computer Music Journal,* Vol 23, no. 3, pp. 59-72, 1999.

[2] J. Baboni-Schilingi, and F. Voisin. *Morphologie: Fonctions d'Analyse, de Reconaissance, de Classification et de Reconsituition de Séquences Symboliques et Numeriques.* Software documentation. IRCAM, Paris, 1999.

[3] D. Cope. *Computers and Musical Styles.* AR Editions, 1991.

[4] D. Cope. *Experiments in musical intelligence*. AR editions, 1996.

[5] D. Guigue. *SOAL–Sonic Object Analysis Library–OpenMusic Tools for analyzing musical objects structure*. Software documentation. IRCAM, Paris, 2010.

[6] L. Hiller, and R. Baker. "Computer cantata: A study in compositional method." *Perspectives of New Music,* pp. 62-90,1964.

[7] L. Hiller, and L. Isaacson. *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill, 1959.

[8] R. A. Howard. *Dynamic Probabilistic Systems (Volume 1: Markov Models)*. John Wiley And Sons, 1971.

[9] B. D. Hughes. *Random Walks and Random Environments.* Oxford University Press, 1996.

[10] J. James. *The Music of the Spheres: Music, Science and the Natural Order of the Universe.* Springer Science & Business Media, 1995

[11] K. McAlpine, E. R. Miranda, and S. Hoggar, "Making music with algorithms: A case-study system." *Computer Music Journal,* Vol 23, no. 2, pp. 19-30, 1999.

[12] E. R. Miranda. *Composing Music with Computers.* Focal Press, 2001.

[13] W. A. Mozart, and H. Norden. *Musikalisches Würfelspiel.* B. Schott's Soehne, 1957.

[14] C. Roads. *The Computer Music Tutorial.* MIT Press, 1996.

[15] K. H. Rosen. *Elementary Number Theory and its Applications.* Reading Mass, 1988.

[16] O. Strunk. *Source Readings in Music History*. Vail-Ballou Press, 1950.

[17] A. Whittall. *Serialism*. Cambridge University Press, 2008.

[18] S. Wollenberg. "Music and mathematics: An overview." In *Music and Mathematics. From Pythagoras to Fractals*, by R. F. John Fauvel and R. Wilson, Oxford University Press, 2003.

[19] I. Xenakis. *Formalized Music*. Indiana University Press, 1971.