Łukasz MAZUROWSKI

Faculty of Computer Science and Information Technology, West Pomeranian University of Technology

doi:10.15199/48.2015.02.27

Algorithmic composition transformational-generative system for background music generation

Abstract. *In this work the problem of algorithmic composition of music has been presented. Author has attempted to locate presented solution between other related approaches within the musical algorithm positioning framework proposed by Woller et al. The main idea of the system is that the music composition process is performed by number of mini-models parameterized by further described properties. The mini-models generate fragments of musical patterns used in output composition. Musical pattern and output generation are controlled by container for the mini-models – a host-model. Overall mechanism has been presented including the example synthesized output compositions.*

Streszczenie. *W pracy przedstawiony został problem algorytmicznego komponowania muzyki. Autor dokonał próby ulokowania prezentowanego systemu komponowania pomiędzy innymi istniejącymi rozwiązaniami w tej dziedzinie z wykorzystaniem algorytmu pozycjonowania systemów algorytmicznego komponowania muzyki, zaproponowanego przez Woollera i innych. Główną ideą autorskiego systemu jest przeprowadzenie procesu komponowania muzyki z wykorzystaniem mini-modeli, sparametryzowanych przez zdefiniowane i opisane w pracy parametry. Mini-modele generują fragmenty wzorców muzycznych, wykorzystanych następnie w wyjściowej kompozycji. Generowanie wzorców muzycznych oraz kompozycji wyjściowej jest sterowane przez kontener zawierający mini-modele. W pracy został przedstawiony ogólny mechanizm powstawania kompozycji z załączonymi syntezowanymi przykładami kompozycji wyjściowych. (**Transformacyjno-generatywny system algorytmicznego komponowania muzyki tła**)*

Keywords: generative music, algorithmic music composition, background music, MIDI.
Słowa kluczowe: muzyka generatywna, algorytmiczne komponowanie muzyki, muzyka tła, MIDI.

Introduction

In the literature the term “generative music” is a subset for the wider set of the “generative art” and it refers to music that has been created with the use of an autonomous system. The meaning of autonomous system in this context is connected with non-human system which can independently create art (music) with the use of algorithm that determines a manner of generation.

In [1] Boden and Edmonds noted that “the terms of ‘generative art’ and ‘computer art’ have been used in tandem, and more or less interchangeably, since the very earliest days”. In the domain connected with this paper, the first time the computer systems were used to generate music was probably the “Illiac Suite for String Quartet” created by Hiller and Isaacson in 1967 [2]. Another early approach to the algorithmic computer music was a system called “Stochastic Music Program” by Xenakis that was completed in 1962. Details about this application were published in a form of essays on “Formalized Music” in the same year [3] and source code for this program was published in [4].

Lately, the term “generative music” has been created by Eno who was promoting and using generative art methods in his work [5]. In 2003 Galanter extended the term “generative art” by giving definition: “Generative art refers to any art practice where the artist creates a process, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is then set into motion with some degree of autonomy contributing to or resulting in a completed work of art”[6].

This work focuses on the generative algorithmic music composition system that generates various kinds of background music with the use of parameterized mini-models, which define possible musical fragments used in output composition. These mini-models constitute a host-model, in which author defines various transformational rules to generate music. The details about author’s model have been preceded by the presentation of the related work located within the framework for positioning musical algorithms.

Background

Author of this work has focused on the aspect of generating a background music. It refers to “music of any kind that is played while some other activity is going on, so that people do not actively attend to it” [7]. Background music plays a second role in the lives activities, but it can have an influence on many aspects of our behaviour. In [8] Kämpfe et al. analyzed the impact of music on various tasks in human live. The authors suggested that “global analysis shows a null effect, but detailed examination of the studies that allow the calculation of effects sizes reveals that null effect is most probably due to averaging out specific effects”. In this work several cases in which the background music has negative or positive impact on the activity process have been identified. Cockerton et al. in [9] examined the positive and negative effects of background music on cognitive test performance. Finally, Chaudhury et al. in [10] presented the role of the music in reprogramming brain activity. Authors reasoned that “music can trigger mechanisms of brain functions including learning and memory”.

