A simple genetic algorithm for music generation by means of algorithmic information theory

Conference Paper · September 2007

DOI: 10.1109/CEC.2007.4424858 · Source: DBLP

3 authors, including:

Manuel Alfonseca
Universidad Autónoma de Madrid
183 PUBLICATIONS **1,082** CITATIONS
SEE PROFILE

Alfonso Ortega De la Puente
Universidad Autónoma de Madrid
70 PUBLICATIONS **536** CITATIONS
SEE PROFILE

A Simple Genetic Algorithm for Music Generation by means of Algorithmic Information Theory

Manuel Alfonseca, Manuel Cebrián and Alfonso Ortega

Abstract— **Recent large scale experiments have shown that the Normalized Information Distance, an algorithmic information measure, is among the best similarity metrics for melody classification. This paper proposes the use of this distance as a fitness function which may be used by genetic algorithms to automatically generate music in a given pre-defined style. The minimization of this distance of the generated music to a set of musical guides makes it possible to obtain computer-generated music which recalls the style of a certain human author. The recombination operator plays an important role in this problem and thus several variations are tested to fine tune the genetic algorithm for this application. The superiority of the relative pitch envelope over other music parameters, such as the lengths of the notes, brought us to develop a simplified algorithm that nevertheless obtains interesting results.**

I. 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 of history at its back.

Some of the current approaches try to simulate how the musicians play [1] or improvise [2] on the fly, while others are not concerned with execution time and mainly try to generate some 'good' output. Many of them apply models and procedures of theoretical computer science (cellular automata [3], parallel derivation grammars [1], or evolutionary programming [4], [5], [6], [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.

In a previous paper [8] we proposed the use of the well-known Normalized Compression Distance [9], an algorithmic information measure , 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 in [10], bringing us to develop a simplified algorithm that nevertheless obtains interesting results.

In this paper we start on the results of the previous work and refine them, trying to increase the efficiency of the procedures described in the above mentioned paper. This is done by testing several variations of the recombination operator to fine tune the genetic algorithm for this application, as it has been observed that this operator plays an important role in this procedure.

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 introduces the Normalized Compression Distance, which has been used to compute the distance from the results of the genetic algorithm to 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 six different recombination procedures for the genetic algorithm. Finally, the last section presents our conclusions and possibilities for future work.

II. 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 the English notation for 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*b* or B-, which represent B flat). 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

M. Alfonseca, M. Cebrián and A. Ortega are with the Escuela Politécnica Superior of the Universidad Autónoma de Madrid, Tomas y Valiente 11, P. O. Box 28049, Madrid, Spain (e-mail: {manuel.alfonseca, manuel.cebrian, alfonso.ortega}@uam.es).

This work has been supported by grant TSI 2005-08255-C07-06 of the Spanish Ministry of Education and Science.

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 (whole, half, quarter, ...). Intermediate durations can be obtained by means of dots or periods, triplets and other constructs. The complete specification of notes and silences includes their lengths.

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

1) With the traditional Western bi-dimensional graphic notation on a pentagram.
2) 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.
3) By numbering (1 to 88) the pitches of the notes in the piano keyboard. Note 0 would represent a silence. The length of a note can be represented by 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. Polyphonic music may be represented by means of parallel sets of integer pairs.
4) 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.

III. The Normalized Compression Distance

The search for a universal distance has been, for a long time, one of the main objectives of cluster theory. The availability of such a distance 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.

A deep result from Algorithmic Information Theory is that there exists such a universal similarity distance, which summarizes all computable similarities: the Normalized Information Distance (NID) [11]. It is universal in the sense that, when when a small distance is measured by any means between any two given objects, the NID is also small between these objects. Thus, it is at least as good as any other computable similarity distance. The NID is mathematically defined as follows:

$$\text{NID}(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \quad (1)$$

where $K(x|y)$ is the conditional Kolmogorov complexity of string x given string y, whose value is 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 λ is the empty string; see [12] for a detailed exposition on the subject. Unfortunately it can be proven that, due to the well-known *halting problem*, both the conditional and the unconditional complexities happen to be incomputable functions.

