Proceedings of the 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION, Corfu, Greece, August 17-19, 2005 (pp325-330)

Evolving Computer-Generated Music By Means of the Normalized Compression Distance

MANUEL ALFONSECA, MANUEL CEBRIÁN and ALFONSO ORTEGA
Escuela Politécnica Superior
Universidad Autónoma de Madrid
SPAIN
{ Manuel.Alfonseca, Manuel.Cebrian, Alfonso.Ortega}@uam.es

Abstract: - This paper proposes the use of the well-known normalized compression distance as a fitness function which may be used by genetic algorithms to automatically generate music in a given pre-defined style. The superiority of the relative pitch envelope over other musical parameters, such as the lengths of the notes, has been confirmed, bringing us to develop a simplified algorithm that nevertheless obtains interesting results.

Keywords: - Evolutionary Computation, Coding and Information Theory, Genetic Algorithms, Computer Generated Music, Classification, Clustering

Acknowledgement: - This work has been sponsored by the Spanish Ministry of Science and Technology (MCYT), project number TIC2002-01948.

1. Introduction

The automatic generation of musical compositions is a long standing, multi disciplinary area of interest and research in computer science, with over thirty years history at its back.

Some of the current approaches try to simulate how the musicians play [1] or improvise [2], while others do not deal with the time spent in the process. Many of them apply models and procedures of Theoretical Computer Science (cellular automata [3], parallel derivation grammars [1], or evolutionary programming [4-7]) to the generation of complex compositions. The models are then assigned a musical meaning. In some cases, the music may be automatically found (composed) by means of genetic programming.

Our group is interested in the simulation of complex systems by means of formal models, their equivalence and their design, not only by hand, but also by means of automatic processes, such as genetic programming.

2. A brief introduction to musical parameters

In the following paragraphs, some essential musical theory concepts will be reviewed.

Melody, rhythm and harmony are considered the three fundamental elements in music.

Melody is a series of musical silences (rests) and sounds (or notes) with different lengths and stresses, arranged in succession in a particular rhythmic pattern, to form a recognizable unit.

Notes' names belong to the set {A, B, C, D, E, F, G}. These letters represent musical pitches and correspond to the white keys on the piano. The black keys on the piano are considered as modifications of the white key notes, and are called sharp or flat notes. From left to right, the key that follows a white key is its sharp key, while the previous key is its flat key. To indicate a modification, a symbol is added to the white key name (as in A# or A+ to represent A sharp, or in B♭ or B-, which represent B flat). The interested reader can find an amusing simulation of a virtual keyboard at [8].

The distance from a note to its flat or sharp notes is called "a half step" and is the smallest unit of pitch used in the piano, where every pair of two adjacent keys are separated by a half step, no matter their color. Two consecutive half steps are called a whole step.

Notes and rests have a length (a duration in time). There are seven different standard lengths (from 1, corresponding to a *whole* or *round note*, to 1/64), each of which has double duration than the next. Their names are: whole, half, quarter, quaver, semi-quaver, quarter-quaver and half quarter-quaver. The complete specification of notes and silences includes their lengths.

An interval may be defined as the number of half steps between two notes.

3. The normalized compression distance

The search for a universal metric has been one for a long time of the main objectives of clustering theory. The availability of such a metric would make it possible to apply the same algorithms to widely different clustering problems: to name a few, the classification of music, texts or gene sequences.

In particular, genetic algorithms need to define *fitness functions* that make it possible to compare many different individuals, those subject to simulated evolution, and classify them according to their degree of adaptation to the environment.

In many cases, these fitness functions have to compute the distance of each individual (or one of its properties) to a desired goal. Let's suppose that we want to generate a composition that resembles a Mozart symphony; in this situation we can elaborate a natural fitness measure: an individual (representing a composition) has a high fitness if it shares many features with one (or more) of the Mozart's symphonies. The problem now is how to select those features and their respective metrics. Surprisingly, there exists a universal similarity metric that summarizes all the possible features: the *normalized information distance* [9]. It is universal in the following sense: if any metric measures a small distance between two objects, then the normalized information distance also measures a small distance between the same objects; thus, it minorizes every computable metric. The normalized information distance is mathematically defined as follows:

$$d(x,y) = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}}$$

where $K(x|y)$ is the conditional Kolmogorov complexity of the string x given the string y, whose value is the length of the shortest program (for some universal machine) which, when run on input y, outputs the string x. $K(x)$ is the degenerate case $K(x|\lambda)$, where λ is the empty string; see [10] for a detailed exposition of the appropriate algorithmic information theory. Unfortunately, both the conditional and the unconditional complexities happen to be incomputable functions.