These analyses are the basis for the attempt to create a system that could generate background music in a semi-automatic manner. Such a system could be a tool that allows to make a different kind of background music, according to the given activity. According to the work of Man-Kwan and Shih-Chuan [11] “current research on computer composition may be classified under two approaches based upon the method used to generate the compositional rules”. In the first approach the explicit rules are being used, which are specified by humans. Second approach uses the implicit rules that are derived from a sample piece of music.

Comparison framework for algorithmic music system by Wooller

The process of music composition in this kind of approach is controlled by the algorithm that generally produces music in the analytic, transformational or generative manner. Wooller et al. in [12] presented a framework that allows authors of various algorithmic music systems to locate their solutions between other approaches.

In Figure 1 some current applications and author's system within the algorithmic music framework have been presented. The algorithmic music systems are located on a plane described by two functions. The first function ranges from analytic, through transformational, to generative character of the algorithm. The second function ranges from narrow to broad musical context used in a process of music generation.

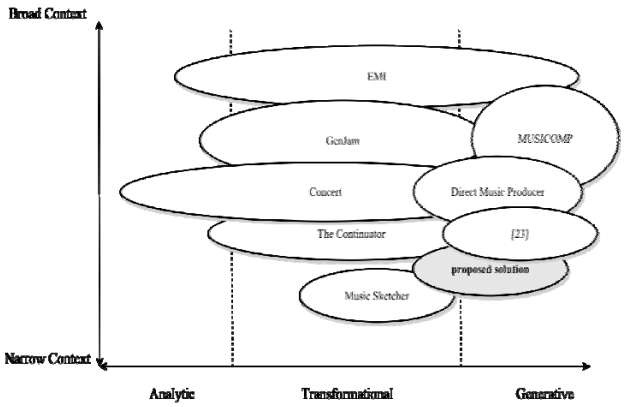

Fig.2. Some related works and author's system located within the algorithmic music framework presented in [12]

Wooller et al. clearly reviewed the meanings of the given functions ranges. Analytic musical algorithms according to [12] "tend to reduce the potential data size and the general music predisposition of the representation by extracting specific features". The example of this kind of algorithms can be described as a process that takes a set of musical sequences and outputs a set of notes. Transformational musical algorithms can only alter the information (they tend to not have a significant impact on the general musical predisposition of the data representations). The example of a transformational musical algorithm can be a process that transposes the individual notes or retrogrades the order of a phrase. Generative musical algorithms can be described as a process that outputs more complex sequence based on the reduced input data. For example, a process that takes a number as a seed and returns a sequence of notes is generative.

The second function describes the context for musical algorithms used in the given approach. Wooller et al. in [12] defined context as "the surrounding information that influences the computation of an algorithm and therefore an algorithmic music system". The context width in the musical manner can be presented as a note located on a musical score with a circle drawn around the note. The scope of the context is related to the circle surrounding the note. The more radius the circle has, the wider the context is (more notes are included inside the circle). Authors of [12] claimed that the width of the context depends not only on notes occurrences but also it can be influenced by more dimensions of a note (more parameters). This means that an algorithm that takes into account four parameters of a note (for example: onset, pitch, velocity and duration) has a broader context than a similar algorithm that considers only pitch parameter.

Taking the above definitions into consideration, author has placed his approach within the positioning algorithmic music framework as it has been presented in Figure 1. Before presenting the explanation of this placement, the next section contains the analyses of the related approaches that have been located by Wooller et al. within the framework.

Related work

The early solution of algorithmic music composition with the use of computer was the MUSICOMP software by Hiller and Isaacson presented in [2]. In the experiments authors explored different aspects of the music composition process in the algorithmic manner. In general, these experiments were based on modifications of "random white-noise music" using rules from counterpoint harmony and some of the processing music structure was transformed by adding chords or rules that limited the number of possible successive repetitions of a phrase. In the additional experiments of the MUSICOMP project the Markov process was used to control a consonance and dissonance instances by affecting the influence of various probability distributions. The use of tonality system in these models increased the contextual width by marking notes for tonal reference and relating the following notes to it. Overall experiments described by Hiller and Isaacson were positioned by Wooller et al. in [12] as a generative system that uses a broad musical context during the music generation process (Figure 1).

