Music Composition using the EMILE Grammar Inductor

David Ortega-Pacheco and Hiram Calvo

Centro de Investigación en Computación, Instituto Politécnico Nacional,
Av. Juan de Dios Bátiz s/n, esq. Av. Mendizábal, México, D. F., 07738. México

david82d@gmail.com, hcalvo@cic.ipn.mx www.hiramcalvo.com

Abstract: We use the grammar inference engine EMILE for automatic music generation. The input to this algorithm is a collection of texts and then, based in the context of each word, it infers certain grammar rules. This engine has been used mostly to infer natural language grammar rules. In this work, we used the collections of MIDI music files from three classical music composers: Johann Sebastian Bach, Frédéric Chopin and Claude Debussy. We processed each file so that each sentence is formed from the notes found in a musical bar; each word is a musical note with a specific length and volume. We experimented with the capability of creating non-crossing compositions following Pedro P. Cruz-Alcázar, Enrique Vidal-Cruz, 1997. That is, we trained the grammar inductor with the compositions of certain composer, then generated a new composition, and finally we compared it statistically with the original compositions to obtain a degree of similarity. We expected the compositions C, obtained when training with a composer *A* to have a lower coefficient of similarity when they are compared with the corpus of composer *B*. These results are presented in incremental intervals to analyze the impact of the size of the training corpus in the grammar induction process.

Keywords: Music Composition, Grammar Induction, EMILE Grammar Inductor.

1 Introduction

There are several techniques for automatic music composition such as Markov chains [4, 9], fractals [16], chaotic systems [10], cellular automata [3], artificial neural networks [11], grammars [2], etc. In particular, systems for music composition based on grammars have used algorithms such as L-systems [13], and grammar induction algorithms like Sequitur [5].

The goal of grammar induction is to learn in an unsupervised way the syntax of a particular language from a corpus of this language. Grammar induction algorithms are used in several areas, for example computational linguistics, natural language processing, bio-informatics, time series analysis, computer music, etc. Particularly, for automatic music composition and music style recognition some commonly used algorithms are ECGI, K-TSI and ALERGIA [6, 7].

Advances in Artificial Intelligence and Applications
Research in Computer Science 32, 2007, pp. 341–351

Received 24/06/07
Accepted 31/08/07
Final version 30/09/07

In this work we use one of the existing algorithms for grammar induction: EMILE (Entity Modeling Intelligent Learning Engine) [1, 8, 15] for composing music automatically. As far as we know, EMILE has not been used for this purpose.

A particular problem we faced was evaluation—many works which use formal grammars for musical composition generally do not include an objective analysis of the resulting compositions; it is very difficult to tag neutrally if a musical composition is *good* or not [6, 7]. As an immediate workaround, we propose carrying out a statistical test of similarity between each new composition k_i versus each training corpus C_j to obtain a degree of similarity between the new composition k_i and each training corpus C_j. The degree of similarity may define the quality of each new composition k_i, and finally the average of the similarities of various compositions may define the algorithm performance for this application.

2 EMILE Grammar Inductor

EMILE (Entity Modeling Intelligent Learning Engine) is an unsupervised method for grammar induction and it is a method based on categorical grammars. The algorithm tries to obtain the language's grammatical structure using positive samples. The EMILE analysis is based on finding each word's context into each sentence (delimited by a dot (.)) in the training corpus and then carrying out two clustering processes for obtaining a context free grammar which represents the analyzed language (Fig. 1). EMILE can generate new sentences by means of the inducted grammar rules (Fig. 2). [1, 8, 15]