In [9] a computable estimate of the NID, the Normalized Compression Distance (NCD), is presented:

$$\text{NCD}(x,y) = \frac{\max\{C(xy) - C(x), C(yx) - C(y)\}}{\max\{C(x), C(y)\}} \quad (2)$$

where xy is the concatenation of strings x and y, and $C(x)$ denotes the length of the text x compressed by some compression algorithm which approximates $K(x)$ from above. Distances near 0 indicate similarity between objects, while those near 1 reveal dissimilarity.

Li and Sleep have reported that this distance, together with a nearest neighbor or a cladistic classifier, outperforms some of the finest (more complex) algorithms for clustering music by genre [10]. An earlier research has also reported success of the same distance for clustering tasks [13]. These results suggest that the NCD, although not achieving the universality of its incomputable predecessor (the NID), works well at extracting features shared between two musical pieces.

On the other hand, genetic algorithms need to define a fitness function 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 to a desired goal.

Assume 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 as many as possible of Mozart's symphonies. We propose to use a genetic algorithm (with musical compositions as individuals of the population) which uses the NCD as the fitness measure. This measure may compute these shared features between the individuals and the target musical guides which, in this example, would be the set of Mozart's symphonies.

It remains to choose the compressor used to estimate the NCD. Li and Sleep compute it by counting the number of blocks generated by executing the LZ78 compression algorithm [14] on an input. In our initial experiments, we used both the LZ78 and LZ77 algorithms, and found that LZ77 performs better, which agrees with theoretical results from Kosaraju and Manzini [15]. Therefore, LZ77 has been used as our reference compressor in all the experimental results presented in this paper.

IV. The genetic algorithm used to generate music

Our genetic algorithm generates music coded as pairs of integers, the third format described in section II, which is specially fitted for our purpose. This notation can then be transformed to a note string (the second representation) for reproduction.

As we have previously stated (see sect. II), we also decided to start with monophonic music, leaving harmony for a later phase and working with melodies.

Finally, we made the decision, in this first stage 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. The reason is that our own studies and others' [10] suggest that a given piece of music remains recognizable by a human being when the lengths of its notes are replaced by random lengths, while the opposite does not 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 (M) musical pieces of the same style/author are selected as targets or guides for music generation. We define each of the guides as g_i and the guide set as $G = \{g_i\}_{i=1}^{M}$.
- All the guides are coded as pairs of integers, as described above. In our experiments, the set of guides may include pieces of music of unequal (but not too dissimilar) lengths.
- The individuals in the population are coded in the same way as the guides.
- The fitness function for an individual x and a guide set G is defined as

$$f(x) = \left(\sum_{g_i \in G} \text{NCD}(x, g_i) \right)^{-1} \tag{3}$$

  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.

The remaining steps of the genetic algorithm are:

1) The program generates an initial 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 and represents the note interval. The second is in the [1,16] interval and represents its length as multiple of the minimum unit of time. Each vector represents a genotype.
2) The fitness of the genotypes is computed as in eq. 3, where x is the relative pitch envelope, i.e. the set of running differences between each note and the next, while lengths are ignored.
3) The genotypes are ordered by their increasing distance to the guide set, i.e. decreasing fitness.
4) If some predetermined fitness has been reached, the genetic algorithm 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 lowest fitness are removed. The 16 genotypes with highest fitness are paired randomly. Each pair generates a pair of children, a copy of the parents modified by four genetic operators. 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.

We need to say some words about our coding choice. The use of only two octaves for the notes (i.e. [24,48]) does not represent an important restriction (actually many real melodies comply with it), while it reduces significantly the size of the search space. The fact that absolute notes are later converted to intervals has the consequence that a piece of music becomes invariant under transposition, which is proper, because human ear recognizes transposed pieces as very similar.

The second number, belonging to the [1,16] interval in each pair, represents the length of the note and is currently ignored (remember that these lengths are replaced by a function of the lengths of the guide pieces). In this way, however, the system is ready for the future objective of automatically generating the lengths.