In [11] a computable estimate of the normalized information distance is presented

$$\hat{d}(x,y) = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}}$$

where $C(x)$ is the length resulting of compressing x with compressor C, and xy is the concatenation of x and y. Li and Sleep have reported that this metric, together with a nearest neighbour classifier, outperforms some of the finest (and much more complex) algorithms for clustering music by genre [12]. Some earlier researchers have also reported a great success in clustering tasks with the same metric [13]. These results suggest that the normalized compression distance, although not achieving the universality of its incomputable predecessor, works finely to extracting shared features between two musical pieces.

Putting it all together, we present the following scheme for the automatic generation of music:

1. Select one or more musical pieces as the guides for music generation.
   $\Omega = \{\omega_i\}_1^N$

All the ω_i must be coded in the same way.

2. Code the population's individuals with the same coding as the guides.
3. Use the following fitness function:

$$f(x) = \frac{1}{\sum_i \hat{d}\,(x, \omega_i)}$$

We expect that, by maximizing $f(x)$, we will maximize the number of features shared by the evolving individuals with the guide set. For example, if Ω were the set of Mozart's symphonies, an individual with a high fitness should resemble (when played) a Mozart symphony.

It remains to choose the compressor used to estimate $\hat{d}$. Li and Sleep compute $C(\alpha)$ by counting the number of blocks generated by the LZ78 parsing algorithm [14] for an input α. In our initial experiments, we used both the LZ78 and LZ77 algorithms, and found that LZ77 performs better, which agrees with [15]; therefore, LZ77 has been used as our reference compressor in all the experimental results presented in this paper.

4. A genetic algorithm that generates music

A piece of music can be represented in several different, but equivalent ways:

- With the traditional Western bi-dimensional graphic notation on a pentagram.

- With a set of strings interpreted in the following way: the notes are represented by letters (A-G), silence by a P, sharp and flat alterations by + and – signs, and the lengths of notes by a number (i.e. 0 would represent a whole note, 1 a half note, and so on). Intermediate lengths can be indicated by inserting a period. Additional codes can define the tempo and the octave, whether the notes must be executed in a normal way (or legato or staccato), and so forth. Polyphonic music may be represented by means of parallel strings.

- By numbering consecutively the pitches of all the notes in the piano keyboard, and representing each by its number (1 to 88). The length of each note can be represented by a number indicating a multiple of the minimum unit of time or a sub-multiple of the round note. A piece of music can be represented by a series of integer pairs. The first number in each pair is the note, the second its length. Note 0 can represent a silence. Polyphonic music may be represented by means of parallel sets of integer pairs.

- Other coding systems are used to keep and reproduce music in a computer, such as MIDI, MP3, et cetera.

In our experiments, we decided to represent music by means of the second and third notation systems indicated above. In fact, the genetic algorithm generates music coded as pairs of integers, a format specially fitted for that purpose. This notation can then be transformed to a note string for reproduction purposes. We also decided to start with monophonic music, leaving harmony for a later phase and working only with melodies. Finally, we took the decision to apply the genetic algorithm only to the relative pitches of the notes in the melody (i.e. we only consider the relative pitch envelope), ignoring the absolute pitches and the note lengths, because our own studies and other's [12] suggest that a given piece of music remains recognizable when the lengths of its notes are replaced by random lengths, while the opposite doesn't happen (the piece becomes completely unrecognizable if its notes are replaced by a random set, while maintaining their lengths).

The scheme for the genetic algorithm is as follows:

1. Provide one or more pre-written pieces of music, which will be used as the guide set for the algorithm. These pieces will be expressed as N pairs of integers, as described above. Provide also a goal distance to the guide piece of music (a minimum distance that, on being reached, will make the goal fulfilled).

2. Generate a random population of 64 vectors of N pairs of integers. The first integer in each

pair is in the [24,48] interval, the second in the [1,16] interval. Each vector represents a genotype.

3. Compute the fitness of every genotype as the distance to the guide set, measured by means of the normalized compression distance (algorithm LZ77).

