A Novel Approach to Automatic Music Composing: Using Genetic Algorithm

Damon Daylamani Zad [*], Babak N. Araabi[†] and Caru Lucas[**]

[*]Department of Information Systems and Computing, Brunel University
ci05ddd@brunel.ac.uk
[†]Control and Intelligent Processing Center of Excellence, Dept. of Electrical and Computer Engineering, University of Tehran
araabi@ut.ac.ir
[**]Control and Intelligent Processing Center of Excellence, Dept. of Electrical and Computer Engineering, University of Tehran
lucas@ipm.ir

Abstract

Artificial music composition is one of the ever rising problems of computer science. Genetic Algorithm has been one of the most useful means in our hands to solve optimization problems. By use of precise assumptions and adequate fitness function it is possible to change the music composing into an optimization problem. This paper proposes a new genetic algorithm for composing music. Considering entropy of the notes distribution as a factor of fitness function and developing mutation and crossover functions based on harmonic rules and trying to keep the melodies intact during these processes would result in a musical piece pleasant to human ears and interesting for human mind. This algorithm does not have the constraints of the previous algorithms. Restraining mutation and crossover functions with a goal of producing melodies based on acceptable melodies composed by humans, this algorithm is not bound to any genre, instrument or melody. The experimental results of this approach show that it is near to the human composing and the results produced from it are more acceptable than the ones produced by its predecessors.

1 Introduction

Music and Composing music has always amazed human beings. The process of composing a musical piece, the rhythms and melodies has ever astonished us.

In order to understand this area many have tried to formulate these processes. The result of these attempts is the Harmony and Melody rules that have been established in 16th centaury, yet they change everyday by new professionals and new rhythms made by composers. The most important reason for these changes is that composing is the product of the state of mind, emissions and talents etc. of a composer. It seems impossible to formulate these factors.

By entering the computer era a new idea emerged. Scholars wondered if it was possible to create an artificial composer. This problem raised the question of simulating human emissions and talents by computers.

Over the years there have been many attempts to create an artificial composer. Contemporary algorithmic composition ranges from traditional stochastic methods seen in M and Jam Factory (Zicarelli, 1987) to complex rule-based systems such as EMI (Cope, 1987, 1992) and Cypher (Rowe, 1993). Later on Genetic Algorithm became a popular way of solving this problem. Horner, A. (1991) describes the application of genetic algorithms to thematic transformation, yet he only deals with morphing one melody into another. Biles, J. (1994, 2001, 2002a, 2002b) describes a genetic-based jazz soloist, he also only generates single melodies on top of given chord progressions and Horowitz, D. (1994) describes a genetic algorithm for creating interesting rhythms but deals with rhythms that span only one measure. Jacob, Bruce L. (1995) presents his application of genetic algorithms *Variations*, although very effective but half human driven, Using human ear as a part of fitness function. Moroni et al (2000) present another Genetic based algorithm for algorithmic music composition, their algorithm still suffers the human fatigue problem as all other Interactive Evolutionary Algorithms. Ayesh and Hugill (2005) describe their genetic approach for evolving of music forms into another. Later on Tuohy and Potter (2005) present their algorithm for creating guitar tablatures. Although quit interesting, this algorithm only produces only progressions of chords for guitar without any considerable melody.

Here presented is a new approach to the problem using genetic algorithm and MIDI file format, creating complete pieces with nearly no human supervision. Having no restriction on instruments, genre, composer and rhythm this application can create many kinds of music according to the initial population and the instruments specified. This application produces musical pieces with interesting melodies and rhythms and pleasant to human ears. These pieces can nearly compete with pieces composed by real composers in creativity, style and amusement.

The rest of this paper is organized as follow. In the next section a quick overview of MIDI file format is provided, follows a description of some of its features. Then it goes on a study of the genetic algorithm proposed in this project. Then presented are experimental results in section 4. Finally, is concluded with a summary in section 5.

2 Musical Instruments Digital Interface Standard (MIDI)

2.1 History

MIDI, the Musical Instruments Digital Interface standard, was established in 1983, and has since revolutionized the world of electronic music. First created to help two synthesizers communicate with each other, MIDI soon took over the electrical music world with his wide use in PC's as the musical file format.