Other computer model was developed by David Cope and presented in [13] and [14]. The EMI project (Experiments in Musical Intelligence) focused on the use of recombination and grammars in the creative musical process. EMI was supplied by a manually incorporated musical structure, while phrase and measure level organization is generated with the use of ATNs – Augmented Transition Networks, and note organization by MATNs – Micro-Augmented Transition Networks [12]. The complex systems of grammar were used inside EMI. The rules complexity and the use of many dimensions of the algorithms input, denoting a wide musical context and the specific combinatorial constraints were determined through the analysis of the database of music. That is the reason EMI system has a wide range in the function of describing musical composition algorithm (see Figure 1).

Next solution presented by Mozer in [15] reviewed the experiments of note-by-note prediction. Mozer studied if a model, without explicit representations of a form, could learn structural trends that can be used to generate complex music compositions. In his Concert he used a third-order Markov chain to create an output composition. The system was initially trained to predict the next note in a sequence. The training process was based on the recursive feedback – the first notes of a sequences are given, then predictions are transferred to the input, and thus an entire output is predicted. The Concert was located by Wooller et al. in [12], in the ranges of functions presented in Figure 1.

IBM provided (in 1999) and abandoned a next algorithmic music system called Music Sketcher. Abrams et al. (the members of IBM Music Sketcher project) in [16-17] reviewed the structure of Music Sketcher and subsystems that provide different functionality in the music generation process. Music Sketcher was a meta-sequencer which generates music with the use of small pre-composed pieces of data called riffs. Users could transform a musical parameters (pitch, duration or velocity) via graphs controlling primitive arithmetic operations. Overall composition process was based on transformational procedures: musical riffs (patterns) can be rearranged by number of functions without affecting music predisposition of the data representation and size. According to Wooller et al. [12] the Music Sketcher is located within the framework inside the transformational algorithm class with the narrow context.

In 2001 Microsoft presented its tool for composing non-linear music called DirectMusic Producer (DMP) which was reviewed by Buttram in [18] as a tool for interactive music

composition for dynamic context like computer games. In DMP a basic musical structure is “segment” that defines a section of music that renders differently on each playback depending on user defined settings. The segment is a container for a different kinds of “tracks”. The track can hold linear sequence (“Sequence Track”) or bundle of user defined variations (“Pattern Track”), which are flipped between by the playback engine. In DMP the musical algorithm has a fairly wide context, because of the system of grammars that allows for high level relatedness between data and the algorithms that use it. Despite the fact that in DMP there is no analysis (random chord traversal and variation selection) and the algorithm uses recombination of patterns, the system can be classified as generative with a small functionality in the transformational range of Woller et al. [12] framework function (Figure 1).

Pachet in [19-20] presented The Continuator - a system for music improvisation which learns a Markov model of possible note sequences from a database of music. Major feature of the system is the use of a real-time “fitness function” to influence probabilities of the notes. Pachet proposed an algorithm which generates polyphonic music with the use of clustering notes within the same temporal region. The clustering rules base on explicit and implicit rules. For example, explicit rules of clustering define fixed metrical structure of the sequence, while implicit rules use a parameter to determine the processing of the sequences (for example cut-off point between overlapping notes and chords). According to Woller et al. [12] positioning framework, The Continuator is placed as an algorithm that uses analytic, transformational and generative techniques in music composition and has a fairly wide context due to musical input and a large number of instructional parameters.

Other described system is a solution proposed by Biles in [21-22]. He reviewed the system called GenJam which generates jazz improvisations and provide harmony to a human player. The GenJam database of music containing structural and motific data for the entire standard jazz compositions provides the algorithm with a predetermined global structure around which to operate. The major mechanism is based on Interactive Genetic Algorithm (IGA), in which local musical structure is derived from recombination and selection. The pitches are mapped to a chord progression and scale. The user defines the fitness of the lick combinations. GenJam can randomly combine phrases or use a function to ascertain the most rhythmically appropriate crossover point. Biles used a number of other heuristics to ensure a certain musicality of the output. With finite licks and potentially infinite musical output, the system can be classified as generative [22]. Woller et al. located the GenJam within the positioning framework as major transformational system with the use of broad musical context (input database of music).

