# A fitness function for computer-generated music using genetic algorithms

**Article** *in* WSEAS Transactions on Information Science and Applications · March 2006

CITATIONS
13

READS
409

**3 authors**, including:

Manuel Alfonseca
Universidad Autónoma de Madrid
**184** PUBLICATIONS **1,264** CITATIONS
SEE PROFILE

Alfonso Ortega De la Puente
Universidad Autónoma de Madrid
**73** PUBLICATIONS **638** CITATIONS
SEE PROFILE

# A Fitness Function for Computer-Generated Music using Genetic Algorithms

MANUEL ALFONSECA, MANUEL CEBRIÁN and ALFONSO ORTEGA
Escuela Politécnica Superior
Tomás y Valiente, 11
Universidad Autónoma de Madrid
28049 Madrid SPAIN
{Manuel.Alfonseca, Manuel.Cebrian, Alfonso.Ortega}@uam.es

*Abstract:* - This paper describes the use of genetic algorithms for the automatic generation of music, by means of a fitness function computed with the normalized compression distance. Different recombination procedures are tested. The computer music generated tries to recover the style of a selected human author. In spite of the simplicity of the algorithm, the procedure 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 [1-7].

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 [8-9].

This paper is organized thus: the second section provides a short introduction to musical concepts needed to better understand the remainder, with a description of the restrictions applied in our experiments and an enumeration of different ways of representing music. The third section explains the normalized compression distance, which has been used to compute the distance of the results of the genetic algorithm from the target musical pieces. The fourth section describes the genetic algorithm we have used for music generation. In the fifth and sixth sections we describe our experiments, where we have compared the use of one or two target guides, and four different recombination procedures for the genetic algorithm. Finally, the last section presents our conclusions and possibilities for future work.

## 2. Musical representation: restrictions

Melody, rhythm and harmony are considered the three fundamental elements in music. In the experiments performed in this paper, we shall restrict ourselves to melody, leaving the management of rhythm and harmony as future objectives. In this way, we can forget about different instruments (parts and voices) and focus on monophonic music: a single performer executing, at most, a single note on a piano at a given point in time. Melody consists of a series of musical sounds (notes) or silences (rests) with different lengths and stresses, arranged in succession in a particular rhythmic pattern, to form a recognizable unit.

In Western music, the names of the notes 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 [10]. 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. Instruments different from the piano may generate additional notes; in fact, flat and sharp notes may not coincide; also, in different musical traditions (such as Arab or Hindu music) additional notes exist. However, in these experiments, we shall restrict to the Western piano lay-up, thus simplifying the problem to just 88 different notes separated by half steps. An interval may be defined as the number of half steps between two notes.

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 duration double than the next. Their names are: whole, half, quarter, quaver, semi-quaver, quarter-quaver and half quarter-quaver. Intermediate durations can be obtained by means of dots or periods. The complete specification of notes and silences includes their lengths.

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

- With the traditional Western bi-dimensional graphic notation on a pentagram.
- By a set of character strings: 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 (0 would represent a whole note, 1 a half note, and so on). Adding a period provides intermediate lengths. Additional codes define the tempo, the octave and the performance style (normal, legato or staccato). Polyphonic music is represented with sets of parallel strings.
- By numbering (1 to 88) the pitches of the notes in the piano keyboard. Another number can represent the length of the note as a multiple of the minimum unit of time. A voice in a piece of music would be a series of integer pairs representing notes and lengths. Note 0 would 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 or a recording system, with or without compression, such as wave sampling, MIDI, MP3, etc.

In our experiments, we represent melodies by the second and third notation systems.

## 3. The normalized compression distance

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

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

In many cases, fitness functions compute the distance from 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 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* [11]. It is universal in the sense that, when any metric measures a small distance between two given objects, the normalized information distance also measures a small distance between the same two objects; thus, it is at least as good as any other 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 string $x$ given string $y$, whose value would be the length of the shortest program (for some universal machine) which, when run on input string $y$, outputs string $x$. $K(x)$ is the degenerate case $K(x|\lambda)$, where $\lambda$ is the empty string; see [12] for a detailed exposition of the appropriate algorithmic information theory. Unfortunately, both the conditional and the unconditional complexities happen to be incomputable functions.