2.2 Format

MIDI files contain one or more MIDI streams, with time information for each event. Song, sequence, and track structures, tempo and time signature information, are all supported. Track names and other descriptive information may be stored with the MIDI data. This format supports multiple tracks and multiple sequences so that if the user of a program which supports multiple tracks intends to move a file to another one, this format can allow that to happen.

Each sequence contains tracks and each track contains note events that together create a musical piece. Each note event contains the following data:

- *Onset*: Specifying the moment of track when the note starts to play.
- *Duration*: The duration of a note being played.
- *MIDI Channel*: Indicating by which instrument the note is being played (1 – 16).
- *MIDI Pitch*: Denotes the note on numeric basis, where middle C (C4) is 60.
- *Velocity*: Describes how fast the key of the note is pressed, in other words, how loud the note is played (0 - 127).

3 Our Genetic Algorithm Based Approach

The approach of this project to the composing problem is based on genetic algorithms, transforming the dilemma into an optimization problem of optimizing the harmonic relationship between the notes and producing purposeful melodies with as less repetition as possible.

3.1 Chromosomes and Genes

The project begins with defining the genetic algorithm's genes and chromosomes as they are the basis of this algorithm.

- Chromosome: Each string of notes in a musical piece is defined as a chromosome in this project. Leaving the algorithm with a vast field of chromosomes to work on and the returning answer would be a complete song. As a result Crossovers occur on songs.
- Gene: Each note presents a gene. So Mutations take place on notes themselves.

The other three most important factors are Mutation, Crossover and Fitness functions. These functions work on genes and chromosomes produce, control and optimize the results.

3.2 Mutation Function

Mutation function is based on harmonic and melodic rules with goal of producing new melodies based on old ones, without disturbing the purpose of the song or conflicting with harmonic rules.

In order to gain such function it was necessary to calculate the scale of each song. This is done using Krumhansl & Schmuckler (1990) key-finding algorithm. This algorithm is based on key profiles obtained from empirical work by Krumhansl & Kessler (1982), where listeners heard a context sequence, consisting of an incomplete major or minor scale or a chord cadence, followed by each of the chromatic scale pitches in separate trials. In this key-finding algorithm, the 24 individual key profiles, 12 major and 12 minor key profiles, are correlated with the pitch-class distribution of the piece weighted according to their duration. This gives a measure of the strength of each key. It is possible to see the approach of this algorithm in Figure 1.

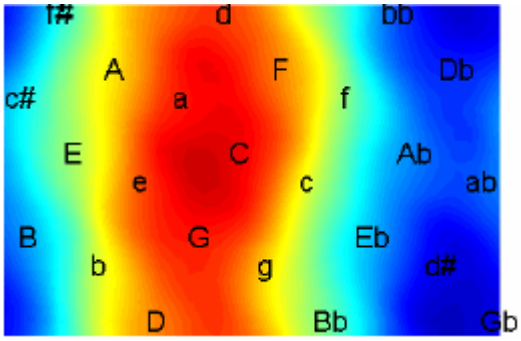

Figure1. Self-organizing map (SOM) of the tonality in Bach's C-major Prelude, *Wohltemperierte Klavier II* (BWV 870)

When the scale is known to us, we can decide on the available mutations for each note according to its scale. Each note is allowed to change on the range of the notes on the key's original chord or on the notes of it's chord's relative chords.

This approach is implemented using a random number generated from a normal distribution. This number is between 0 and 1. If this number is smaller than 0.25, the mutating note will not mutate or will mutate to the same note yet in another octave. If the number is greater than 0.75, the note is mutated to its 7^{th} distance on the related chord in this scale either in the same octave or another octave. Finally, if the number is between 0.25 and 0.75 which means that it is in the peak of the distribution and has numbers have higher probability to be in this interval, then the note will mutate to either its 4^{th} distance on the related Minor chord if the scale was Major or to the 4^{th} distance on the main chord if it was a Minor scale. The reason for this emphasis on the 4^{th} distance is that the 4^{th} distance is the best harmonic substitute of a note.

