

Automatic Melody Generation using Neural Networks and Cellular Automata

Ivana D. Matić, António Pedro Oliveira, and Amílcar Cardoso

Abstract— This paper discusses solutions for generating melodies in the context of a system that intends to produce music with a specified emotional content. The research sets up from an existing system, which produces music by manipulating and combining MIDI music. We aim to analyze the benefit from using automatic music composition techniques in order to improve musical quality, while conforming with the desired emotional content. We considered Neural Networks and Cellular Automata for the task.

Index Terms—Neural Networks, Cellular Automata, Emotions and Music, Automatic Melody Generation.

I. INTRODUCTION

THROUGHOUT the history, many researches in psychology have shown that music is powerfull way of affecting human emotions. That is popular especially in the domain of movies, video-games, theatres, in supporting music concerts, but also in hospitals as a way of therapy. For these reasons, a system that produces music with appropriate emotional content would be very useful. António Pedro Oliveira created a system that generates melodies with specified emotional content – EDME (Emotion Driven Music Engine) [1].

EDME is a Knowledge-Based system that automatically generates melodies with appropriate emotional content. The input of the system is an emotional specification, represented in a discrete dimensional space with horizontal and vertical axes - representing respectively degree of arousal (activation/relaxation) and valence (pleasantness/unpleasantness), according to Russell's model [2] (See Fig. 1).

There is an interface that controls the system in real time by using input values of arousal and valence. This interface enables the user to choose emotion from a list of emotions or to directly type values of arousal and valence. Each emotion from the list has its place in the diagram (Fig. 2). For example, if user wants to hear melody expressing emotion delighted, it is represented as a Valence, Arousal = (0.8, 0.4).

Ivana D. Matić is with the Department of Electronics, School of Electrical Engineering of the University of Belgrade, Serbia, (corresponding author's phone: +381644671508; fax: +38134701551; e-mail: ivana@dei.uc.pt).

António Pedro Oliveira is with the Centre for Informatics and Systems (CISUC), University of Coimbra, Portugal, (e-mail: apsimoes@dei.uc.pt).

Amílcar Cardoso is with the Centre for Informatics and Systems (CISUC), University of Coimbra, Portugal, (e-mail: amilcar@dei.uc.pt).

The chosen emotion proceeds to the rest of the Emotion Driven Music Engine (EDME), which is responsible for the other operations for production of appropriate music. EDME is an adaptive and flexible system consisting of four main modules which control emotional content of music: Music segmentation, Music classification, Music selection, Music transformation, three secondary modules, necessary for work of main modules: Music Feature Extraction, Music sequencing, Music synthesis, and four auxiliary structures: Music Base, Pattern Base, Knowledge Base, Library of Sounds, which store content for mentioned modules.

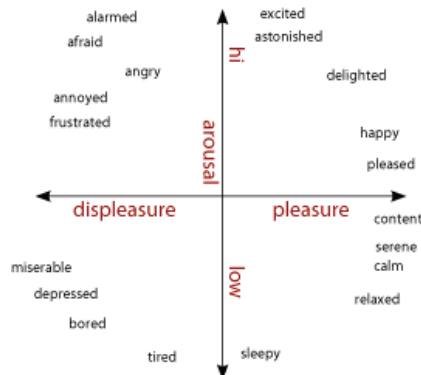


Fig. 1. Two-dimensional space for presenting emotions (from [2]).

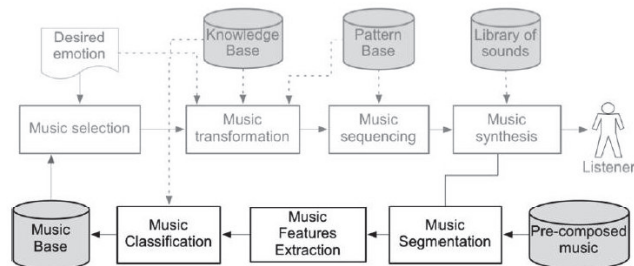


Fig. 2. EDME architecture (from [1]).

