University of Hasselt

Genetic music

Aytug Altin

text1
text2

July 16, 2021

Abstract

19 abstract **Keywords** Keyword-Keyword; Keyword; Keyword Keyword; Keyword-Keyword Keyword

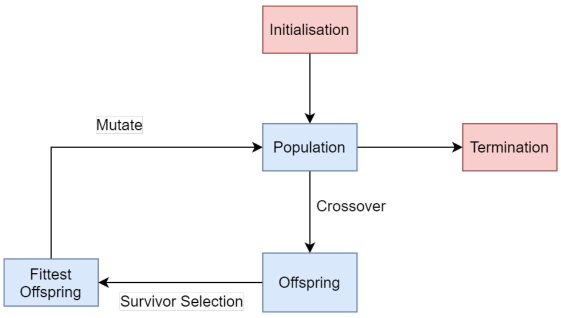

Figure 1: The genetic model, an overview of the important components.

1 Introduction

Evolutionary algorithms can be used for both optimization problems and modelling problems [1]. In both cases, we are looking for some input that creates a known or desired output. Modelling problems can be transformed to optimization problems where our search space is defined by all the potential models.

In the domain of music composition we lack a model that could judge the input, the fitness function, and our desired output may be ill-defined. In GenJam [3], the quality of the genetically produced solos are rated by human individuals. A human mentor gives "real-time feedback" which is used to derive a fitness score. Fitness can also be calculated by different aspects based on music theory as shown in [7], [2], [4]. By splitting the fitness function into subcategories, we are allowed to rate an individual song based on different aspects and set importance to certain preferred aspect. In [2], songs were compared to a well known group of songs. In [7], each bar of the song is rated by different criteria and summed up together to obtain the total fitness. The fitness is here calculated by the similarity between the to be rated individual and a reference individual or by reference values. Five fitness functions where used in [4], one per type of user preference. These preferences are: transition, repetition, variety, range and mood.

In this paper, we emphasized on the structure of the individual songs. Instead of focusing on theories that define music to be better than others, we focussed on a master song that defines the theory similar to [7]. The similarities between the population and the master can be based on the result of an absolute comparison i.e. comparing the exact number of notes of the candidate, which is the to be rated song, to the master. However, these absolute ratings would result in a population that is exactly the same as the master song, this is not what we are looking for. We introduce a new way to rate the candidates: relative ratings.

2 Model

In this section, each part of the model is discussed. On figure 1 a general overview of the model is illustrated. Before we go into detail, we briefly discuss the different components of our model.

The initial population can consist of both random songs or non-random songs, these are called $GEN0$. The next generation is calculated by applying a crossover function on $GEN0$ to obtain $GEN1$ which is the offspring. Each individual song in the offspring will be rated by the fitness function and obtains a score. Now we can select the best rated songs in $GEN1$ and mutate them. This process is repeated until a certain condition is met or until user is satisfied with the results and manually terminates the loop.

3 Initialization

First, the initial population needs to be determined, we call it $GEN0$ throughout this paper. $GEN0$ usually consist of a mix of different songs from different genres making the initial population diverse. The size of the population, which is static during the execution, is decided here and is related to the number of parents.

4 Recombination

In the recombination step, a crossover function is used to pair parents in order to produce a child. This child will belong to the next generation. The model performs a uniform crossover, meaning each gene will be considered separately when pairing the parents [1]. For each song, the notes, chords and rests are considered as the genes of a song. Two parents produce only one child by uniformly selecting genes from one of the two parents. On figure 2, an example of two songs, that are considered as parents, are graphically displayed. During the crossover process, the model iterates over all the genes in both parents and selects only one or the other to pass to the resulting child. Both genes have an equal chance of being selected. On figure 3 a sample child of the corresponding parents is illustrated.

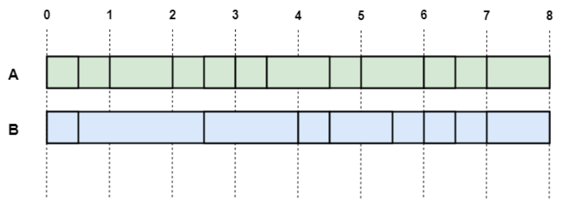

Figure 2: Graphical display of two parents A and B. Each rectangle represents a note, chord or rest with their corresponding length. The horizontal axis represents time.