The last described solution is a generative approach proposed by Zabierowski and Napieralski in [23]. The authors described the possibilities of generating music by implemented neural networks methods. In this approach the music creation process focuses on the harmonic sequence generation with the use of tonal harmony rules.

In the next section author has proposed a model of the transformational-generative system of algorithmic music composition with the use of musical patterns generation techniques. The example of the generated output of model has been presented with the use of a piano roll plot and digital audio files on the author’s webpage.

Transformational-generative system for background music generation

The overall structure of the proposed system has been presented in Figure 2 in the form of a Component UML Diagram.

The basic component for the system is a mini-model, that can belong to one of the following class: monophonic model, chord model or drum model. A mini-model can be described as:

$$Mm = \{C, NM, OPV, CMM, cc\} \tag{1}$$

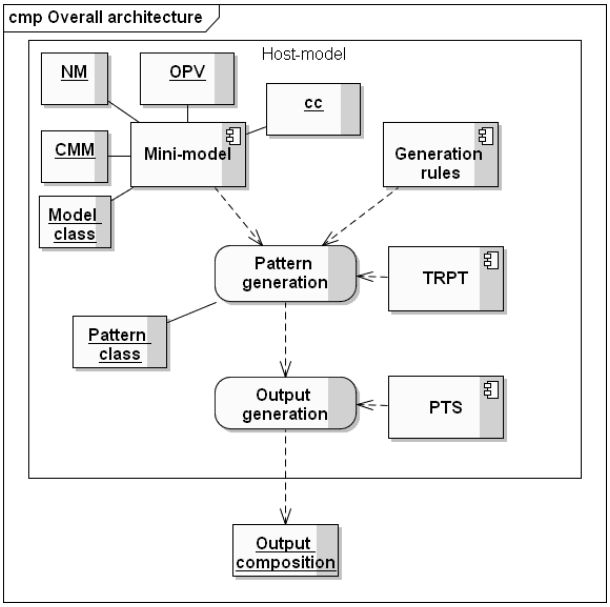

Fig.2. The overall structure of the transformational-generative system

The C is a Class of the given Mm and it defines the manner in which the Mm is used by a host-model. NM is an acronym of Note Matrix that has been proposed by Toiviainen and Eerola within the MIDI Toolbox framework and described in [24-25]. Table 1 shows an example of NM describing an octave C4-C5 according to the MIDI standard. The values in the first two columns of the matrix define the onset and the duration of the subsequent sounds (lines). In the 'MIDI channel' column there is a MIDI channel number to which the sound has been assigned. The 'Pitch' column includes sound pitches in the range 0-127 according to the MIDI standard. In the 'Velocity' column the velocity of sound (volume) has been defined, while the remaining sixth and seventh columns contain the onset and the duration of the sounds in seconds. In author model all durations of notes within NM have a constant fixed value defined by user.

Table 1. The example of NM describing an octave C4-C5 with constant duration values equal to 0.25 second

Onset	Dur.	MIDI	Pitch	Vel.	Onset	Dur.
0.00	0.42	1	60	100	0.00	0.25
0.42	0.42	1	62	100	0.25	0.25
0.83	0.42	1	64	100	0.50	0.25
1.25	0.42	1	65	100	0.75	0.25
1.67	0.42	1	67	100	1.00	0.25
2.08	0.42	1	69	100	1.25	0.25
2.50	0.42	1	71	100	1.50	0.25
2.92	0.42	1	72	100	1.75	0.25

OPV is an Onset Positions Vector that describes in which rows within NM the notes can be generated. OPV is generated with the use of the number of rows defined in NM as “l” and “rs” value that defines interonset interval (IOI)

refers to the amount of time between the start of one note and the start of the next note:

$$OPV = \{n : rs : l\}; n, rs < l \tag{2}$$

The CMM is a Combination Motifs Matrix (CMM), that defines possible combinations of pitch sequence defined by user. The CMM consists of k-element combinations C of the n-element pitch sequence arranged in rows in matrix with the size of "m" rows (number of the combinations) by "k" column (length of the defined pitch sequence). The CMM can be defined as:

$$CMM = \begin{bmatrix} {}_1C_k^n \\ {}_2C_k^n \\ \dots \\ {}_mC_k^n \end{bmatrix}, k = n \tag{3}$$

From CMM a single row referring to PM (Pitch Motif) is selected for a given mini-model that generates a pattern with the use of defined generation rule. The special kind of PM is a single constant value referring to mini models that generate sequences consisting of notes with one equal pitch value. This kind of mini models is being used to generate sequences for single drum instrument related to MIDI pitch value [26].

