Evolving Music with Genetic Algorithm

Nisha Shareef Shaikh
BSc. Computer Science
Class of 2020
Habib University
Karachi, Pakistan
ns02530@st.habib.edu.pk

Abeera Tariq
BSc. Computer Science
Class of 2020
Habib University
Karachi, Pakistan
at02787@st.habib.edu.pk

***Abstract*—This paper aims to use a genetic algorithm to produce music. This approach involves constructing melodies from random notes and pitches and evolving them to create a pleasant composition.**

***Index Terms*—music, evolution, genetic algorithm**

I. Introduction

Music composition involves innovation, creativity and a sense of melody. Through this project, we hope to discover how music can be composed without the involvement of musicians. Taking motivation from [1], we will try to utilize our knowledge of genetic algorithms to compose melodies.

Composing music is thought to be a process that can't work without human involvement due to the subjective nature of the task. However, since genetic algorithms mimic nature, they are believed to be a suitable approach to eliminate human involvement.

With the genetic algorithm we implemented, melodies will be represented as a list of integers that describe each beat's note, octave and duration. The algorithm will start with a group of random compositions created with random parameters to describe each beat. This group will be evolved with the algorithm to continuously improve the compositions and get a pleasant-sounding piece at the end.

II. Technical Background

The main technique used in this paper is the genetic algorithm. It has been inspired by the process of biological evolution and proves beneficial for optimization. Using concepts of reproduction, mutation, selection and recombination on candidate solutions (chromosomes), genetic algorithms can solve an optimization problem. A fitness function determines how good the candidate solution is and works by evolving the solutions to obtain the optimum solution to the problem.

The algorithm starts by generating a population with n chromosomes, where n is determined by the user and every chromosome denotes a candidate solution to the problem.

From this population, certain chromosomes are selected for reproduction on the basis of their fitness value (depending on the selection scheme used). For reproduction, a crossover method is used where a crossover point is selected, the two parents selected are split at this crossover point, and a new solution or offspring is created by joining the head of one parent with the tail of the other. Once the offspring have been generated and added to the population, they will be mutated with a certain probability. For mutation, a certain gene of the chromosome will be changed. The population will again be reduced to just n solutions by selecting the best n chromosomes from the current population.

This process of reproduction, mutation and selection will keep repeating until the criteria of termination have been fulfilled.

III. Related Work

Our presentation paper,

IV. Methodology

Our approach is to utilize our learning of genetic algorithm to compose music. We start by randomly generating notes to create random melodies, which will comprise our population. These melodies will then go through the reproduction, mutation and selection processes for a certain number of generations.

A. Chromosome Structure

The chromosome for a melody is represented as a list of 4-tuples, where each tuple defines a single beat. For example, a beat may be represented as $(0, 4, 32, 4)$, where each index is defined as follows

- Index 0: Note Index
- Index 1: Octave Index
- Index 2: Absolute Note
- Index 3: Duration

The note index will refer to the notes $C, D, E, F, G, A,$ and B along with rest (which, in music terminology, means an interval of silence). Hence, in the example above, the beat is a C note.

Octave is basically the interval between two notes with the same letter name with one having double or half the frequency of the other. For example, the octave can go from $C - C$ with one C note having half or double the frequency of the other C note. Hence, the beat above is a $C4$ note.

The absolute note represents the note and octave together as one integer to facilitate in the functioning of the genetic algorithm. It is basically the product of the octave index and the number of notes per octave added to the letter index.

The duration specifies the amount of time that the beat would play for. The duration d is a denominator in the fraction $\frac{1}{d}$, so in the example above, the beat will play for one-fourth or a quarter of a second. If the duration was 2, the beat would play for half a second.

[(0, 4, 32, 4), (6, 0, 6, 4), (4, 2, 20, 4), (6, 1, 14, 4),
(2, 1, 10, 4), (3, 1, 11, 4), (1, 3, 25, 4), (3, 4, 35, 4),
(5, 0, 5, 4), (0, 2, 16, 4), (7, 3, 31, 4), (2, 5, 42, 4),
(5, 3, 29, 4), (0, 2, 16, 4), (1, 5, 41, 4), (6, 0, 6, 4),
(7, 3, 31, 4), (7, 5, 47, 4), (5, 0, 5, 4), (0, 3, 24, 4),
(2, 3, 26, 4), (0, 5, 40, 4), (2, 5, 42, 4), (0, 1, 8, 4),
(3, 5, 43, 4), (4, 4, 36, 4), (5, 3, 29, 4), (5, 5, 45, 4),
(1, 1, 9, 4), (7, 3, 31, 4), (7, 4, 39, 4), (1, 1, 9, 4)]
Representation of a single melody as a chromosome

B. Selection Schemes

Selection schemes are applied to choose the fittest parents and find the best solutions to survive in the next generation. The selection schemes used include binary tournament and truncation.