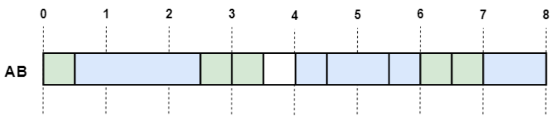

Figure 3: The child generated from the parents A and B.

Notice that there is a gap in the child song at time offset 3.5. This is the result of selecting parent B's gene at time offset 4 instead of the corresponding gene of parent A's at time offset 3.5.

At each recombination step, all the parents are paired with each other. If there are N parents, the number of children will be equal to $\frac{N*(N-1)}{2}$. After the recombination, the fitness of each child will be calculated.

5 The fitness function

A composition has multiple aspects that can be rated at each individual level, therefore, the fitness of an individual is determined by different **rating functions**. Each of these rating functions calculates a **score** for a particular **concept** of the song. The tendency to follow a musical scale within the composition is a concept for example. Every rating function calculates a score and to this score there is a predetermined weight attached that implies the importance of the rated concept. The total fitness of a song x is equal to the sum of the products of all rating functions S for each concept and their corresponding weights W.

$$TotalFitness(x) = \sum_{i=1}^{C} S_i * W_i$$

where C is the number of concepts.
A rating function calculates a score for a concept based on a reference. For the reference, an existing song is used. Throughout this paper, this reference song is called the **master song**. The goal is to generate songs that are similar to the master song. The master song defines the genre that the candidates have to follow. Each rating function R of a concept calculates a score based on the difference of the candidate song x to the master song m for that concept c.

$$R_{concept}(x) = difference(f_{concept}(x), f_{concept}(m))$$

With $f()$ a sub-function that gives a rating for the concept.
We introduce two ways to compare the master song with the candidate song: absolute and relative comparison. In the absolute comparison,

the elements of the master song are directly compared with the candidate song. The rating functions belonging to the absolute comparison make sure the candidate songs look similar to the master song. In the relative comparison, the structure of the master song is directly compared with the candidate song. Here the candidate songs will not exactly match the master.
In the following sections, every single rating function is described. For each rating function of a concept, We describe the concept first. Followed by the sub-function $f()$ that gives a rating for the concept. At last, the methodology on how the difference between the master and the candidate songs are calculated is described.

fix intro

5.1 The Rating Functions

intro

5.1.1 Zipf's law distance

[5] and [6] described that songs that followed Zipf's law, or were closer to it than other songs, were more likely to be preferred by users.
A Zipfian distribution is a distribution where the 2nd highest occurring element, occurs 1/2 times the number of occurrences of the highest occurring element, the 3rd highest occurring element, occurs 1/3 times the number of occurrences of the highest occurring element and so on. Suppose L is the number of occurrences of the most popular element. For every element E, with rank R (position of the element on the list of unique elements ordered by its frequency of appearance), the amount of occurrences is defined by the following function:

$$Occurences(E) = \frac{1}{R} * L$$

The tendency of the song's elements to follow the Zipf's law is the concept for this rating function. The song's elements can be both intervals or notes. This rating function can be used in two ways. The candidate song can be rated on its tendency to follow the Zipf's law. The candidate song can also be rated based on how similar this tendency is between the master song and the candidate song.

5.1.1.1 Pitches

Consider the distribution for the pitches that occur in the candidate song. With the same pitches, consider a Zipfian distribution that has the same pitches. Calculating the Wasserstein distance between these two distributions results in a metric that represents the tendency of a song to follow zipf's law. This distance can be normalized by the Wasserstein distance between the Zipfian distribution and a uniform distribution of the same pitches. The following equation illustrates how a this score can be calculated.

$$ZipfPitchScore(x) = \frac{WD(D_{song}, D_{zipf})}{WD(D_{uniform}, D_{zipf})}$$

with $WD()$ the Wasserstein distance between two distributions, D_{song} is the pitch distribution of the candidate song, D_{zipf} is the zipfian distribution of the pitches of the candidate song and $D_{uniform}$ the uniform distribution of the pitches of the candidate song.