The "cc" is a coverage coefficient that defines on which position in OPV the notes defined in PM are generated. The cc is defined as:

$$cc = \frac{l}{ms}, ms \le l \tag{4}$$

The "ms" (model step) refers to value that defines how many notes in NM should be generated according to positions given in OPV. The greater "ms" value is, the smaller the IOIs are.

It is possible that a number of the mini-models uses the same PM, OPV and cc values. This situation is presented in the "Results" section.

Pattern generation

The number of the mini models is defined arbitrarily. A mini model can generate a simple drum sequence with the use of only one instrument, a drum sequence consisting of a group of instruments, a monophonic melody or chord progression. The behavior of a mini model is controlled by the host-model that is a container for the mini models participating in the musical pattern generation process. In Table 2 relationship between mini-model class and pattern generation rule has been presented.

Table 2. Relationship between mini-model class and pattern generation rule

Rule	Mini-model class	PM
R1	Monophonic model	vector
R2	Monophonic model	single pitch value
R3	Chord model	vector
R4	Chord model	single pitch value
R5	Drum model	vector
R6	Drum model	single pitch value

The host-model uses a given pattern generation rule for a given mini-model class. In Table 3 pattern generation rules and their descriptions referring to pattern generation behavior have been presented. These rules are used in example experiments in the "Results" section. Rules R4 and R5 have been defined only for major or minor basic chords generation and are assigned with the key-finding Krumhansl-Kessler algorithm which analyzes selected row from CMM [27-28]. This is a weak constraint for pattern generation process, but it could be extended by automatic feedback analysis of pattern key for monophonic model and chord model and adding ornamentations for generated chords.

After the pattern generation rule has been used, host-model selects transformational rule, that is used with the generated pattern, according to TRPT (Transformational Rules Probability Table). TRPT can be described as an extended version of probability tables used in musical context by Miranda in [29] on page 68. Besides selection probabilities defined for every transformational rule, author assigned transformational rule to given mini-model classes. The exemplary TRPT used in the presented research has been shown in Table 4.

Table 3. Defined pattern generation rules and their behaviour

Rule	Description of the behaviour
R1	First, fill every cc position in OPV with a given pitch vector. If there are more OPV positions than elements in pitch vector, then continue filling from the beginning of the pitch vector. Finally, assign generated pattern to the given MIDI channel.
R2	First, fill every cc position in OPV with a given pitch value. Then assign generated pattern to the given MIDI channel.
R3	First, create two copies of a given pitch vector. Next, transpose first copy by 4 or 5 (according to key) pitch up and second copy by 7 pitch up. Then, concatenate the given pitch vector with the copies and fill every cc position in OPV with a created concatenation. If there are more OPV positions than elements in the created concatenation, then continue filling from the beginning of concatenation. Finally, assign generated pattern to the given MIDI channel.
R4	First, create two copies of a given pitch value. Next, transpose first copy by 4 or 5 (according to key) pitch up and second copy by 7 pitch up. Then, concatenate the given pitch value with the copies and fill every cc position in OPV with a created concatenation. If there are more OPV positions than elements in the created concatenation, then continue filling from the beginning of concatenation. Finally, assign generated pattern to the given MIDI channel.
R5	First, fill every cc position in OPV with a given pitch vector. If there are more OPV positions than elements in pitch vector, then continue filling from the beginning of pitch vector. Then, assign generated pattern to MIDI channel 10.
R6	First, fill every cc position in OPV with a given pitch value. Then, assign generated pattern to MIDI channel 10.