In [13] 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 obtained by 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 or a cladistic classifier, outperforms some of the finest (more complex) algorithms for clustering music by genre [14]. Earlier researchers have also reported a great success in clustering tasks with the same metric [15]. These results suggest that the normalized compression distance, although not achieving the universality of its incomputable predecessor, works well to extract features shared between two musical pieces.

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 compression algorithm [16] for an input $\alpha$. In our initial experiments, we used both the LZ78 and LZ77 algorithms, and found that LZ77 performs better, which agrees with [17]; therefore, LZ77 has been used as our reference compressor in all the experimental results presented in this paper.

## 4. The genetic algorithm used to generate music

Our genetic algorithm generates music coded as pairs of integers, the third format described in section 2, which is specially fitted for our purpose. This notation can then be transformed to a note string (the second format) for reproduction. We also decided, in this first set of experiments, 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 [14] 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 proposed genetic algorithm scheme is now described. It includes a previous pre-process step, made of the following parts:

- One or more musical pieces are selected as targets or guides for music generation.

  $$\Omega = \{\omega_i\}_1^N$$

  All the $\omega_i$ must be coded in the same way, as pairs of integers, as described above.
- The individuals in the population are coded in the same way as the guides.
- The following fitness function is used:

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

  where $\hat{d}(x,y)$ was defined in section 3. By maximizing $f(x)$ (minimizing the sum of the distances), we expect to maximize the number of features shared by the evolving individuals with the guide set. For example, if $\Omega$ were the set of Mozart's symphonies, an individual with a high fitness should resemble (when played) a Mozart symphony.

The remaining steps of the genetic algorithm are:

1. The program generates a random population of 64 vectors of N pairs of integers, where N is the length of the first piece of music in the guide set. The first integer in each pair is in the [24,48] interval, the second in the [1,16] interval. Each vector represents a genotype.
2. The fitness of the genotypes is computed as the distance to the guide set, measured by means of the normalized compression distance.

3. The genotypes are ordered by their increasing distance to the guide set.
4. If the goal distance has been reached, the program stops. The notes in the target genotype are paired with a function of the lengths of the guide piece(s).
5. The 16 genotypes with least fitness are removed. The 16 genotypes with most fitness are paired randomly. Each pair generates a pair of children, a copy of the parents modified by four genetic operations. The children are added to the population to make again 64, and their fitness is computed as in step 2.
6. Go to step 3.

The four genetic operations mentioned in the algorithm are:

- Recombination (applied to 100% generated genotypes). The genotypes of both parents are combined using different procedures to generate the genotypes of the progeny. Different recombination procedures have been tested in this set of experiments to find the best combination.
- 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, together with some recombination procedures, allow longer or shorter genotypes to be obtained from the original N element vectors.

## 5. Testing different numbers of guide pieces

In our first experiments, we selected the simplest recombination procedure and tested the effect of varying the number of guide pieces and the functions that generate the lengths of the notes in the best output pieces. First, we used as the guide a single piece of music, *Yankee Doodle*, represented by the following string:

M2T2O3L2C+4C+4D+4F4C+4F4D+4O2G+4O3
C+4C+4D+4F4C+3C4P4C+4C+4D+4F4F+4F4D
+4C+4C4O2G+4A+4O3C4C+3C+4P4O2A+4.O3
C5.O2A+4G+4A+4O3C4C+4P4O2G+4.A+5.G+4
F+4F3G+4P4A+4.O3C5.O2A+4G+4A+4O3C4C
+4O2A+4G+4O3C+4C4D+4C+3C+3

In this case, the fitness function was straightforward: the objective was to minimize the normalized compression distance of the vectors of note intervals of the evolving individuals to the corresponding vector in the guide piece. After applying the genetic algorithm, the succession of notes obtained was completed by adding 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 shortened or circularly extended, if needed, to make it the same length as the generated piece, which could be shorter or longer). 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 the guide piece. For instance, the distance to the guide of the following generated piece is 0.43:

T5O3D+2O2G+2O3C+2C+2D+2F2F+2F2E2C2
D2E2O2F1D2E2D2C2D2E2F+2G2G2A2B2O3C
2O2B2O3D2E1O2F2D+2F2.G3.G+2F2D+2G+2F
+2E2F+2.F3.C+2C2D+1C+2C+2A+2.O3C3.C+2
O2G+2A+2G+2F+2O3D+2B2O3D+2C+2