4. Order the 64 genotypes by increasing distance to the guide set.

5. If the lowest distance is less or equal to the goal distance, stop and return the notes in the corresponding genotype, paired with the lengths in the guide piece(s) of music.

6. From the ordered list of 64 genotypes created in step 5, remove the 16 genotypes with least fitness/highest distance (leaving 48) and take the 16 genotypes with most fitness/lowest distance. Pair these 16 genotypes randomly to make 8 pairs. Each pair generates another pair, a copy of their parents, modified according to four genetic operations. The new 16 genotypes are added to the remaining population of 48 to make again 64, and their fitness is computed as in step 3.

7. Go to step 4.

The four genetic operations mentioned in the algorithm are:

- Recombination (applied to 100% generated genotypes). Given a pair of genotypes, (x_1, x_2 ... x_n) and (y_1, y_2 ... y_m), a random integer is generated in the interval [0, mín(n,m)]. Let it be *i*. The resulting recombined genotypes are: (x_1, x_2 ... x_{i-1}, y_i, y_{i+1} ... y_m) and (y_1, y_2 ... y_{i-1}, x_i, x_{i+1} ... x_n).

- Mutation (one mutation was applied to every generated genotype, although this rate may be modified in different experiments). It consists of replacing a random element of the vector by a random integer in the same interval.

- Fusion (applied to a certain percentage of the generated genotypes, which in our experiments was varied between 5 and 10). The genotype is replaced by a catenation of itself with a piece randomly broken from either itself or its brother's genotype.

- Elision (applied to a certain percentage of the generated genotypes, in our experiments between 2 and 5). One integer in the vector (in a random position) is eliminated.

The last two operations allow longer or shorter genotypes to be obtained from the original N element vectors.

In our first experiment, we used as the guide a single piece of music: Chopin's seventh prelude, represented by the following string:

M2T2O3L2EO4C+3.D4O3M1BBB1M2O4F+D+
3.E4M1AAA1M2C+O3A+3.B4O4M1DDD1O3G
+M2G+3.A4O4M1C+C+C+1O3M2EO4C+3.D4
O3M1BBB1M2O4F+D+3.E4O5M1C+C+C+1O4
C+M2C+3.D4M1F+F+F+1M2O3G+B3.A4O4M1
AAA1

After applying the genetic algorithm, we completed the succession of notes obtained with length information in the following way: each note was assigned the length of the note in the same position in the guide piece (the guide piece was circularly extended, if needed, to make it the same length as the generated piece). In successive executions of the algorithm, we obtained different melodies at different distances from the guide. It was observed that a lower distance made the generated music more recognizable to the ear, as related to Chopin's style. For instance, the distance to the guide of the following generated piece is 0.39:

T5O4C1O3E2.C+3A+1A+1A+0O2A+1O3C+2.O
2B3O3B1B1B0D+1D+2.E3G+1O4C1O3E0C+1D
2.F3F1F1O2B0C+1A+2.B3G+1G+1G+0O3G+1
G+2.G+3O2B1O3E1E0E1O2B2.O3F+3D+1E1A
0A1A2.

The number of generations needed to reach a given distance to the guide depends on the guide length and on the random seed used in each experiment, and follows an approximate Poisson curve, as shown in figure 1, that represents the result of one experiment.

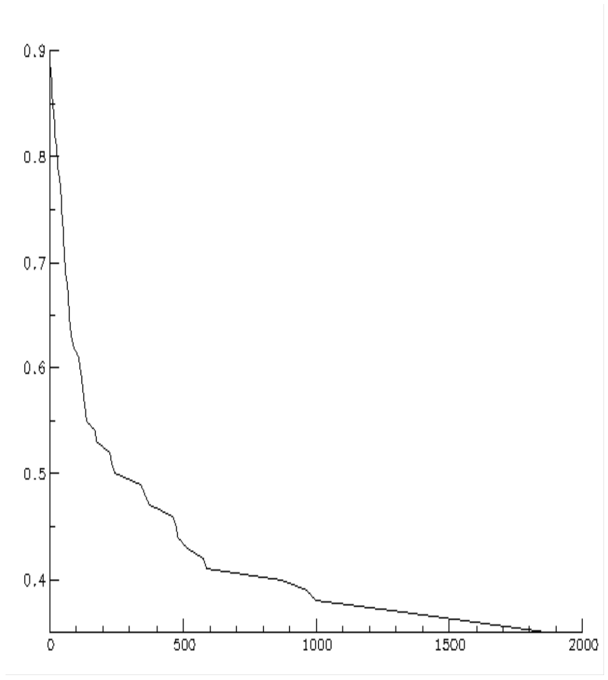

Figure 1. Number of generations needed to reach a given distance to the target.