System EDME (Figure 2) works in two stages – online and offline stage:

Offline stage includes process: Compositions of different music styles (in MIDI format) are placed in the Pre-composed music module. Music segmentation module analyses features of these compositions and divide them into music segments with only one emotion. The module for extracting features takes each segment, extracts its features and attaches them to the segment as a label. The module for music classification uses the Knowledge Base to classify segments according to their characteristics. Then, classified segments with information about their features are stored in Music Base. In

the online stage, the Music Selection module gets the desired emotion and, according to it, determines which of the segments in Music Base is the closest to the emotion, and takes those segments with its coordinates in a smallest Euclidean distance from the mapped emotion. The segments that best fit, go to the module Music Transformation. Music Transformation module approximates the emotional content of these segments to the desired emotion changing emotionally relevant features with assistance of the Knowledge Base and the Pattern Base. The module Music Sequencing arranges segments, with assistance of Pattern Base, to form the melodies. Finally, Music Synthesis module converts formed MIDI file into audio.

The module for classification computes the estimated value for arousal and valence according to a weighted sum of the values obtained during the features extraction process:

$$Valence = \sum_{i=0}^n valenceFeatureWeight_i * FeatureValue \quad (1)$$

$$Arousal = \sum_{i=0}^n arousalFeatureWeight_i * FeatureValue \quad (2)$$

where each feature's weight represent its relative relevance to valence/arousal, according to the experiments with human subjects. The estimated values are within the interval [0..1] for both valence and arousal.

Music transformation intends to make segment's content closer to the desired emotion by decreasing the Euclidean distance. For example, if we have emotion with its coordinates (0.95, 0.4), (where the maximum is Valence, Arousal = [-1, 1]), and the segment with the closest emotional context is (0.5, 0.4), the dimension of the arousal does not have to be changed, but the dimension of the valence has to be changed, using equations [1].

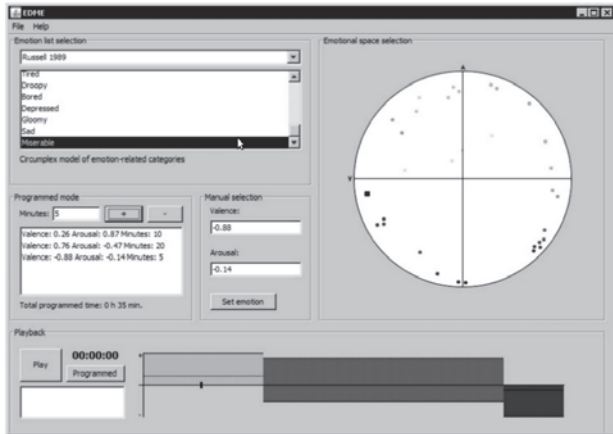


Fig.3. User interface of EDME (from [1]).

The Pattern Base contains descriptions of possible ways of organizing segments into songs. For example, segments can be arranged in AABA or ABAB pattern, etc. All rules contained in Auxiliary structures were assembled manually.

In general, during melody reproduction, EDME quickly adapts music to new ordered emotion.

Reasons for proposing improvement of EDME system

EDME currently depends on a music database and it would benefit from having the ability to compose music without that

dependency. Furthermore, the music produced currently is perceived as a collage, despite several smoothing mechanisms. The coherence of the music would increase if other composition solutions would be used. That is why we decided to investigate possibilities for giving EDME the ability to automatically compose music.

The music composition task as a whole is a very complex task, and therefore it is important to tackle the problem by steps. Composition involves the definition of several aspects that must mutually articulate, like rhythm, harmony, melody, instrumentation, overall structure, etc. In this paper we have decided to focus just on melody composition because it is a relatively narrow field and less hard in terms of complexity. Other aspects of composition will be tackled in further steps.

In the remaining of this paper we will analyze two alternative techniques for automatic generation of melody: Neural Networks and Cellular Automata. These techniques were selected for being the most frequently referred in the literature for this task. The analysis will take into consideration that we intend to produce music from an emotional description.