Table 4. Defined exemplary transformational rules within TRPT

Rule	Description	Defined probability	Mini-model class
TR1	Transpose 4 pitch up	P1=0,125	Chord, monophonic
TR2	Transpose 7 pitch up	P2=0,125	Chord, monophonic
TR3	Transpose octave up	P3=0,125	Chord, monophonic
TR4	Transpose octave down	P4=0,125	Chord, monophonic
TR5	TR3, than TR1	P5=0,125	Chord, monophonic
TR6	TR3, than TR2	P6=0,125	Chord, monophonic
TR7	TR4, than TR1	P7=0,125	Chord, monophonic
TR8	TR4, than TR2	P8=0,125	Chord, monophonic

In the presented system generation of the pattern is connected with the pattern classes defined in the system (Figure 2). Pattern class describes from which mini-models a pattern should be generated. It is possible that pattern class consists of the patterns generated by the only one mini-model.

Pattern generation process is constrained by defined length of the used NM, duration value used in NM and number of the patterns to generate. These values are defined arbitrary. It should be noted that a small collection of the generated patterns results frequent repetitions within the output composition.

Output generation

Output generation process (Figure 2) is connected with the selection of previously generated patterns by defined mini-models. Within selection the Parncutt's durational accent model is used to compute the similarity between generated patterns [30].The similarity of generated patterns is defined as a Euclidean distance between each pattern in 12-components vector consisting of the probabilities of pitch-classes presented in [30]. Similarities of generated patterns are used to generate a Pattern Transition Structure (PTS, Figure 2) from which the patterns included in the output composition are selected.

The similarities are sorted for a given pattern from the most similar to the least similar pattern. During output generation the pattern selection process uses patterns with the similarity threshold defined by user. In authors experiments 50% from the most similar patterns are used during selection.

Output generation process is limited by the defined number of the patterns that should be used in the output composition. Selected patterns are repeated in the output composition according to the randomly chosen value from defined repeats array. This array consists of integer values defining how many selected patterns should be looped in the output. When the pattern is looped and added to the composition, the whole process of output generation is repeated until the limited number of the used patterns is reached.

Results

In the current research the model has been implemented with the use of MIDI Toolbox for Matlab [25]. The author defined following parameters describing an algorithmic composition process:

- NM duration value – 0.25 sec,
- NM length – 32 rows (in connection with the duration 0.25 sec, it lasts for 8 seconds),
- The number of the patterns to generate – 128,
- The number of the patterns to be used in the output composition – 64,
- Array of the patterns repetitions – a={2,4} (selected pattern is repeated two or four times within the output).

In Table 5 the example parameters for the 20 mini-models defined in the system has been presented. There were 13 drum mini-models, 6 mini-models generating monophonic melody and 1 mini-model for chord generation purpose.

The PM vector for 13th mini-model is selected from the example CMM defined as:

$$CMM = \begin{bmatrix} {}_1C(24,24,29,29,31,31,26,26) \\ {}_2C(24,24,29,29,31,31,26,26 \\ \dots \\ {}_mC(24,24,29,29,31,31,26,26) \end{bmatrix} \quad (5)$$

The mini-models from 14th to 20th use PM vector selected for the 13th mini-model. For the pattern transformational process the rules defined in Table 4 have been used. The pattern classes have been defined as it is presented in Table 6.

Table 5. The example of the mini-models parameters for the generation process

Mini-model	Class	OPV	cc	PM
1	Drum	[1:4:29]	2	36
2	Drum	[2:3:32]	2	36
3	Drum	[16:4:32]	3	40
4	Drum	[1:4:29]	5	38
5	Drum	[1:1:32]	2	42
6	Drum	[16:4:32]	3	47
7	Drum	[16:3:31]	3	48
8	Drum	[1:4:29]	3	54
9	Drum	[17:4:29]	5	[38,39,40,41]
10	Drum	[17:4:29]	5	[35,36,37,38]
11	Drum	[2:3:32]	2	51
12	Drum	[17:4:29]	5	57
13	Monophonic	[2:4:30]	2	from CMM*
14	Monophonic	[2:4:30]	5	same as *
15	Drum	[1:5:31]	2	same as *
16	Monophonic	[2:2:32]	3	same as *
17	Monophonic	[3:3:30]	2	same as *
18	Monophonic	[1:1:32]	2	same as *
19	Monophonic	[2:3:32]	3	same as *
20	Chord	[1:4:29]	2	same as *

Table 6. Defined exemplary pattern class generated by mini-models.