After this decision is made, another random number from a normal distribution is produced. This number is also between 0 and 1. If this number is smaller than 0.1, the mutated note will be in a random number of octaves before the original note's octave. If the random number is between 0.1 and 0.25 the new note will be in the previous octave of the original note and in case the number is between 0.25 and 0.75 the new note will be in the same octave as the original note. Then if the number is between 0.75 and 0.9 the note will be on the next octave of the original note and at last if the number is between 0.9 and 1, the note will be in a random number of octaves higher than the original octave. The reason for this approach is to prevent big jumps and discontinuity in the melody as much as possible without totally restricting jumps.

3.3 Crossover Function

The crossover function is not much restricted; it follows a uniform crossover pattern with a simple constraint so as to prevent segmentations in the middle of a melodic sequence.

This limitation is executed by the calculation of musical distance between successive notes played by the same instrument. This distance is defined as the pitch number of a note minus it's successor's pitch number. While these distances are only positive or only negative, it is not allowed to break the chromosome.

If the crossover function decides to break from a point that has the condition above that prevents breaking, the break will be rejected and the crossover will be preformed again.

3.4 Fitness Function

The fitness function of this project plays a great role for accepting the best song possible by far and to diminish the dissonance notes from the resulting song.

The fitness function of this program is calculated by minimizing the distance of the generated songs' entropy from the Mean-Square of the initial population's entropies

The withdrawing of dissonance notes is accomplished by multiplying the fitness value of the song by a fixed number so that it would be much greater than the fitness value of a song with no dissonance notes.

The major factor of fitness function of this program is entropy of distribution of notes. Entropy along with originality has been found to correspond to the predictability ratings given by listeners in experiments (Eerola, , Toiviainen & Krumhansl, 2002). This measure offers a possibility to observe the moment-by-moment fluctuations in melodic predictability.

Maximizing the entropy up to a certain amount guaranties there will not be any additional repetitions except the ones caused by the melody. This is known to be a key element in measuring the popularity of a music the amount of it's acceptance between people. Over-maximizing the entropy, results in an unpleasant and disturbing piece of music. To keep a balance the program is trying to have closer entropy to the initial population's entropy.

4 Experimental Results

In order to evaluate and analyze the performance of this genetic algorithm, MATLAB environment was used for developing this project. MATLAB has a toolbox for handling MIDI files. This toolbox transforms a MIDI file into a matrix such as Table 1.

ONSET (BEATS)	DURATION (BEATS)	MIDI channel	MIDI PITCH	VELOCITY	ONSET (SEC)	DURATION (SEC)
0	0.9000	1.0000	64.0000	82.0000	0	0.5510
1.0000	0.9000	1.0000	71.0000	89.0000	0.6122	0.5510
2.0000	0.4500	1.0000	71.0000	82.0000	1.2245	0.2755
2.5000	0.4500	1.0000	69.0000	70.0000	1.5306	0.2755
3.0000	0.4528	1.0000	67.0000	72.0000	1.8367	0.2772
3.5000	0.4528	1.0000	66.0000	72.0000	2.1429	0.2772
4.0000	0.9000	1.0000	64.0000	70.0000	2.4490	0.5510
5.0000	0.9000	1.0000	66.0000	79.0000	3.0612	0.5510
6.0000	0.9000	1.0000	67.0000	85.0000	3.6735	0.5510
7.0000	1.7500	1.0000	66.0000	72.0000	4.2857	1.0714

Table 1. First two verses of the Finish folk song Laskin in MIDI format.

The first test described here has the initial population containing 20 of J.S Bach's preludes imported from MIDI. The plot of fitness values generated for this test is illustrated in Figure 2. This test was done for 500 generations, considering that fitness values of generations with dissonance notes are multiplied by 1.5.