In our second experiment, we used two simultaneous guide pieces: *Beguin the Beguine*, and *My heart belongs to daddy*, both by Cole Porter, and minimized the sum of the distances of the evolving individuals to the two guide pieces.

The following represents one of the results we obtained, which happens to be at a distance of 0.67 from the first guide piece, and 0.72 from the second, while the normalized compression distance between both guide pieces is 0.81, i.e. the generated piece resulted to be nearer both of the guides than they are among themselves:

```
T5O3C+3.D3.O2A3.O3F+1.F+3.D+3.O2G+3.O3
C+1O2G+3.C+3.D+3.D3.F1D+3.C+3.C3.C3.C+3
.D+1O3C3.D3.F3.D1F3.E3.O2G+3.O3D+2C2O2
A+1O3C3.O2A+3.A+3.A+1A+1G+3.E1.F+3.O4
C3.O3F3.G1.F+3.C+3.D+3.E1G+3.E3.E3.O2C3.
D+1C+3.D+3.O3C3.C3.G+3.C+1D+2E2F+3.E1.
O2B3.O3G+3.O2C3.C+3.C+1C+3.F3.G3.G1F1D
+3.O3C1C3.O2A3.D3.A+3.O3C1D3.O2F+3.F+3.
D3.G3.F1O3D3.E2D+2E1.D+3.G+3.O2D+3.D+1
G3.A3.G+3.O3C3.O2A1A3.E3.F+3.G3.B3.G1D+
3.D+3.F3.A+3.B1O3D+3.C+3.F+3.F+1.E3.D+3.
D+3.C2O2B1O3D+3.C3.O2B2O3E2O2A1A2G+
3.G+3.E2.F+3.F3.D+2F1D3.D+3.O3F2D+3.F+3.
D1O2A2G+3.O3C+3.G+2F2C+2O2A2F1O3F+3.
B2.O2F+3.E3.G3.F+1E3.E3.D+3.C+3.O3C+1.O2
G+1C+3.C+1D+3.F3.A+2G+2G+3.F+1D+2E3D0
D+2F3.D+3O3G3.D2B2O2D+2O3C3C3C3.C2O
2B0A3.A3A3.B2.O3C3.F+1G3.A+3.G+3F+3A3.
F1.C2O2G+3.G+3.O3G+0A+3.B3.B0B3.O2E3A
+3B3.O3D3.D3.O2A+1O3D+3F+3F0F+3.D+3F3
G3.G3.G3.G+2A+3O4C3O3A3A+2G+0O2F+3G
+3F+3.F3.F+3.O3G+2F+3A3A+3G+3.F+1D+3.C
+3.C3.O2A+3.A+0A+3.O3C3.O2A+3.O3C3.C+0
D3.C3.O2C3.
```

To obtain the preceding piece, we completed the succession of notes generated by the genetic algorithm with the required length information, in the following way: each note was assigned the average lengths of the two notes in the same position in the two guide pieces (the guide pieces were shortened or circularly extended to make them the same length as the generated piece). This approach happened to provide a much more esthetically appealing result than the one obtained using the length of only one of the guide pieces.

Finally, in the last experiment presented here, two pieces by Mozart were used as simultaneous guides: a few bars of the first movement in Symphony 40, and a part of the second movement in sonata KV545. The result, which appears tantalizingly Mozartian to the ear in some parts, happens to be at distances of 0.65 and 0.58 from the two guide pieces, which on the other hand differ from one another by a distance of 0.90. The length of the notes was generated in the same way as in the preceding experiment.

```
T5O4G+0F+3B3.A+3A3G+2.G+3B1F1G+1.F3.D
+3D+3D3.C3O3A+3O4F2.G+3F1C+1F+2.D+3.F
+2E2D2C+2O3F2.F+3.G+1O4C1O3A3.F3.O4F3.
G3F3F3.G3E3G3.B3O3E3G3.F2.G+3G4G+2A2
O4G2G+2G1G+3.F3.G+3.O5C3O4A+3G+3.A+3
.A+2.G+3.G3.G3.E3D+3O3F+3F3F+3.G3.G+3.A
3.A3.G3F+3F+3.D+3.D3.O4D3.C+2C+3E2O3A+
2O4C+3C2O3B2G1B1O4D3.C3.O3G3.O4D+3G
3D3.D+3D3D+3.D3E3F3.G3.F3.E3F3G1O3D+3
E3D+3.D+3B3B3.A+3.O4F3.G2.G+3A+3.G3
```