5.1.1.2 Intervals

The same can be done for the intervals between notes of the song, which results as the following equation:

$$ZipfIntervalScore(x) = \frac{WD(D_{song}, D_{zipf})}{WD(D_{uniform}, D_{zipf})}$$

5.1.1.3 Usage

Both these sub-functions (ZipfIntervalScore() and ZipfPitchScore()) are used to create two rating functions. Since these two rating functions can be used independently there is no need to compare them to the master. Here the sub-function is equal to the rating function.

5.1.2 Neighbor pitch

This rating function calculates a score based on the amount of wrong or unpleasant intervals. Unpleasant intervals are defined by the master song. The master song defines the lower and the upper bound of the intervals between notes. The neighbor pitch rating function calculates a score based on the number of occurrences of these unpleasant intervals. The following rating function emerges:

$$NeighborPitchScore(x) = \frac{I_{unpleasant}}{I_{total}}$$

with $I_{unpleasant}$ the number of unpleasant intervals and I_{total} the total number of intervals.

5.1.3 Melody direction

The direction of a song can be represented as a number between 0 and 1. If this number is higher than 0.5, the direction is of the melody is going upwards, otherwise it is going downwards. This score is calculated by comparing the amount of upwards intervals (positive semitones) to all the intervals. The melody direction song x is calculated with the following subfunction:

$$MelodyDirection(x) = \frac{I_{upwards}}{I_{total}}$$

with $I_{upwards}$ the number of upwards intervals and I_{total} the total number of intervals. The rating function for the melody direction calculates a score by comparing the direction of the candidate song x and the master song m as followed:

$$MelodyDirectionScore(x) = abs(MelodyDirection(x) - MelodyDirection(m))$$

with $abs()$ as the absolute value function.

5.1.4 Direction stability

This rating function calculates a score based on the number of the times the direction of the melody changes. A direction change occurs when the direction of the current interval is not equal to the previous. The direction stability score of a song x is defined by the following rating function:

$$DirectionStability(x) = \frac{I_{change}}{I_{total}}$$

with I_{change} the number of direction changes and I_{total} the total number of intervals. The rating function for the direction stability calculates a score by comparing the direction stability of the candidate song x and the master song m as followed:

$$DirectionStabilityScore(x) = abs(DirectionStability(x) - DirectionStability(m))$$

5.1.5 Unique pitches

This rating functions gives a score based on the number of unique pitches of the song. This results in the following rating functions

$$UniquePitches(x) = \frac{Number\ of\ uniques\ pitches}{Total\ number\ of\ pitches}$$

Here, the rating function is similar to the previous ones:

$$UniquePitchesScore(x) = abs(UniquePitches(x) - UniquePitches(m))$$

6 Survivor selection

After

Materials & Methods

man

Results

results

Discussion

Discussion

Conclusions

concluded

Acknowledgements

We would like to extend our sincerest gratitude and appreciation to everyone who helped make this project a possibility.Acknowledgements Acknowledgements Acknowledgements Acknowledgements Acknowledgements Acknowledgements Acknowledgements Acknowledgements

References

[1] James E. Smith A. E. Eiben. *Introduction to Evolutionary Computing.* Mairdumont Gmbh & Co. Kg, 2003.

[2] Viktor. Anderling, Olle. Andreasson, Christoffer. Olsson, Sean. Pavlov, Christian. Svensson, and Johannes Wikner. Generation of music through genetic algorithms. Master's thesis, Chalmers University of Technology University of Gothenburg Department of Computer Science and Engineering, Gothenburg, Sweden, 6 2014.

[3] John Biles. Genjam: A genetic algorithm for generating jazz solos. 07 1994.

[4] Adil Haleem Khan. Artificial intelligence approaches to music composition. Northern Kentucky University, Highland Heights, KY 41099, 2013.

[5] Bill Manaris, Juan Romero, Penousal Machado, Dwight Krehbiel, Timothy Hirzel, Walter Pharr, and Robert Davis. Zipf's law, music classification, and aesthetics. *Computer Music Journal*, 29:55–69, 03 2005.

[6] Bill Manaris, Patrick Roos, Penousal Machado, Dwight Krehbiel, Luca Pellicoro, and Juan Romero. A corpus-based hybrid approach to music analysis and composition. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 839. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[7] Dragan Matić. A genetic algorithm for composing music. *Yugoslav Journal of Operations Research*, 20, 01 2010.