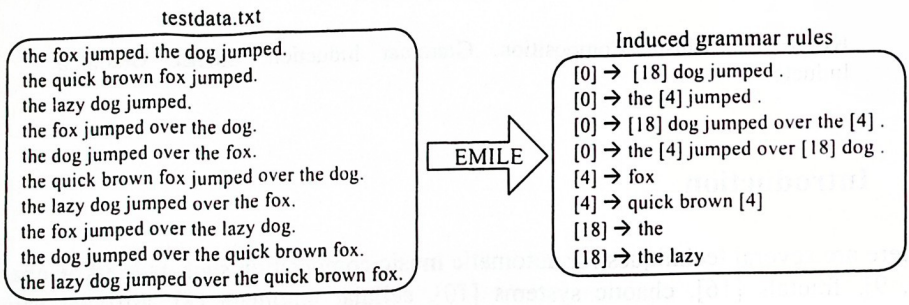


Fig. 1. EMILE grammar induction process [15]

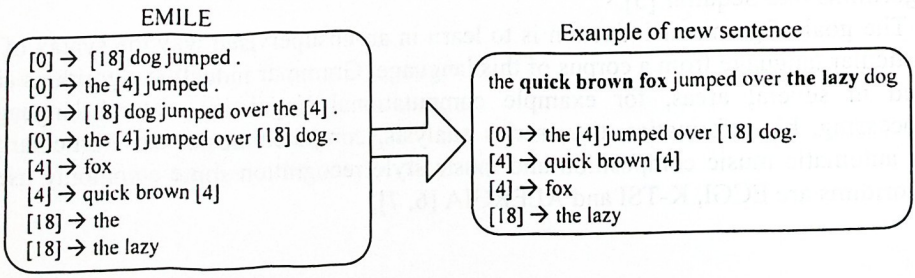


Fig. 2. New sentence generation by EMILE [15]

3 Music Composition using EMILE

The music composition procedure consists of three stages: Obtain the training corpus, Grammar induction and Music composition. We explain each of them below.

3.1 Training corpus

The process to obtain the training corpus for the EMILE grammar inductor is shown in Fig. 3. We explain each part below.

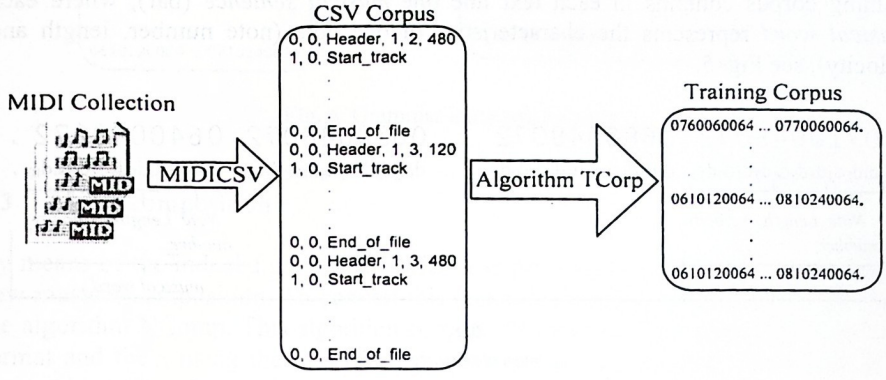


Fig. 3. Process to obtain the training corpus

The musical samples (MIDI Collection) for the Training corpus process are standard MIDI (Musical Instrument Digital Interface) files. We will focus in the events part of a Standard MIDI File [14]: Header, Time_signature, Note_on, Note_off, Note number, Velocity, Time clock, Start_track and End_track.

We convert each MIDI file to a text readable format using the CSVMIDI and MIDICSV programs[1]. This programs, transform a MIDI file to a CSV (Comma-Separated Values) text file and viceversa. CSV files are commonly used to store and transport tabular data between different applications. Fig. 4 shows the process to convert a MIDI file in CSV text format and viceversa.