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

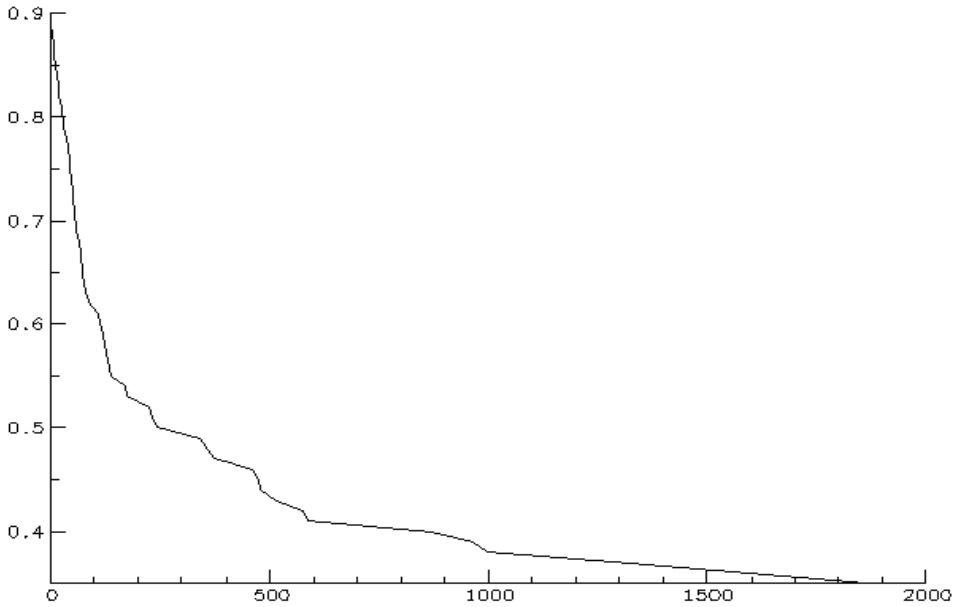


Figure 1. Number of generations needed to reach a given distance to the target.

In our second experiment, we used two simultaneous guide pieces: *Beguin the Beguine*, and *My heart belongs to daddy*, both by Cole Porter. In this case, the fitness function to be minimized was the sum of the normalized compression distances of the note intervals of the evolving individuals to the note intervals of 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 was nearer to both 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 happens to provide a more esthetically appealing result than the one obtained when the length of only one of the guide pieces is used.

In our third experiment, Chopin preludes numbers 4 and 7 were used as simultaneous guides. The result came to be at distances of 0.61 and 0.74 from the two guide pieces, which are separated from one another by a distance of 0.96. The length of the notes was generated in the same way as in the preceding experiment. Compared with this, the

piece obtained using as guides two works by Cole Porter has a distinctly lighter sound.

T5O3G+2.O2A+2O3G1.O2A+1O3G0O3F+1.O3C0O2B2.O3D+1.O3F+1O2F+0O2F+1.O2G0O2F+2.O2F1.O2E2.O2E2O2E0O2B1.O3C2O3D+3.O3D+2.O3D+2.O3D1O3C+2.O2A+1.O2A2O2G+0O2G+2O2A1O3C2.O3E2O3G3.O2B2.O3D2.O3C1O4C2.O4C2O2C3O2D0O2F1O2D+0O2A1O3F+1.O3G2O3E2.O2F+2O2B1.O2B2O2B3.O2D+4O2G+2O2F1O2G+1O2F2O2F+2O2A+3.O2A+2.O2A+2.O2C+1O2A+2.O2A+2O2A3.O2A+2.O3C+2.O3F1O2B2O2B2O3C+2.O2B2.O3B0O3B1O2B2.O3F+1.O3G2O3B2O2B0O3C+1.O2B0O3C+2.

## 6. Testing different recombination procedures

In our next set of experiments, we tested the effect of changing the recombination procedure used by the genetic algorithm. The following strategies were used:

- Strategy 1: 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 ($y_1$, $y_2$ ... $y_{i-1}$, $x_i$, $x_{i+1}$ ... $x_n$). This is the base case (the simplest recombination strategy) which was used in all the experiments described in the preceding section.