The four genetic operators mentioned in the algorithm are:

- Recombination (applied to all 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 strategy (see sect. VI):
  - Single point crossover, adjusted for variable length genomes: given a pair of genotypes, $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_m)$, a random integer is generated in the interval $[0, \min(n, m)]$, let it be i. The resulting recombined genotypes are: $(x_1, x_2, \ldots, x_{i-1}, y_i, y_{i+1}, \ldots, y_m)$ and $(y_1, y_2, \ldots, y_{i-1}, x_i, x_{i+1}, \ldots, x_n)$.
  - Modified two-point crossover for variable length genomes: given a pair of genotypes, $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_m)$, 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: $(x_1, x_2, \ldots, x_{i-1}, y_p, y_{p+1}, \ldots, y_{q-1}, x_j, x_{j+1}, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_{p-1}, x_i, x_{i+1}, \ldots, x_{j-1}, y_q, y_{q+1}, \ldots, y_m)$.
  - Recombination based on a four point crossover: given a pair of genotypes, $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_m)$, four random ordered integers are

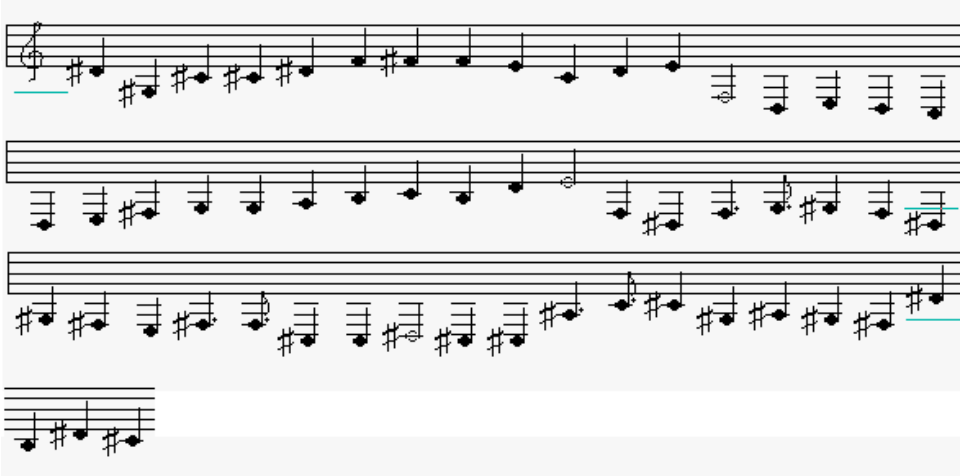

Fig. 1. A piece generated with *Yankee Doodle* as the only guide (NCD 0.43).

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). Finally, the genotypes of the progeny are obtained by concatenating five of the pieces in the shuffled

  - Recombination based on a three point crossover: 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.

- 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 individual genotype 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 paired 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 vectors.

V. Testing different number of guide pieces

In our first experiments, we selected the simplest recombination procedure (strategy 1 in sect. VI) and tested the effect of varying the number of guide pieces and the functions which generate the lengths of the notes in the best output pieces.

First, we used *Yankee Doodle* as the guide a single piece of music, described by the following string with the second representation defined in sect. II:

```
M2T2O3L2C+4C+4D+4F4C+4F4D+4O2G+4O3C+4C+4D+4F4
C+3C4P4C+4C+4D+4F4F+4F4D+4C+4C4O2G+4A+4O3C4C+
3C+4P4O2A+4.O3C5.O2A+4G+4A+4O3C4C+4P4O2G+4.A+
5.G+4F+4F3G+4P4A+4.O3C5.O2A+4G+4A+4O3C4C+4O2A
+4G+4O3C+4C4D+4C+3C+3
```

The corresponding WAV formatted file, Yankee.wav, together with all the musical pieces mentioned in this paper, can be found at:

www.eps.uam.es/˜mcebrian/music

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. (see also figure 1, where the same music appears in standard musical notation), named Yankee_NEW.wav is 0.43:

```
T5O3D+2O2G+2O3C+2C+2D+2F2F+2F2E2C2D2E2O2F1D2E
2D2C2D2E2F+2G2G2A2B2O3C2O2B2O3D2E1O2F2D+2F2.G
3.G+2F2D+2G+2F+2E2F+2.F3.C+2C2D+1C+2C+2A+2.O3
C3.C+2O2G+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 2, which represents the result of one experiment.

In our second experiment, we used two guide pieces

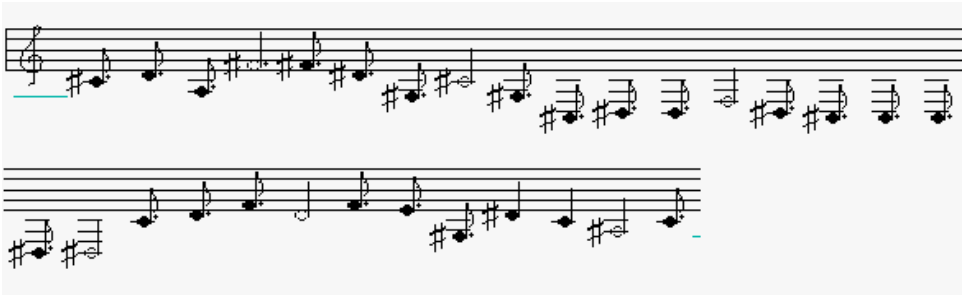

Fig. 3. Beginning of a piece generated with two songs by Cole Porter as guides (NCD 0.67 from *Begin the beguine* and 0.72 from *My heart belongs to daddy*).

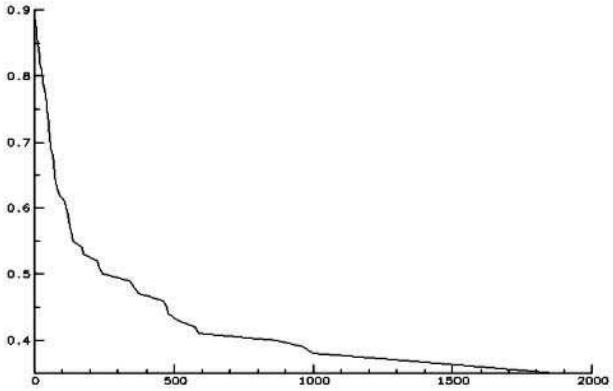

Fig. 2. Number of generations needed to reach a given distance to the target (only one guide).

simultaneously: *Begin the beguine* (Begin.wav), and *My heart belongs to daddy* (Heart.wav), both by Cole Porter.

The following represents one of the results we obtained (figure 3 shows the first notes in standard musical notation), which we named as Porter_NEW.wav, which happens to be at a distance of 0.67 from the first guide piece, and 0.72 from the second, while the NCD 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.O3C+1O2G+
3.C+3.D+3.D3.F1D+3.C+3.C3.C3.C+3.D+1O3C3.D3.F
3.D1F3.E3.O2G+3.O3D+2C2O2A+1O3C3.O2A+3.A+3.A+
1A+1G+3.E1.F+3.O4C3.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.F1
O3D3.E2D+2E1.D+3.G+3.O2D+3.D+1G3.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.O2B2O3E
2O2A1A2G+3.G+3.E2.F+3.F3.D+2F1D3.D+3.O3F2D+3.
F+3.D1O2A2G+3.O3C+3.G+2F2C+2O2A2F1O3F+3.B2.O2
F+3.E3.G3.F+1E3.E3.D+3.C+3.O3C+1.O+2G+1C+3.C+
1D+3.F3.A+2G+2G+3.F+1D+2E3D0D+2F3.D+3O3G3.D2B
2O2D+2O3C3C3C3.C2O2B0A3.A3A3.B2.O3C3.F+1G3.A+
3.G+3F+3A3.F1.C2O2G+3.G+3.O3G+0A+3.B3.B0B3.O2
E3A+3B3.O3D3.D3.O2A+1O3D+3F+3F0F+3.D+3F3G3.G3
.G3.G+2A+3O4C3O3A3A+2G+0O2F+3G+3F+3.F3.F+3.O3
G+2F+3A3A+3G+3.F+1D+3.C+3.C3.O2A+3.A+0A+3.O3C
3.O2A+3.O3C3.C+0D3.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's *prelude number 4* (Chopin4.wav) and *prelude number 7* (Chopin7.wav) were used as simultaneous guides. The result (Chopin47_NEW.wav, see also the first notes in figure 4) 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.O2E2O2
E0O2B1.O3C2O3D+3.O3D+2.O3D+2.O3D1O3C+2.O2A+1.
O2A2O2G+0O2G+2O2A1O3C2.O3E2O3G3.O2B2.O3D2.O3C
1O4C2.O4C2O2C3O2D0O2F1O2D+0O2A1O3F+1.O3G2O3E2
.O2F+2O2B1.O2B2O2B3.O2D+4O2G+2O2F1O2G+1O2F2O2
F+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.
```