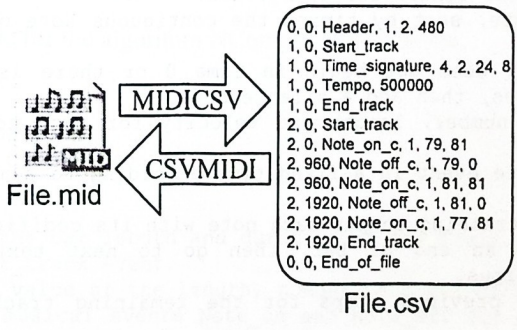


Fig. 4. MIDI file in CSV format.

[1] By John Walker, available from http://www.fourmilab.ch/webtools/midicsv/

We used the MIDICSV tool for obtain the CSV corpus by the MIDI Collection and then extracted the musical events (Header, Time_signature, Note_on, Note_off, Note number, Velocity, Time clock, Start_track and End_track) to create the training corpus for the grammar inductor.

Musical events express the length of each bar, the note number, length and velocity of each note into each bar. We represented each bar in the training corpus with one text line delimited with a dot (.), and each note number and their characteristics of length and velocity with a number of 10 digits, where the first three digits of left to right represent the note number, the next four represent the note length and the next three represent its velocity and each number is separated with one space. Thus, the training corpus contains in each text line one *musical sentence* (bar), where each *musical word* represents the characteristics of one note (note number, length and velocity), see Fig. 5.

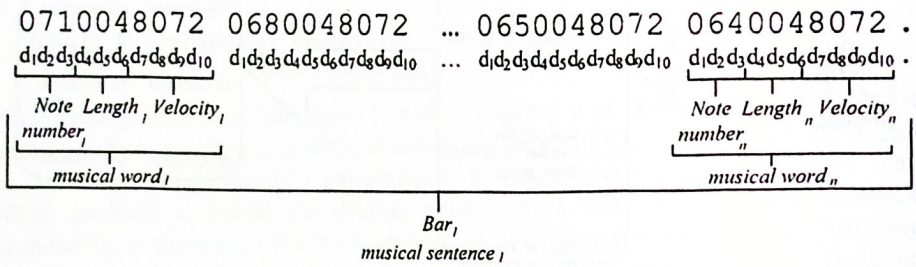


Fig. 5. Format for each text line into the training corpus

We implemented the algorithm TCORP to obtain the length of each bar, the characteristics of each note and the training corpus with the previous scheme for each text line. The pseudo-code for the algorithm is shown below:

```
Algorithm TCorp

Input: CSV Corpus
Output: Training Corpus

Start
  1. Obtain the bar length for each Track.
  2. For each note, sort by timing the contiguous Note_on and Note_off
     events.
  3 If the Track does not begin in Time 0 or there is a time space
     between notes, then add a silence note.
  4. Obtain the number, length and velocity for each note in the CSV
     corpus.
  5. End each note event in a fixed time according to a multiple of the
     bar length.
  6. Add to the training corpus each note with its codified values.
  7. If finding an end of bar, then go to next text line of the
     training corpus.
  8. Repeat the previous steps for the remaining tracks in the CSV
     Corpus.
End
```

3.2 Grammar induction

The input to EMILE is the Training Corpus obtained in the previous process. Once the grammar induction on the training corpus is done by EMILE, the result is a hierarchical structure of composition rules (Fig. 6).

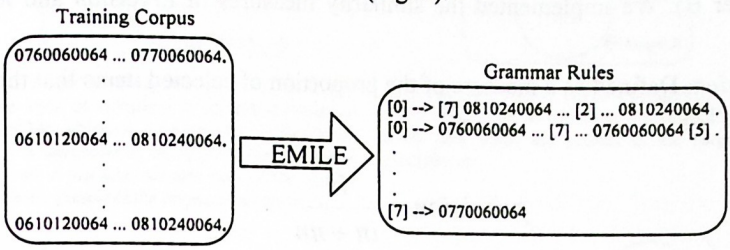


Fig. 6. Grammar induction process

3.3 Music composition

By means of the induced grammar, EMILE can generate new bars which will form a new musical composition. To convert this composition to the MIDI format, we used the algorithm NComp. This algorithm converts the new musical composition to CSV format and then, using the CSVMIDI program we obtain the MIDI file of the new composition (Fig. 7).