In comparison with this, the piece obtained by evolution towards the two works by Cole Porter has a distinctly lighter sound.

4. Conclusions and future work

We have found that the normalized compression distance is a promising tool to provide genetic algorithms for automatic music generation with a measure of the distance to the desired target, which may be used as an appropriate fitness function. Some of the generated pieces of music have a significant similarity to the style of well-known authors, in spite of the fact that our fitness function ignores the duration of the notes and takes into account only the relative pitch envelope. Our results have been much better than those we obtained with a different procedure and fitness function in [16].

In the future we intend to combine our results with those of other authors [12-13], so as to use as the target for the genetic algorithm, not just one or two pieces of music by a given author, but a cluster of pieces by the same author, in this way trying to capture the style in a more general way. We shall also use the information about note duration in the algorithm.

We shall also try to work with a more standard and richer system of music representation, such as MIDI or MP3.

References:

[1] J. McCormack, 1996 Grammar-based music composition. *Complex International,* Vol 3.

[2] J. Biles, 1994 GenJam: A Genetic Algorithm for Generating Jazz Solos, *Proceedings of the 1994 International Computer Music Conference*, ICMA, pp. 131-137, San Francisco, 1994.

[3] E. Bilotta, P. Pantano, V. Talarico, 2000 Synthetic Harmonies: an approach to musical semiosis by means of cellular automata, *Leonardo*, MIT Press, vol. 35:2, pp. 153-159, April 2002.

[4] D. Lidov, J. Gabura, 1973 A melody writing algorithm using a formal language model, *Computer Studies in the Humanities* Vol. 4:3-4, pp. 138-148, 1973.

[5] P. Laine, M. Kuuskankare, Genetic Algorithms in Musical Style oriented Generation, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp 858-862, Orlando, Florida, vol. 2, 1994.

[6] D. Horowitz, Generating Rhythms with Genetic Algorithms, *Proceedings of the ICMC 1994*, pp. 142-143, International Computer Music Association, Århus, 1994.

[7] B. Jacob, Composing with Genetic Algorithms, *Proceedings of the 1995 International Computer Music Conference*, pp. 452-455, ICMC, Banff Canada, 1995.

[8] M. Moncur: The www virtual keyboard, http://www.xmission.com/~mgm/misc/keyboard.html.

[9] M. Li, X. Chen, X. Li, B. Ma and P. Vitányi, The similarity metric, *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 863-872.

[10] P. and M. Li, An Introduction to Kolmogorov Complexity and its Applications, *Springer-Verlag*, 1993.

[11] R. Cilibrasi and P. Vitanyi, Clustering by Compression, *IEEE Trans. Information Theory*, Vol.51 No.4, 2005, pp. 1523-1545.

[12] M. Li and R. Sleep, Melody Classification using a Similarity Metric based on Kolmogorov Complexity, *Sound and Music Computing*, 2004.

[13] R. Cilibrasi and P. Vitanyi, Algorithmic Clustering of Music, *Proc. Of the Fourth Intl. Conf. on Web Delivering of Music (WEDELMUSIC'04),* pp. 49-67, IEEE Computer Society, ISBN: 0.7695-2157-6, 2004.

[14] J. Ziv and A. Lempel, A universal algorithm for sequential data compression, *IEEE Transactions on Information Theory*, Vol.23:3, pp. 337-343, 1997.

[15] S. R. Kosaraju and G. Manzini, Some entropic bounds for Lempel-Ziv algorithms, *Data Compression Conference*, pp. 446, 1997.

[16] A.Ortega, R.Sánchez Alfonso, M.Alfonseca: Automatic Composition of Music by means of Grammatical Evolution, APL Quote Quad (ACM SIGAPL), Vol. 32:4, p. 148-155, Jun. 2002.