Pattern class	Generated by the mini-models
1	1,2,3,4,5,6,7,8,9,11,13,14,15,16,17,18,19
2	1,2,3,5,6,7,8,10,13,14,15,16,18,19
3	1,3,5,6,7,8,11,12,13,14,15,17,18,20
4	1,4,5,8,12,13,14,15,16,18,19,20
5	1,3,4,5,13,14,15,16,17,18,19,20
6	1,4,12,13,14,15,16,17,18,19,20

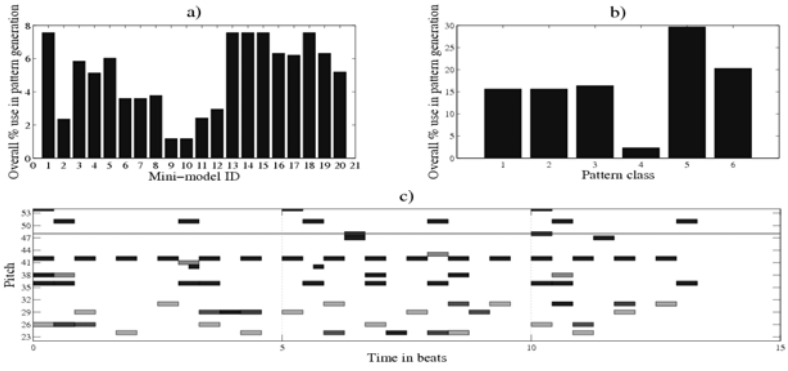

Fig.3. Overall % use of defined mini-models (a) and pattern class (b) in pattern generation. The piano roll plot for the example generated pattern from pattern class 1

In Figure 3 the selected overall statistics for the current experiments have been presented. The mini-models 9th and 10th have the lowest usage in the pattern generation process (Figure 3a), while the mini-models 1st, 13th, 14th, 15th and 18th have the highest usage (about 7,5%). In the current experiment pattern class 4 has the lowest usage during pattern generation, while the pattern class 5 has about 30% of the usage (Figure 3b). In Figure 3c the piano roll plot for the example pattern from class 1 has been presented, which is the pattern class consisting of the most complex patterns (generated by 20 mini-models, Table 6).

The output composition has been synthesized with the use of SynthFont application [31] and soundfont instruments bank Evanessenve2 to provide more realism to the listening MIDI music. The MIDI file has been converted to the MP3 file in order to keep equal quality of music for the listeners. The exemplary final composition generated by the described model is available on [32].

Conclusion

The presented system generates compositions that contain repetitions of the generated patterns. Output compositions fluctuate around base motif with the number of variations. Generative feature of the system could be proved by generating small pieces of music by mini-models and using them to generate more complex music. Transformational behavior of the system is connected with the use of probability table for the transformational procedures processing generated patterns.

Author has attempted to locate the system within the Woller et al. framework for musical algorithm positioning. The narrow musical context used in the current approach could be extended by defining CMMs and OPVs structures based on the analysis of the reference compositions and extracting characteristic motif from the given melody. Because of limited place for this work, author has only focused on a few aspects of current research. It should be mentioned, that instruments used in the synthesized version of output compositions have been selected arbitrary by author.

Further research is focused on extending current limitations of the model (i.e. constant durations value and simple chords generation) to use characteristics from the reference compositions that allow to define CMM, OPV and other mini-model parameters in automatic or semiautomatic manner. Number of generated outputs are available at ResearchGate as datasets called “Music composition generated by transformational-generative system…” [33].

REFERENCES

[1] Boden M.A., Edmonds E.A., What is generative art? *Digital Creativity*, 20 (2009), 21-46

[2] Hiller L., Isaacson L., Musical Composition with a High-Speed Digital Computer. *Journal of the Audio-Engineering Society*, 6 (1958), 154-160

[3] Xenakis I., Formalized Music: Thought and Mathematics in Composition. Indiana University Press, Bloomington, 1971

[4] Xenakis I.: Free Stochastic Music from the Computer. Programme of Stochastic music in Fortran. Gravesaner Blätter, 26, 1965, 54-92

[5] Eno B., Generative Music: Evolving metaphors, in my opinion, is what artists do. *Motion Magazine*, 1996, http://www.inmotionmagazine.com/eno1.html

[6] Galanter P.: What is generative art? Complexity theory as a context for art theory. *Proceedings of the 6th Generative Art Conference*, 2003