II. GENERATING MELODY WITH EVOLVING RECURRENT NEURAL NETWORKS

As it is well known, a neural network simulates behavior of the human neural system. In this paper, we investigate how to compose melody using neural networks. The idea is to train NNs on a set of melodies, from a composer or from a given style, and have the NN generating new ones, preserving stylistic characteristics of the original set.

Given the temporally cyclic character of most music, pattern repetition is a central element of most of the music composition techniques, which makes the more conventional three-layered feed forward network not suitable for the task. Recurrent Neural Networks (RNN) provide a mechanism to store a number of previous events, which makes them a viable alternative. Early examples of RNNs for melody generation are the works of P. Todd [3], with a 3-layer recurrent Jordan net, and M. Mozer [4], with a recurrent Elman net trained with a modified back-propagation algorithm.

Chun-Chi J. Chen and Risto Miikkulainen [5] have created melodies with Recurrent Neural Networks. Their approach consisted in evolving RNNs to find the most suitable configurations for the task. A set of composition rules (some of them from Bela Bartok) was used to evaluate the fitness of the networks for the composing task.

The approach followed in this work seems particularly suited for generating melody in EDME, as we may use its Knowledge Base as a source for evaluating fitness, i.e., one may evolve a population of RNNs where those which are more capable to generate melodies according to an emotional specification will have more probability to survive.

Figure 4 presents the input/output scheme of the NN used in [5]. The input layer represents a measure at time T and the output layer represents the measure at time T+1. This means that one feeds the SRN network on the input with a current measure from the output. This way, the network can generate output sequences from an initial starting point. The network is

fully connected within the forward direction, and its forward weights are evolved.

To avoid large complexity and possible overloading of system, the authors limit the available rhythmic notation to five types: a whole note, a half note, a quarter note, an eighth note and a sixteenth note. Rests and dotted notes are not used. The range of pitches is also shortened to three octaves, C2 to C5.

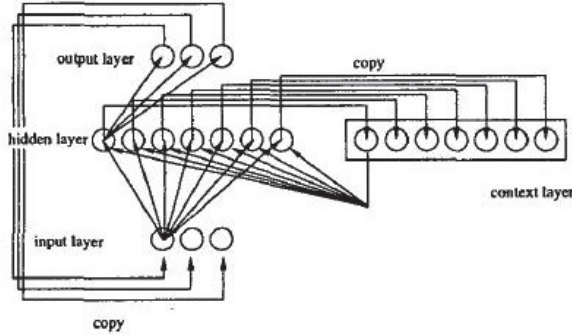


Fig. 4. The melody generation network (from [5]).

Relative pitch is represented with an array of nodes that correspond to interval steps, relative to reference notes (Figure 5). Each node is separated by one half-step, the smallest unit in tone. The most left node has the most negative number, and the most right node has the most positive number. The middle node represents zero difference in pitch. So, as each output node corresponds to an increase or decrease in pitch of a half step. The node that has the highest value wins. In example, shown in Figure 5, the winner is -4. If the reference note is G4, then the result is G4 - 4 = D4#.

There might be the case when output will drive the pitch out of available range of pitches in representation. The solution to this problem is to decrease/increase the pitch by an octave to get perceptually the closest pitch (in available range).

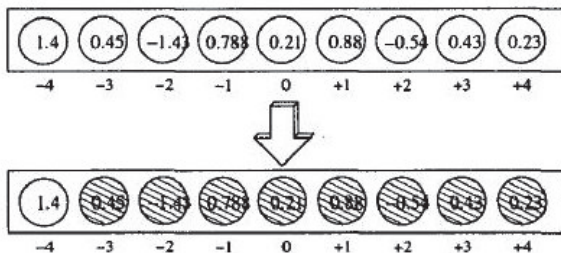


Fig.5. Relative pitch representation (from [5]).