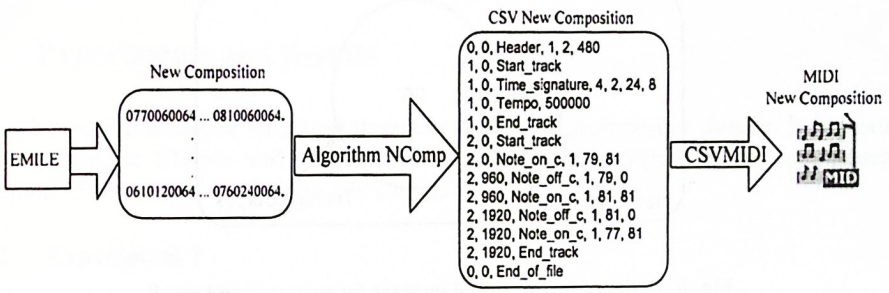


Fig. 7. Music composition process

The pseudo-code for the algorithm NComp is shown below:

```
Algorithm NComp

Input: New Composition
Output: CSV New Composition

Start
  1. Determine the bar length and duration of the Track.
  2. Print Start_track event.
  3. Obtain the value of the length, number, and velocity of each note.
  4. Write the musical events Note_on and Note_off for each note until
     completing the duration of the Track.
  5. Print End_track event.
  6. Print End_of_file event.
End
```

4 Similarity Analysis

The similarity analysis is based on a comparison of each new composition C_i obtained by a corpus A (Composer A) with the same corpus A and with other corpus B (composer B). We implemented the similarity measures of *Precision* and *Recall* as follows:

Precision. Defined as a measure of the proportion of selected items that the system got right [12]:

$$precision = \frac{in}{in + nn} \tag{1}$$

Recall. Defined as a proportion of the target items that the system selected [12]:

$$recall = \frac{in}{in + cn} \tag{2}$$

In the Fig. 8 we show the areas corresponding to new notes (*nn*), intersecting notes (*in*) and corpus notes (*cn*).

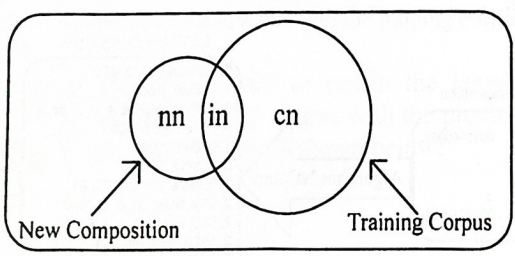
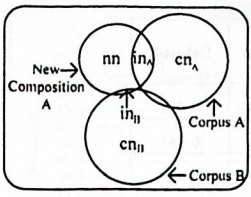
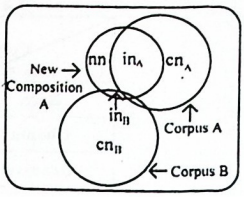


Fig. 8. Diagram for *nn*, *in* and *cn* areas for precision and recall

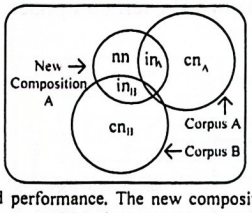
For a good performance of the grammar inductor for this application, we expected that a new composition obtained by a corpus A (Composer A) will have an *Precision A higher than Precision B*, and an *Recall A higher than Recall B*, when is compared with the corpus A and other corpus B (composer B), this may shown that the new composition is an new composition of the composer A. Fig. 9 show samples for a good and bad performance for the grammar inductor.