We performed another two experiments which the reader can also find online. One of them generates a piece (Chopin7_NEW.dat, see figure 5) at a 0.39 distance from its guide, Chopins's *seventh prelude*. The lengths were generated as in the first experiment:

```
T5O4C1O3E2.C+3A+1A+1A+0O2A+1O3C+2.O2B3O3B1B1B
0D+1D+2.E3G+1O4C1O3E0C+1D2.F3F1F1O2B0C+1A+2.B
3G+1G+1G+0O3G+1G+2.G+3O2B1O3E1E0E1O2B2.O3F+3D
+1E1A0A1A2.
```

Finally, in the last experiment, two pieces by Mozart were used as simultaneous guides: a few bars of the first

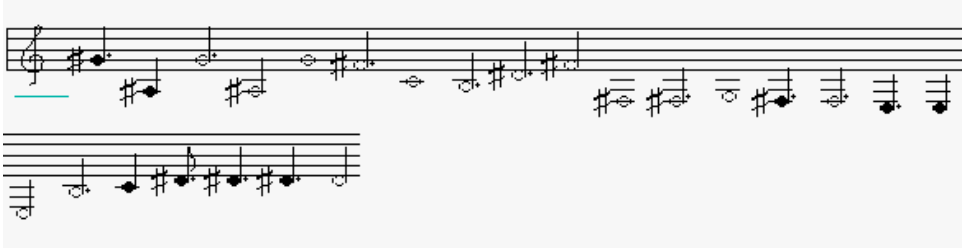

Fig. 4. Beginning of a piece generated with two Chopin preludes as guides (NCD 0.61 from *prelude number 4* and 0.74 from *prelude number 7*).

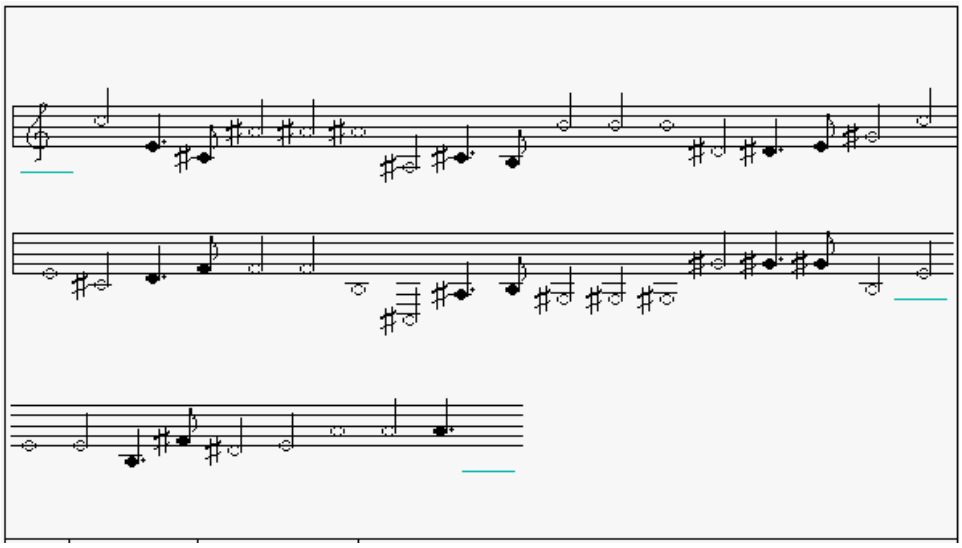

Fig. 5. A piece generated using Chopin's seventh prelude as the only guide (NCD 0.39).

movement in *symphony 40* (Mozart40.wav), and a part of the second movement in *sonata KV545* (KV545.wav). The result (Mozart_NEW.wav), which sounds like a mixture of both sources 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:

```
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+2A2O4G2G+2G1G+3.F3.G+3.O5C3O4A+3G+3
.A+3.A+2.G+3.G3.G3.E3D+3O3F+3F3F+3.G3.G+3.A3.
A3.G3F+3F+3.D+3.D3.O4D3.C+2C+3E2O3A+2O4C+3C2O
3B2G1B1O4D3.C3.O3G3.O4D+3G3D3.D+3D3D+3.D3E3F3
.G3.F3.E3F3G1O3D+3E3D+3.D+3B3B3.A+3.O4F3.G2.G
+3A+3.G3
```

The length of the notes was generated in the same way as in the second experiment.

VI. Testing different recombination procedures

We have evidence that the recombination operator plays a key role in our approach, both in the quality of the generated musical pieces and in the time the algorithm takes to generate it.