As for duration an array of 5 nodes is used. The idea is similar with pitch representation, with the difference that each node corresponds to the duration, instead of specific offset. The node with the highest value wins and its duration is assigned to a current note. If more than one node ends up with a same value, the node that corresponds to a highest value wins.

The melody is generated using the measure as the building unit. The measure corresponds to one length of time and the duration of all notes inside one measure must add up to that length. One unit of the time is the length of the measure so we can fit exactly one whole note or two half notes or four quarter notes or any combination of notes that sum up to one in a

measure. The duration representation and pitch representation are arranged in concatenation called D-P pair. A measure representation consists of sixteen D-P pairs (Figure 6). A measure is extracted from this representation according to the following algorithm: One D-P pair is decoded at a time starting from the most left pair on the output layer. The duration of the note is summed to a variable T. If the duration in T has not exceeded one unit of time, we decode next D-P pair. If T has exceeded one unit of time, we take only a portion of the duration to make T exactly one unit of time.

The authors [5] use a genetic algorithm (GA) to evolve the networks in order to produce better melodies. There is a population of NN where each one produces a melody with some quality. The evolutionary process intends to result in NNs that are more capable to produce best melodies. In the case of EDM, the GA will analyze features from the melody to compute a fitness value according to the intended valence and arousal. NNs producing melodies with greatest fitness value will have more probability to reproduce (see Figure 7).

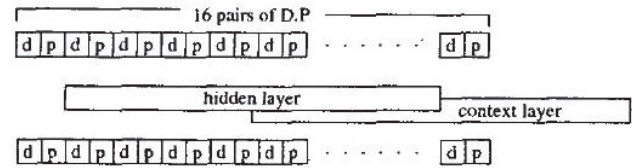


Fig. 6. Duration and pitch pairing, (from [5]).

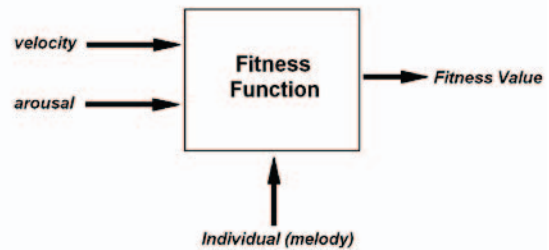


Fig.7. Fitness Function module

The fitness value of each melody represents how close are the estimated arousal/valence values of the melody, computed by applying the formulae (1) and (2), from the intended arousal/valence. In the application of the referred formula, only the features relative to melody are used, the remaining ones (relative to harmony, rhythm, etc.) being considered null. The exact formula is:

$$FV = 1 - \frac{\sqrt{(Valence_{in} - Valence_{est})^2 + (Arousal_{in} - Arousal_{est})^2}}{\sqrt{2}}$$

where $Valence_{in}$ and $Arousal_{in}$ are the emotional specification and $Valence_{est}$ and $Arousal_{est}$ the estimated values for the current melody. The formula takes into consideration that the values for valence and arousal in EDM are represented in the scale [0..1].

III. CELLULAR AUTOMATA

An alternative way of creating melody is by using Cellular Automata (CA) [9]. A cellular automaton (pl. cellular automata) is a discrete model studied in computability theory, mathematics, physics, complexity science, theoretical biology

and microstructure modeling. It consists of a regular grid of cells, each in one of a finite number of states, such as 2 states "On" and "Off" or more states that are usually presented by different colors. The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighborhood (usually including the cell itself) is defined in accordance with the specified cell. For example, the neighborhood of a cell might be defined as the set of cells a distance of 2 or less from the cell. An initial state (time $t=0$) is selected by assigning a state for each cell. A new generation is created according to some fixed rule, (generally, a mathematical function), that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood. For example, the rule might be that the cell is "On" in the next generation if exactly two of the cells in the neighborhood are "On" in the current generation, otherwise the cell is "Off" in the next generation. Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and it is applied to the whole grid simultaneously, although exceptions are known.