[7] Collins English Dictionary: background music. http://www.collinsdictionary.com/dictionary/english/background-music

[8] Kämpfe J., Sedlmeier P., Renkewitz F., The impact of background music on adult listeners: A meta-analysis. *Psychology of Music*, 39(2011), No. 4, 424-448

[9] Cockerton T., Moore S., Dale, N., Cognitive test performance and background music. *Perceptual and Motor Skills*, 85 (1997), 1435-1438

[10] Chaudhury S., Nag T.C., Jain S., Wadhwa S., Role of sound stimulation in reprogramming brain connectivity. *Journal of Biosciences*, 38(2013), No. 3, 605-614

[11] Man-Kwan S., Shih-Chuan C., Algorithmic compositions based on discovered musical patterns. *Multimedia Tools Applications*, 46 (2010), 1-23

[12] Wooller R., Brown A.R., Miranda E.R., Berry R., Diederich J., A framework for comparison of process in algorithmic music systems. Generative Arts Pracitce-2005. A Creativity & Cognition Symposium, 2005

[13] Cope D., Experiments in Musical Intelligence. Wisconsin: A-R Editions, Madison, 1996

[14] Cope D., Hofstadter D.R.: Virtual music: computer synthesis of musical style. MIT Press, Cambridge, 2001

[15] Mozer M.C., Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing. *Connective Science*, 6 (1994), 247-280

[16] Abrams S., Oppenheim D.V., Pazel D.P., Wright J., Higher-level Composition Control in Music Sketcher: Modifiers and Smart Harmony. *ICMC Conference Proceedings*, 1999

[17] Abrams S., Furher R., Oppenheim D.V., Pazel D.P., Wright, J., A framework for Representing and Manipulating Tonal Music. *ICMC Conference Proceedings*, 2000

[18] Buttram T., Beyond Games: Bringing DirectMusic into the Living Room. DirectX 9 Audio Exposed: Interactive Audio Development, USA: Wordware Publishing Inc, Texas, 2004

[19] Pachet F., The Continuator: Musical Interaction with Style. *ICMC Conference Proceedings*, 2002, 2011-2018

[20] Pachet F., Playing with Virtual Musicians: the Continuator in Practice. *IEEE Multimedia*, 9(2002), No. 3, 77-82

[21] Biles J.A., GenJam: Evolutionary Computation Gets a Gig. 3rd Conference on Information Technology Curriculum, Rochester, New York, 2002

[22] Biles J.A., GenJam in Transition: from Genetic Jammer to Generative Jammer. International Conference on Generative Art, Milan, 2002

[23] Zabierowski W., Napieralski A., Generation of harmonic sequences in accordance to tonal harmony rules with artificial neural networks. *Elektronika-Konstrukcje Technologie Zastosowania*, 11 (2009), 22-24

[24] Eerola T., Toiviainen P., MIR in Matlab: The Midi Toolbox. *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004, 22-27

[25] Eerola T., Toiviainen P.: MIDI Toolbox: MATLAB Tools for Music Research. University of Jyväskylä: Kopijyvä, 2004

[26] Computer Music: General MIDI Percussion Key Map. http://computermusicresource.com/GM.Percussion.KeyMap.html

[27] Krumhansl C.L., Cognitive Foundations of Musical Pitch, Oxford University Press, New York, 1990

[28] Huron D., Parncutt R.: An improved model of tonality perception incorporating pitch salience and echoic memory. *Psychomusicology*, 12 (1993), 152-169

[29] Miranda E.R., Composing Music with Computers (Music Technology). Focal Press, 2001

[30] Parncutt R.: A Perceptual Model of Pulse Salience and Metrical Accent in Musical Rhythms. Music Perception, 11 (1994), 409-464

[31] SynthFont, http://www.synthfont.com/.

[32] Example background music, https://www.researchgate.net/publication/263311140_Example_background_music.

[33]ResearchGate, https://www.researchgate.net/profile/ukasz_Mazurowski/publications.

***Authors**: mgr inż. Łukasz Mazurowski, Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, Wydział Informatyki, ul. Żołnierska 49, 71-210 Szczecin, E-mail: lmazurowski@wi.zut.edu.pl;*