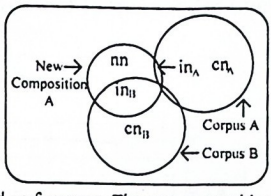
a) Good performance. The new composition A is in the style of composer A (corpus A) because contain bars which are not completely present in the corpus B (composer B) and is a new composition of composer A because contains bars which are not completely present in the corpus of composer A.



b) Bad performance. The new composition is not a new composition of composer A because contains many bars which are present in the corpus of composer A.



c) Bad performance. The new composition is a crossing composition because contains the same number of bars for a composer A and composer B.



d) Bad performance. The new composition A is a crossing composition because is in the style of composer B (corpus B) because contain bars which are not completely present in the corpus A (composer A).

Fig. 9. Good and bad performance for the grammar inductor

5 Experiments and Results

For the experimentation we chose three classical music composers: Johann Sebastian Bach, Frédéric Chopin and Claude Debussy. The MIDI corpus of each composer contains 50 monophonic Tracks. We carried out 2 experiments.

5.1 Experiment 1

1. Generate for each composer his corresponding training corpus.
2. Obtain for each composer his grammar rules (rules for composition).
3. Create 10 new compositions for each composer.
4. Carry out the similarity analysis between each new composition obtained and the corpus for each composer.
5. Repeat the previous steps for each training corpus of each classical music composer.

Table 1. Average Precision.

Corpus / Compositions	Bach	Chopin	Debussy
Bach	**91.476**	20.928	19.621
Chopin	21.682	**89.873**	44.822
Debussy	33.034	38.137	**93.977**

Table 2. Average Recall

Corpus / Compositions	Bach	Chopin	Debussy
Bach	**3.488**	0.059	0.057
Chopin	0.1	**0.177**	0.055
Debussy	0.179	0.11	**0.44**

5.2 Experiment 2

Consisted on carrying out experiment 1 varying the size of the training corpus using 20, 40, 60 and 80 percent of the original training corpus of each composer.

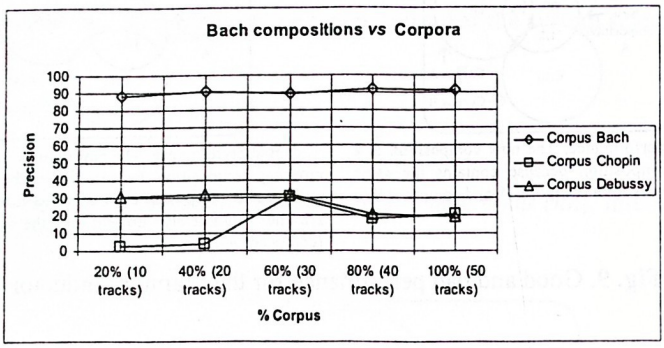


Fig. 10. Precision obtained by varying the size of the training corpus of Bach

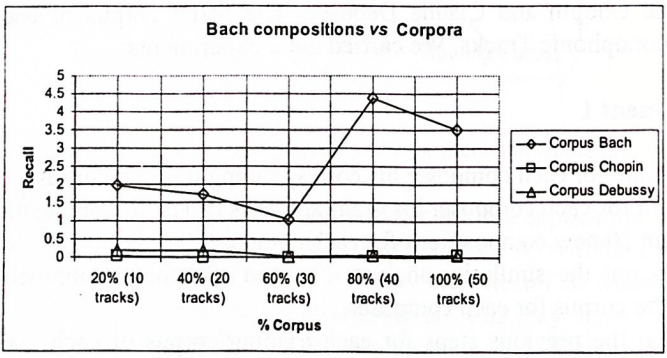


Fig. 11. Recall obtained by varying the size of the training corpus of Bach

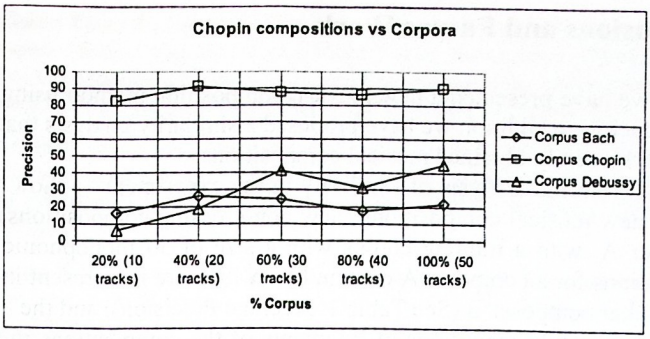


Fig. 12. Precision obtained by varying the size of the training corpus of Chopin.