- Strategy 2: given a pair of genotypes, ($x_1$, $x_2$ ... $x_n$) and ($y_1$, $y_2$ ... $y_m$), two random integers are generated in the interval [0, n] (let us call them i, j, i<j) and another two in the interval [0,m] (let us call them p, q, p<q). The resulting recombined genotypes are: ($x_1$, $x_2$ ... $x_{i-1}$, $y_p$, $y_{p+1}$ ... $y_{q-1}$, $x_j$, $x_{j+1}$ … $x_n$) and ($y_1$, $y_2$ ... $y_{p-1}$, $x_i$, $x_{i+1}$ ... $x_{j-1}$, $y_q$, $y_{q+1}$ ... $y_m$).

- Strategy 3: given a pair of genotypes, ($x_1$, $x_2$ ... $x_n$) and ($y_1$, $y_2$ ... $y_m$), four random ordered integers are generated in the interval [0, n], [0, m] for each parent genotype. Each genotype is then cut into the five corresponding pieces, which are shuffled together (one of them is reversed). The genotypes of the progeny are obtained by concatenating five of the pieces in the shuffled set.

- Strategy 4: similar to the preceding one, but only three random ordered integers are used to divide the parent genotypes into four pieces, which are then joined, shuffled, and used (four at a time) to generate the genotypes of the progeny.

The one-point crossing-over strategy 1 has the property that the lengths of the parent genomes are invariant under recombination in the progeny. Since mutation also keeps the length of the genome, only fusion and elision change it. In fact, we did notice that, in our preceding experiment, fusion almost never leads to a fitter genome, while elision sometimes does, which means that the version of our genetic algorithm described in the previous section, which starts with a genome length copied from one of the target pieces of music, leads to genome lengths usually reduced by a little (not much) from their initial value. Strategies 2, 3 and 4, however, all lead to progeny genomes with lengths usually quite different from those of their parents (even when both parent genomes had the same length), which provides the population with a much larger genome length variety than strategy 1.

After performing several experiments we noticed that, at the beginning of the evolution, the second recombination strategy converges more quickly towards the target, but after a certain number of generations (usually between 150 and 200), the first and fourth strategies becomes better, while beyond about 500 generations after the beginning of the process the first strategy is clearly the best. Above 1000 generations, the first two strategies tend to converge, i.e. to obtain similar distances to the goal after the same number of generations.

This brought us to our fifth and sixth strategies, which are simple combinations of the four described above:

- In the first 150 to 200 generations, the algorithm uses the second strategy (the two point recombination procedure with four different crossing-over points between both parents). During all the remaining generations, the first strategy is used instead (i.e., the one point recombination procedure with a single crossing-over point for both parents).
- In the first 200 generations, the program uses

the second strategy; between generations 200 and 500 it switches to the fourth strategy, and above 500 generations it uses the first strategy.

The results of the combined strategies are much better than those of any of the four strategies applied separately, as shown in table 1. It can be observed that the first mixed strategy reaches, in just 600 generations, target distances similar to those attained by the first two strategies in over 2500 generations. The improvement of the mixed strategies is therefore quite impressive. On the other way, the two mixed strategies attain comparable results.

| Nr. of generations | Strategy 1 | Strategy 2 | Strategy 3 | Strategy 4 | First mixed strategy |
|---|---|---|---|---|---|
| 1 | 0.930 | 0.930 | 0.930 | 0.930 | 0.930 |
| 100 | 0.782 | 0.766 | 0.807 | 0.791 | 0.766 |
| 200 | 0.734 | 0.710 | 0.756 | 0.744 | 0.697 |
| 300 | 0.714 | 0.692 | 0.740 | 0.712 | 0.676 |
| 400 | 0.702 | 0.692 | 0.722 | 0.704 | 0.659 |
| 500 | 0.690 | 0.689 | 0.722 | 0.704 | 0.648 |
| 600 | 0.681 | 0.683 | 0.716 | 0.704 | 0.643 |
| 1000 | 0.663 | 0.682 | | | |
| 1500 | 0.658 | 0.666 | | | |
| 2000 | 0.656 | 0.658 | | | |
| 2500 | 0.644 | 0.652 | | | |

Table 1. A comparison of the performance of five different recombination strategies.