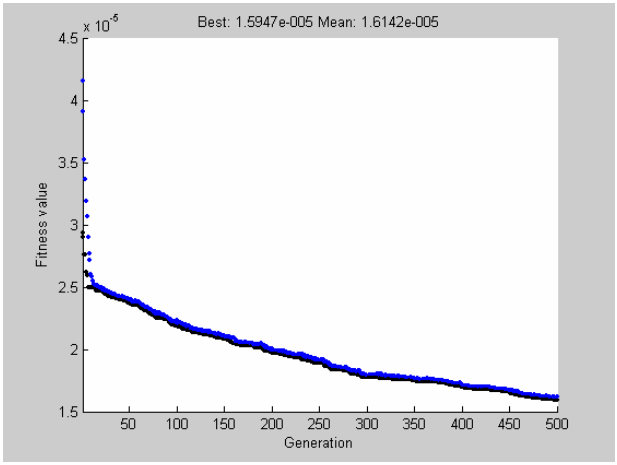

Figure 1. Fitness values for Bach's preludes. The graph shows a struggle for gaining better fitness value right from the start. Finding better generations in around every 15 generations, it is appealing to say that most of the good generations have been accepted for regeneration

The result of this test was played for 50 students in University of Tehran. Their opinion is shown in Figure3.

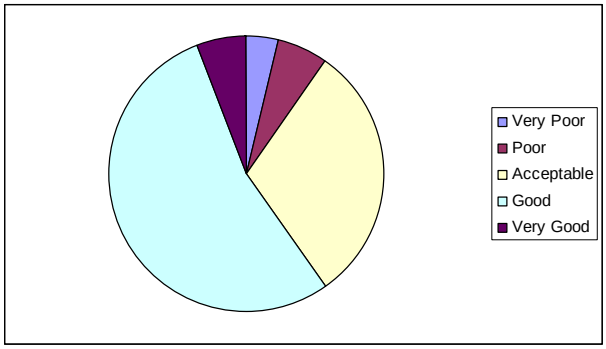

Figure 3. Opinions on the Prelude shows more than 50% of the listeners find this song amusing, considering those who have found the song acceptable, it is possible to say that this experiment has been rather successful.

Another test was done, this time the initial population was 20 songs by Bob Dylan. Other variables were the same as the previous test except that it was running for 100 generations. The plot of fitness values of generations is demonstrated in Figure 4.

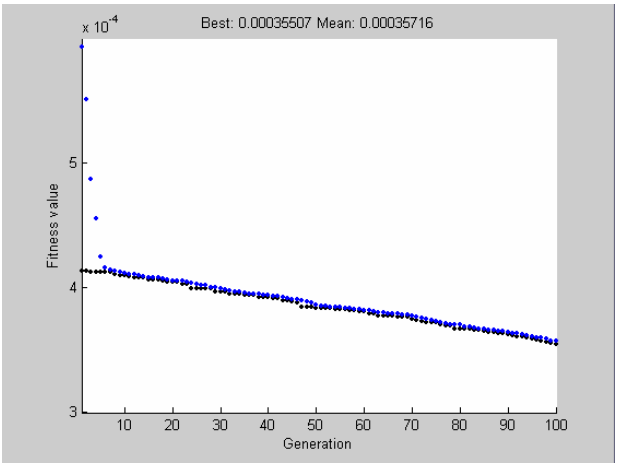

Figure 4. The graph shows a struggle for gaining better fitness value right from the start. Finding better generations in around every 15 generations, it is appealing to say that most of the good generations have been accepted for regeneration.

This song was played for 50 random students of University of Tehran. Their opinion on this song can be seen in Figure 5.

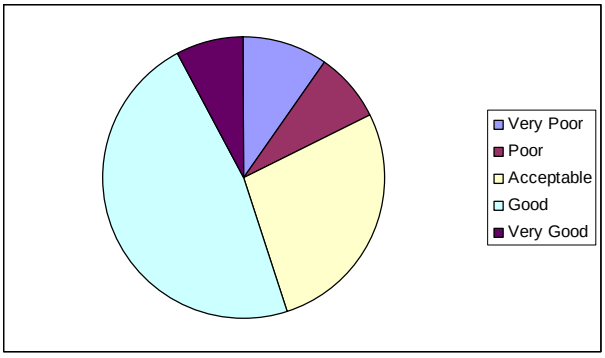

Figure 5. Opinions on the Dylan song shows more than 50% of the listeners find this song amusing, considering those who have found the song acceptable, it is possible to say that this experiment has been rather successful.