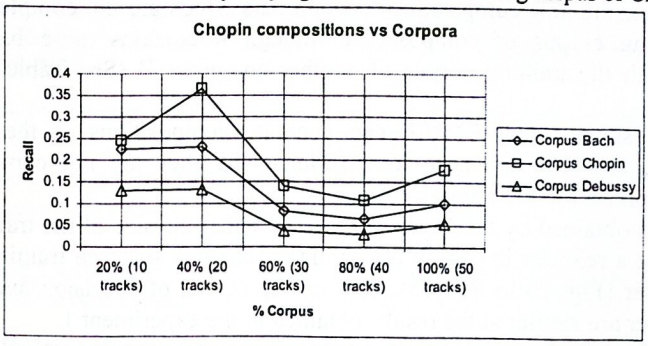


Fig. 13. Recall obtained by varying the size of the training corpus of Chopin

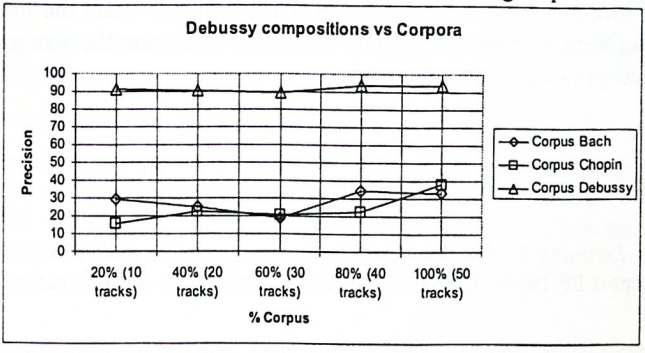


Fig. 14. Precision obtained by varying the size of the training corpus of Debussy

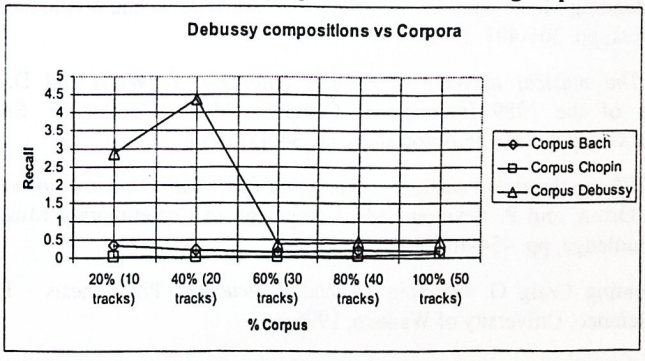


Fig. 15. Recall obtained by varying the size of the training corpus of Debussy

6 Conclusions and Future Work

In this work we have presented a new musical composition method using the EMILE Grammar Inductor; in addition we have proposed a similarity analysis that may define the quality of the newly obtained musical compositions.

The results from the analysis of similarity in the experiment 1, show that EMILE can generate new musical compositions (new non-crossing compositions) in the style of a composer A, with a training corpus with a size of 50 monophonic tracks. The new compositions for a composer A contain bars which are not present in the training corpus of another composer B (See Table 1. Average Precision), and the compositions obtained are not merely repetitions of fragments of the compositions in the learning corpus because the new compositions contain bars which are not completely present in the training corpus of composer A—though it contains more bars when is compared with the training corpus of another composer B (See Table 2. Average Recall).