In order to fine tune the genetic algorithm for this application, we devote a section to discuss several variations we have tested experimentally. We analyzed four strategies that use respectively the four types of recombination described in section III: strategy 1 (single point crossover, adjusted for variable length genomes) is the base case (the simplest recombination strategy) which was used in all the experiments described in the preceding section, strategy 2 (modified two-point crossover for variable length genomes), strategy 3, (recombination based on a four point crossover) and strategy 4 (recombination based on a three point crossover)

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 experiments, 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 from their initial value.

TABLE I

A COMPARISON OF THE PERFORMANCE OF DIFFERENT RECOMBINATION STRATEGIES FOR A TYPICAL MUSIC GENERATION EXPERIMENT. NP STANDS FOR 'NOT PERFORMED.'

number of generations	strategy 1	strategy 2	strategy 3	strategy 4	mixed strategy 1
1	0.930	0.930	0.930	0.930	0.930
100	0.782 (-0.148)	0.766 (-0.164)	0.807 (-0.123)	0.791 (-0.139)	0.766 (-0.164)
200	0.734 (-0.048)	0.710 (-0.056)	0.756 (-0.051)	0.744 (-0.047)	0.697 (-0.069)
300	0.714 (-0.020)	0.692 (-0.018)	0.740 (-0.016)	0.712 (-0.032)	0.676 (-0.021)
400	0.702 (-0.012)	0.692 (-0.000)	0.722 (-0.018)	0.704 (-0.008)	0.659 (-0.017)
500	0.690 (-0.012)	0.689 (-0.003)	0.722 (-0.000)	0.704 (-0.000)	0.648 (-0.011)
600	0.681 (-0.009)	0.683 (-0.006)	0.716 (-0.006)	0.704 (-0.000)	0.643 (-0.005)
1000	0.663 (-0.018)	0.682 (-0.001)	NP	NP	NP
1500	0.658 (-0.005)	0.666 (-0.016)	NP	NP	NP
2000	0.656 (-0.002)	0.658 (-0.008)	NP	NP	NP
2500	0.644 (-0.012)	0.652 (-0.006)	NP	NP	NP

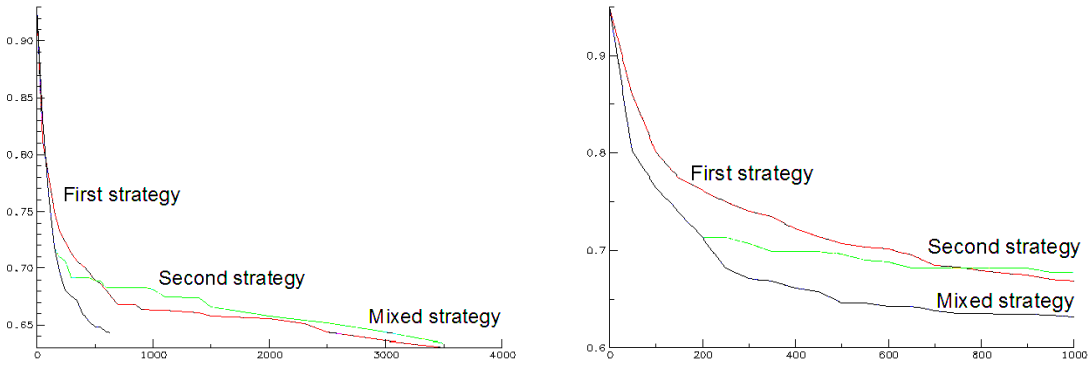

Fig. 6. Two different experiments with a comparison of three recombination strategies. 'Mixed strategy' refers to the mixed strategy 1.

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 larger genome length diversity 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 behave 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 add two new strategies to the testbed, which are simple combinations of the four described above:

Mixed strategy 1: 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).

Mixed strategy 2: 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 data in table I correspond to a typical experiment in which two Mozart's pieces were used as the guide set: *Symphony 40* and *KV545*. The results of the combined strategies are much better than those of any of the four strategies applied separately. It can be observed that the mixed strategy 1 reaches, in just 600 generations, target distances similar to those attained by the first two strategies

after 2500 generations.

Tabulated values for mixed strategy 2 are not shown in table I because they are quite similar to the performance of mixed strategy 1. In case of strategy 3 and 4, no data are shown for more than 600 generations; the reason is that they are clearly outperformed by strategy 1 and 2 in all executions before that point, and no further improvement was experimentally observed.