The idea was to have cellular automata with its cells in discrete states, so each cell can be in one of a finite number of states (2, 3 or more). In our case each cell in CA can be in one of three states: quiescent, depolarized and burned. If the voltage of cell V_i is less than V_{min} (some specified value) the cell is in quiescent state. If the voltage V_i is between values V_{min} and V_{max} the cell is in depolarized state and if the voltage is greater than V_{max} the cell is in burned state. In this system melody is divided into short grains that last few milliseconds. Each cycle of CA produces one sound-grain and that is how CA is used to produce melody. The CA-based grain controller uses an adapted version of a CA that has been used to model the behavior of various oscillatory phenomena, such as Belousov-Zhabotinsky chemical reactions [12,13]

A burned cell at time i is automatically replaced by a new quiescent cell at time $t+1$. In addition to the fundamental parameters mentioned earlier, the functioning of the automaton is also determined by the number n of possible cell states (or colors) such as $n \geq 3$, and the dimension (X,Y) , where (X,Y) are coordinates of the grid.

$$\text{IF } m_{x,y}[t] = 0 \text{ THEN } m_{x,y}[t+1] = \text{int}\left(\frac{A}{r_1}\right)$$

$$\text{IF } 0 < m_{x,y}[t] < n-1 \text{ THEN } m_{x,y}[t+1] = \text{int}\left(\frac{S}{A} + k\right)$$

$$\text{IF } m_{x,y}[t] = n-1 \text{ THEN } m_{x,y}[t+1] = 0$$

where the state of a cell at the time t is denoted $m_{x,y} = [t]$; x and y are the horizontal and vertical coordinates of a cell respectively; A and B represent the number of collapsed and depolarized cells, amongst the eight neighbors respectively; S stands for the sum of the neighbors' states; r_1 and r_2 represent the cell's resistance to become depolarized and k is the electrical capacitance of the cell [6]. All states of cells have to be determined by some boundaries of voltage. According to that transition of the states in a cell will cause the changes in its electrical capacitance. Choosing the right voltage boundaries for each state we affect the electrical capacitance

of the cell. Consequently, with electrical capacitance, and also with resistances r_1 and r_2 , quasi-periodical oscillations of the cellular automaton can be controlled. The coordinates of the cells are associated to frequencies and amplitudes, and oscillators are associated to a group of cells, or sub-grid. The frequency F and the amplitude A , values for the oscillators, are determined by the arithmetic mean over the frequency and the amplitude values associated to the states of the cells of their corresponding sub-grid.

$$F_i = \frac{\sum_{p=1}^P \phi_p}{P}$$

$$A_i = \frac{\sum_{p=1}^P \tau_p}{P}$$

where ϕ_p and τ_p are the frequency and amplitude of cell p , and P is the total amount of cells of the sub-grid. So this is the way we can have music generator that takes frequency and amplitude as parameters, and gives music on the output directly [6].

Having the notation, we can connect this part of the system with a rhythm generator and get the music on the output ready for MIDI music generation. There is another way of connecting CA with melodies: instead of producing grains, it can give us notations directly, that can be used for generating music in MIDI format. The systems called "CAMUS" [7] and [16] made by Eduardo Reck Miranda can be applied in this purpose. Each cell has its coordinates; when changes happen it will give a signal to the generator of notes. Then generator will take coordinates of mentioned cell, because the information that about chord is in coordinates. X - coordinate gives the distance in semitones between first two notes in chord and Y -coordinate gives the distance in semitones between second and third note in chord (Fig. 8). Chords contain fundamental information about melody, so A. Sheh, D.P. Ellis used this characteristic in [14] and K. Lee, M. Slaney in [15] as well.