This may suggest that EMILE can generate compositions in the style of a composer A, and they are new compositions why they are not copies of some composition present in the training corpus.

Our results obtained by the experiment 2, show that the size of the training corpus does not have a relevant impact in the similarity analysis for each training corpus of each composer (Fig. 10 to Fig. 15), because the results of Precision and Recall for each composer are similar at the results obtained in the experiment 1.

As a future work, we plan to compare this music composition method using other algorithms of Grammar Induction, different scheme for represent the notes and their characteristics (Note number, Length and Velocity), increment the size of the training corpus and incorporate polyphonic tracks.

References

[1] Adriaans P. *Learning Shallow Context-Free Languages under Simple Distributions*. ILLC Research Report PP-1999-13, Institute for Logic, Language and Computation. Netherlands, 1999

[2] Bel B., J. Kippen. *Bol Processor Grammars*. In M. Balaban, K. Ebcioglu, and O. Laske, eds. Understanding Music with AI, Cambridge , Massachusetts and Menlo Park, California: The MIT Press, pp. 366-401, 1992

[3] Beyls P. *The musical universe of cellular automata*. T. Wells and D. Butler, eds. Proceedings of the 1989 International Computer Music Conference. San Francisco: International Computer Music Association, pp. 34-41, 1989

[4] Chadabe J. *Flying through a musical space: about Real Time Composition*. In J. Painter, T. Howell, R. Orton, and P. Seymour, eds. Companion in Contemporary Musical Thought. London: Roudledge, pp. 454-465, 1992

[5] Nevill Manning Craig G. *Inferring Sequential Structure*. PhD Thesis - Philosophy in Computer Science, University of Waikato, 1996

[6] Cruz Alcázar Pedro P., Enrique Vidal Ruiz. *A Study of Grammatical Inference Algorithms in Automatic Music Composition and Musical Style Recognition*. Workshop on Automata Induction, Grammatical Inference, and Language Acquisition. The Fourteenth International Conference on Machine Learning (ICML-97), Nashville, Tennessee, USA, 1997

[7] Cruz Alcázar Pedro P., Enrique Vidal Ruiz. *Learning Regular Grammars to Model Musical Style: Comparing Different Coding Schemes*. Grammatical Inference, Lecture Notes in Artificial Intelligence, 4th International Colloquium, ICGI-98, Ames, Iowa, USA, pp. 211-222. July 1998

[8] Dörnenburg Eric. *Extension of the EMILE algorithm for inductive learning of context-free grammars for natural languages*. Master's Thesis, University of Dortmund, United Kingdom, 1997

[9] Frankel Goldwater Lee. *Computers Composing Music: An Artistic Utilization of Hidden Markov Models for Music Composition*. Department of Computer Science, University of Rochester, 2005.

[10] Gogins M. *Iterated Function Systems Music*. Computer Music Journal, 15 (1): 40-48, 1991.

[11] Griffth N., Peter M. Todd. *Musical Networks, Parallel Distributed Perception and Performance*. The MIT Press, England, pp. 227-260 1999.

[12] Manning Christhopher D., Shütze Hinrich. *Foundations of Statistical Natural Language Processing*. Second Printing. England, MIT Press, 2000

[13] Prusinkiewicz P. *Score generation with L-systems*. Proceedings of the International Computer Music Conference '86, pp. 455–457, 1986.

[14] Selfridge Field Eleanor. *Beyond MIDI, the Handbook of Musical Codes*. The MTI Press, Cambridge, Massachusetts, London, England, 1997

[15] Vervoorte M. *Emile 4.1.7 User Guide*. University of Amsterdam, 2004

[16] Waschka R., A. Kurepa. *Using fractals in timbre construction: an exploratory study*. In T. Wells and D. Butler, eds. Proceedings of the 1989 International Computer Music Conference. San Francisco: International Computer Music Association, pp 159-161, 1989