Figure 6 shows a graphical representation of two experimental results with three strategies on the same guide set. Summarizing, the improvement of the mixed strategies is quite impressive. On the other hand, the two mixed strategies attain comparable results.

In our analysis of the reasons for this behavior, we come to the conclusion that, with the first strategy, the population has a small genetic variability where favorable mutations have a great probability of appearing. On the other hand, the second strategy generates a large genetic variability, both with respect to genome lengths and contents, where favorable mutations are 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 strategies make the best use of both recombination procedures, which is the reason for their outstanding performance success.

The number of experiments performed was not large (tens, rather than hundreds), but the results obtained are consistent and show a small variability, which makes it unlikely that they may be a statistical fluke.

VII. Conclusions and future work

We have found that 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 recall the style of well-known authors, in spite of the fact that the fitness function only takes into account the relative pitch envelope. Qualitative response by human audiences confirms that the results described in this paper are superior to those obtained previously with a different fitness function [16].

Several recombination operators have been tested to fine tune the genetic algorithm for this application, finding that mixed strategies which promote diversity in the first generations and then change to a more exploitative strategy give the best results. This scheme of initial exploration and posterior exploitation is analogue to the idea behind Simulated Annealing [17].

In the future we intend to combine our results with those of other authors [10], [13] and 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.

Our current experiments focus the search on melodies which can be considered an average of the target pieces. In a future work, we intend to add a component to the system that allows and encourages outliers as well.

Although we have introduced the information about note duration in the genetic process, it has been ignored so far. We intend to experiment with different strategies (such as setting all the notes to the same length). Furthermore, as the NCD can easily deal with integers representing note lengths, we intend to let the note length information evolve together with the melody. We shall also experiment with richer systems of music representation, such as MIDI.

This paper serves as a proof-of-concept. As future research, we plan to provide a comparison with state-of-the-art music composition techniques based on machine learning, to reveal both the strengths and the weaknesses of our proposal.

References

[1] J. McCormack, "Grammar based music composition," *Complex Systems*, 1996.

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

[3] E. Bilotta and P. Pantano, "Synthetic Harmonies: An Approach to Musical Semiosis by Means of Cellular Automata," *LEONARDO*, vol. 35, no. 2, pp. 153–159, 2002.

[4] D. Lidov and J. Gabura, "A melody writing algorithm using a formal language model," *Computer Studies in the Humanities*, vol. 4, no. 3-4, pp. 138–148, 1973.

[5] P. Laine and M. Kuuskankare, "Genetic algorithms in musical style oriented generation," *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 858–862, 1994.

[6] D. Horowitz, "Generating rhythms with genetic algorithms," *Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, 1994.

[7] B. Jacob, "Composing with genetic algorithms," *Proceedings of the 1995 International Computer Music Conference*, pp. 452–455, 1995.

[8] M. Alfonseca, M. Cebrian, and A. Ortega, "Evolving Computer-Generated Music By Means of the Normalized Compression Distance," *WSEAS Transactions on Information Science and Applications*, vol. 2, no. 9, pp. 1367–1372, 2005.

[9] R. Cilibrasi and P. Vitani, "Clustering by Compression," *Information Theory, IEEE Transactions on*, vol. 51, no. 4, pp. 1523–1545, 2005, software available at http://www.complearn.org.

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

[11] M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi, "The Similarity Metric," *Information Theory, IEEE Transactions on*, vol. 50, p. 12, 2004.

[12] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

[13] R. Cilibrasi, P. Vitanyi, and R. de Wolf, "Algorithmic clustering of music," *Proceedings of the Fourth International Conference on Web Delivering of Music, 2004. WEDELMUSIC 2004.*, pp. 110–117, 2004.

[14] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *Information Theory, IEEE Transactions on*, vol. 23, no. 3, pp. 337–343, 1977.

[15] S. Kosaraju and G. Manzini, "Some entropic bounds for Lempel-Ziv algorithms," *Proceedings of the Conference on Data Compression*, 1997.

[16] A. Ortega, R. Alfonso, and M. Alfonseca, "Automatic composition of music by means of grammatical evolution," *Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications*, pp. 148–155, 2002.

[17] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, p. 671, 1983.