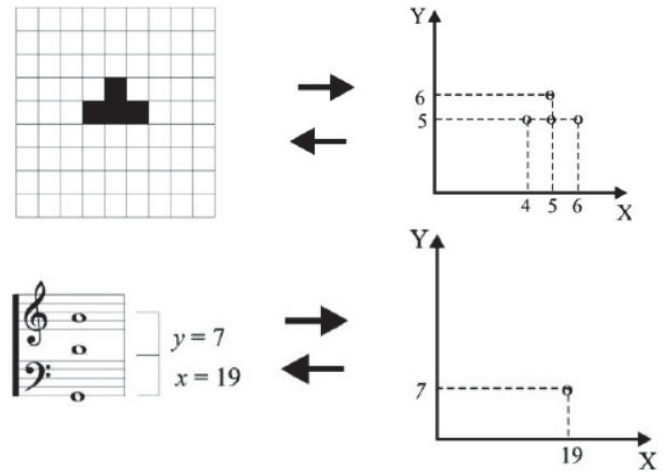


Fig. 8. CAMUS [16] uses Cartesian model to represent a triple.

To generate chord we need fundamental tone. In this purpose we can use Generator of Fundamental Tone (GFT). It can be based on the principle of Evolving Neural Networks. Therefore, it would take more NNs, which will produce

sequence of fundamentals. We can train NNs to learn how to think, by very harmonic compositions. Also, NNs can have knowledge of music theory - (the rules important for its function). For example, it is usual that composition starts in one tonality and ends up in the same tonality, or that changing tonalities sounds natural if it changes in the Circle of Fifths:

C – G –D –A –E –H –Fis –Des –As –Es –B –F –C

It is also possible to use more NNs and then check which sequence of fundamentals is the best and apply the best one; or, to use only one NN (learned well) in that purpose. GFT will produce actually only numbers that represent tone intervals, and the system will generate real fundamental (and then real chord) when user gives tonality. We left freedom of choosing tonality to user because it doesn't change important things, and it makes work easier to GFT.

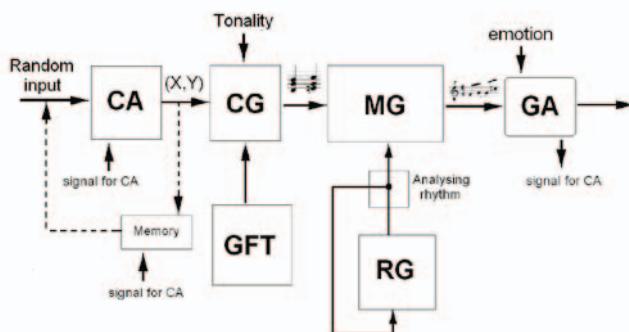


Fig. 9. System for producing melody with emotional content using CA

Combining outputs of CA and GFT with tonality in a new module Chord Generator (CG), we will get a sequence of chords. Each chord corresponds to one measure - tact. We combine them with output from Rhythm Generator (RG), in a new module that produces melody. D. Horowitz presented an interesting rhythm generating system in [11], which might be applied as RG. Also, one possible solution is given in [8]. Chords play important part in a melody - they can define a base of composition, so adding a tempo, rhythm and some other elements, we can obtain melody, composition or song. George Papadopoulos and Geraint Wiggins involved this approach in generating jazz melodies using Genetic algorithms [10].

In our system, on the output point rhythm is analyzed and checked so if density and speed corresponds to the required emotion, generated rhythm proceeds, and if not, RG should generate new better rhythm. Part of the system that generates melody has information about tonality and knowledge how to generate notes. For example, if we have 4/4 measure and all odd placed notes are accented, (information about accentuation comes from RG), then MG will generate 1. and 3. note, according to chord, and second note will be generated correctly with a fine crossing (jump), from 1. to 3. pitch, without big jumps. Then it's going to look at next measure to generate 4. note. In this case it is considered that MG "knows" tonality.

After producing melodies we can do exactly the same extension as we did for Evolving Recurrent Neural Network -

Genetic Algorithm. Therefore, on the output of Melody Generator (MG) GA checks melody in same way as it has done for generating melody with Evolving Recurrent NNs. Therefore, if melody doesn't match well with emotion, MG send signal to CA to generate new cycle of states. About CA, we can arrange that it takes random input or to take the output of last cycle that is saved in some RWM memory. And then all cycle repeats. We considered that this system is good enough to produce good melodies in few cycles, so it won't be infinite. It is recommended to find average interval of intervals of jumps (between each two notes - Figure 10) in analysis of melody, because it is also indication if melody suits with emotion.