1) Binary Tournament: Binary tournament involves randomly selecting two chromosomes and choosing the fittest of the two according to a pre-determined selection probability. A random number is generated and if the number is lower than or equal to the selection probability, the fitter individual is chosen or else the weaker one is chosen. The binary tournament process is run for as many times as the number of chromosomes to be selected.

2) Truncation: Truncation selection chooses the top k fittest chromosomes, where k is the number of chromosomes that have to be selected. For each crossover, 2 fittest chromosomes are selected, while for survivor selection, k is equal to the population size to select chromosomes that will be carried over to the next generation.

C. Fitness Function

The fitness is evaluated based on several weighted attributes of music. The general formula for evaluating the fitness f is

$$f = \sum_{i=1}^{n} \lambda_i f_i$$

where f_i is the musical attribute being used, λ_i is the weight of the attribute and n is the total number of attributes used.

Following are the attributes we used:

1) Octave: A composition sounds more pleasant if the pitch values of its notes are relatively close to each other [2]. Pitch and octave are similar in that they both denote the frequency of the note. Hence, to make sure the pitches are close to each other, we favored chromosomes that have more beats with the same octave.

2) Interval:

3) Variation:

4) Down Beat:

D. Crossover

Crossover scheme for this algorithm involves choosing a random index in the range of the length of the melody. Each crossover leads to generate exactly 2 children. The first part of Offspring 1 is chosen from parent 1 and the second part is from parent 2. On the other hand, the first part of Offspring 2 is chosen from parent 2 and the second part is from parent 1.

E. Mutation

Mutation scheme for this algorithm mutates based on a mutation rate. A random number is generated and if it lies within the set mutation rate, the chromosome is mutated. The process will produce a new note wherever mutation rate is met.

F. Evolutionary Algorithm

Combining all techniques above, we evolved the population in search of finding the melody with the highest fitness. At the end of set n generations, the population is sorted with respect to fitness and the highest fitness melody is stored as the best melody. However, listening to the music allowed us to judge how good the melody is, based on how good it sounds to the human ear. Therefore, judging fitness was not only restricted to a number.

G. Generating music files using PySynth

We used a library ”PySynth” to produce music files. The chromosome structure is treated such that the melody is turned into a ”.wav” file which is playable by any music player.

V. Experiment and Results

We tested our algorithm with different sets of parameters. Controlling population size, mutation rate , selection schemes and number of generations, we tried to produce harmonious melodies. As expected, human interaction was necessary to identify good pieces of melody. At each moment that we made a change in parameters, we would stop and listen to the best melody generated at the end to listen and judge the melody.

VI. Conclusion

The study was limited by human interactions. We were bound to listen to the music besides simply trusting the fitness function. This however breaks the complete dependence on the computer in generating music and involves a human brain to pass judgment. A melodious tune may not be rightly identified by the genetic algorithm but we have gotten close to achieve a harmonious melody at the end of our study. Since we have built up on other works on this topic, we believe that a mathematician who has worked deeply on music’s relation to mathematics can help to find a better fitness function. There is a room for improvement in this search and we believe it is achievable to remove human dependence in the innovative and creative task of music composition.

Acknowledgment

What resources we have used and who has helped us

References

[1] D. Matic, ”A genetic algorithm for composing music”, *Yugoslav Journal of Operations Research*, vol. 20, no. 1, pp. 157-177, 2010. Available: 10.2298/yjor1001157m [Accessed 19 April 2019].

[2] V. Anderling, O. Andreasson, C. Olsson, S. Pavlov, C. Svensson, and J. Wikner, ”Generation of music through genetic algorithms,” 2014.

[3] Z. Geem and J. Choi, ”Music Composition Using Harmony Search Algorithm”, *Lecture Notes in Computer Science*, pp. 593-600. Available: 10.1007/978-3-540-71805-5_65 [Accessed 19 April 2019].

[4] A. Freitas, F. Guimares and R. Barbosa, ”Automatic Evaluation Methods in Evolutionary Music: An Example with Bossa Melodies”, *Lecture Notes in Computer Science*, pp. 458-467, 2012. Available: 10.1007/978-3-642-32964-7_46 [Accessed 19 April 2019].