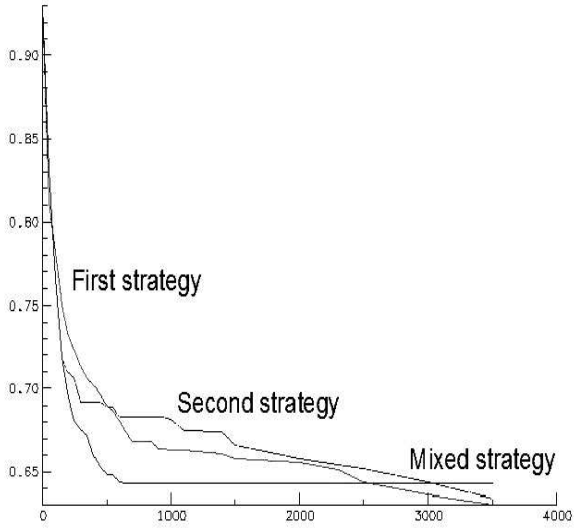


Figure 2. Comparison between three different recombination strategies.

Figure 2 shows a graphical representation of the results. Figure 3 shows the results of a different experiment with the same three strategies.

In our analysis of the reasons for this behaviour, we have come to the conclusion that, with the first strategy, the population reaches a smaller genetic variability, where favourable mutations have a greater probability of appearing. On the other

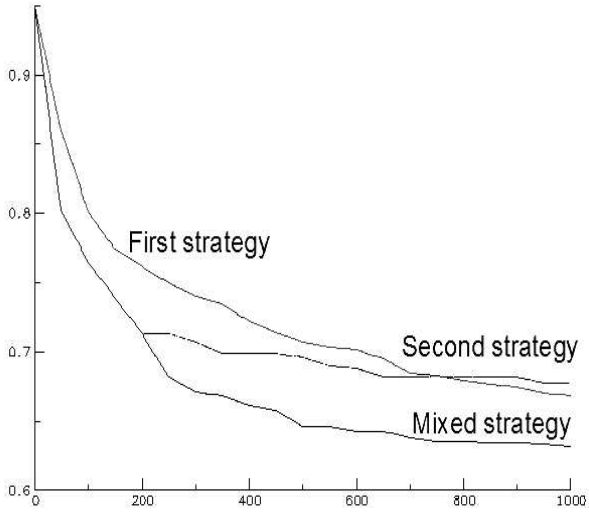


Figure 3. Performance comparison of another experiment with the same recombination strategies.

hand, the second strategy generates a much greater genetic variability, both with respect to genome lengths and contents, where favourable mutations are much harder to come by. This means that, on the long range, the first strategy should work better than the second, which on the other hand gets faster results during the first part of the process, by evolving simultaneously in many directions and testing widely different genomes at the same time. Thus, the mixed

strategy makes the best use of both recombination procedures, which is the reason for its outstanding performance success.

## 7. Conclusions and future work

We prove that the normalized compression distance is a promising fitness function for genetic algorithms used in automatic music generation. Some of the pieces of music thus generated remind the style of well-known authors, in spite of the fact that the fitness function only takes into account the relative pitch envelope. Our results have been much better than those obtained previously with a different fitness function [18].

In the future we intend to combine our results with those of other authors [14-15] to use as the target for the genetic algorithm, not one or two pieces of music by a given author, but a cluster of pieces by the same author, thus trying to capture the style in a more general way. We also intend to modify the algorithm to use the information about note duration.

We shall also work with a standard and richer system of music representation, such as MIDI.

*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 (1994). 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 (1994). Generating Rhythms with Genetic Algorithms, *Proceedings of the ICMC 1994*, pp. 142-143, International Computer Music Association, Århus, 1994.

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

[8] Alfonseca, M., Ortega, A., Suárez, A. (2003). Cellular automata and probabilistic L systems: An example in Ecology, in *Grammars and Automata for String Processing: from Mathematics and Computer Science to Biology, and Back*, ed. C. Martin-Vide & V. Mitrana, Taylor & Francis, pp. 111-120. ISBN: 0415298857.

[9] Ortega, A., Abu Dalhoum, A., Alfonseca, M. (2003). Grammatical evolution to design fractal curves with a given dimension, *IBM Journal of Research and Development*, Vol. 47:4, p. 483-493, Jul. 2003.

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

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

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

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

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

[15] R. Cilibrasi and P. Vitanyi (2004). 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.

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

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

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