Fig. 10. Intervals in semitones

IV. CONCLUSION

In this paper we analyzed alternative techniques to use in the automatic generation of melody for a software tool that intends to produce music according to an emotional specification. We focused our investigation in two alternatives: Neural Networks and Cellular Automata. In the case of Neural Networks, it is important to train them with a convenient corpus. Using the already available knowledge base of EDME, it seems promising to adopt an evolutionary approach in order to build the NNs that show to be more fit to the task. On the other hand, Cellular Automaton gives a wide range of relations between its cells, and according to that, more possible solutions of melodies' quality. Anyway, for both methods it is important to find adequate algorithms and formulas for evaluating fitness values, i.e. how much produced melody and given emotion are related. It is important to possess enough knowledge about which features have a more relevant role in the expression of the appropriate emotion. Fortunately, EDME already disposes of such knowledge.

REFERENCES

- [1] A. P. Oliveira and A. Cardoso, "A musical system for emotional expression", *Knowledge-Based Systems* 23, 2010, Pages 901-913
- [2] J. A. Russell, "A Circumplex Model of Affect", *Journal of Personality and Social Psychology* (1980), Volume: 39, Issue: 6, Publisher: American Psychological Association, Pages: 1161-1178
- [3] P. M. Todd, "A connectionist approach to algorithmic composition". *Computer Music Journal*, 13/4, 1989
- [4] M. C. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing". *Connection Science*, 1994
- [5] Chun-Chi J. Chen, Risto Miikkulainen, "Creating Melodies with Evolving Recurrent neural Networks", *INNS-IEEE International Joint Conference on Neural Networks*, Pages 2241-2246, IEEE, Piscataway, NJ, 2001.
- [6] E. R. Miranda, "At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestra and the Origins of the Melody", *Journal Evolutionary Computation*, Summer 2004, Vol. 12, No. 2, Pages 137-158

- [7] E. R. Miranda, *Composing Music with Computers*, Oxford: Focal Press, 2001
- [8] Nao Tokui, Hitoshi Iba, *Music Composition with Interactive Evolutionary Computation*, July, 2002. Available at <http://www.iba.t.u-tokyo.ac.jp/papers/2000/tokuiGA2000.pdf>
- [9] G. Nierhaus, "Algorithmic Composition", *Algorithmic Composition: Paradigms of Automated Music Generation*. Germany: Springer Verlag, 2009.
- [10] G. Papadopoulos, G. Wiggins, *A Genetic Algorithms for the Generation of Jazz Melodies*, AISB Symposium on Musical Creativity, 1999, Available at <http://citeseer.ist.psu.edu/87682.html>
- [11] D. Horowitz, "Generating Rhythm with Genetic Algorithms", *Proceedings of The National Conference On Artificial Intelligence* (1994), Volume: 142143, Publisher: : JOHN WILEY & SONS LTD, Pages: 1459–1459
- [12] A. Turner, "A simple model of the Belousov-Zhabotinsky reaction from first principles". Bartlett School of Graduate Studies, UCL: London, UK, 2009.
- [13] A.K. Dewdney. The hodgepodge machine makes waves. *Scientific American*. August 1988. p. 104.
- [14] K. Lee, M. Slaney, "Automatic chord recognition from audio using an HMM with supervised learning", in *Proceedings of the International Symposium on Music Information Retrieval*, 2006, Volume: 22, Issue: 1, Publisher: ACM Press.
- [15] A. Sheh, D.P. Ellis, "Chord segmentation and recognition using EM-trained hidden Markov models", in *Proceedings of the International Symposium on Music Information Retrieval*, 2003.
- [16] E. R. Miranda, "Evolving Cellular Automata Music: From Sound Synthesis to Composition", *Proceedings of the Workshop on Artificial Life Models for Musical Applications - ECAL 2001*, September 2001.