There links to these two songs at the end of references section.

5 Conclusion

This paper presented a new genetic algorithm for composing music. This algorithm not only has none of the previous algorithms' deficiencies but is near to the human composing and its results are more acceptable than the ones before. Having no barriers for instrument or genre, it can compose a vast portion of different musical pieces. These abilities can be seen clearly in fitness plots of the composing process and better yet can be heard and confirmed by human ears.

6 Acknowledgment

This paper would not be readable or understandable without the everlasting efforts and guidance of Hamed S. Alavi.

References

Aladdin Ayesh and Andrew Hugill, 2005, Genetic Approaches for Evolving Form in Musical Composition, Proceedings of the 23rd IASTED international Multi-Conference, Artificial Intelligence and Applications, February14-16, 2005, Innsbruck, Austria

Biles, J. 1994. “GenJam: A genetic algorithm for generating jazz solos”, In *Proceedings of the 1994 International Computer Music Conference*. Aarhus, Denmark: International Computer Music Association.

Biles, John A. 2001, “Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness”, Proceedings of the GECCO-2001 Workshop on Non-routine Designwith Evolutionary Systems.. 2001.

Biles, John A. 2002a, “GenJam in Transition: from Genetic Jammer to Generative” Jammer.Generative Art. 2002.

Biles, John A. 2002b, “GenJam: Evolutionary Computation Gets a Gig”, Proceedings of the 2002 Conference for Information Technology Curriculum, Rochester, New York, Society for Information Technology Education. September 2002.

Cope, D. 1987. "An expert system for computer-assisted composition.", *Computer Music Journal*, 11(4):30-46.

Cope, D. 1992. "Computer modeling of musical intelligence in EMI", *Computer Music Journal*, 16(2):69-83.

Eerola, T. & Toiviainen, P. 2004. MIDI Toolbox: MATLAB Tools for Music Research. Proceedings of ISMIR 2004, 5th International Conference on Music Information Retrieval, Barcelona, Spain, October 10-14, 2004

Eerola, T. Toiviainen, P., & Krumhansl, C. L. 2002. "Real-time prediction of melodies:Continuous predictability judgments and dynamic models. In C. Stevens, D. Burnham, G. McPherson, E. Schubert, J. Renwick (Eds.)". In *Proceedings of the Seventh International Conference on Music Perception and Cognition*, Sydney, 2002. Adelaide.

Horner, A. & Goldberg, D. 1991. “Genetic algorithms and computer-assisted music composition”, *Proceedings of the Fourth International Conference on Genetic Algorithms*, Urbana-Champaign, Illinois.

Horowitz, D. 1994. “Generating rhythms with genetic algorithms”, In *Proceedings of the 1994 International Computer Music Conference*, Aarhus, Denmark: International Computer Music Association.

Jacob, Bruce L. 1995, "Composing with genetic algorithms", *Proceedings of the* 1995 *International Computer Music Conference*, Banff, Alberta.

Moroni, Artemis, Manzolli, Jônatas, Zuben, Fernando Von, and Gudwin, Ricardo Vox Populi, 2000, “An Interactive Evolutionary System for Algorithmic Music Composition”, Leonardo Music Journal, Vol. 10, 49-54. 2000.

Rowe, R. 1993. “Interactive Music Systems”, Cambridge, Massachusetts: MIT Press.

Tuohy D. and Potter W.D. 2005, "A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature," In Proceedings of International Computer Music Conference ICMC'05, Barcelona, Spain, September, 2005.

Zicarelli, D. 1987. "M and Jam Factory", *Computer Music Journal*, Vol. 11, issue 4, pages 13-29.

khorshid.ut.ac.ir/~d.dzad/files/preludetest.mid
khorshid.ut.ac.ir/~d.dzad/files/dylantest.mid