University of Hasselt

Genetic music

Aytug Altin

text1
text2

July 27, 2021

Abstract

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

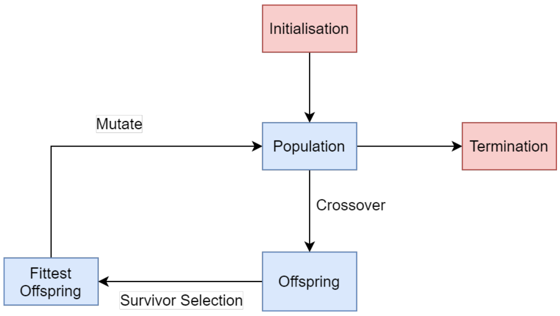

Figure 1: The genetic model, an overview of the important components.

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 ran-

dom 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.

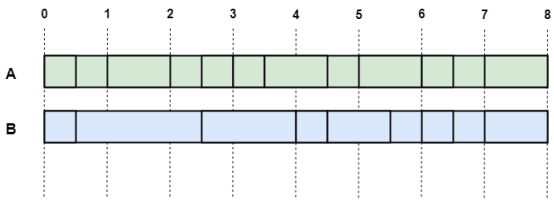

Figure 2: Graphical display of two parents A and B. Each rectangle represents a note, chord or rest with their corresponding length. The horizontal axis represents time.

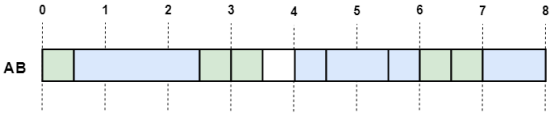

Figure 3: The child generated from the parents A and B.

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 direc-

tion 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:

$$UniquePitchesScore(x) = abs($$

$$UniquePitches(x) - UniquePitches(m))$$

5.1.6 Measure representations and relations

Every measure in a song can be represented at different levels. In this paper, the following levels are considered for the representation of a measure:

- pitch;
- type;
- duration;
- offset;
- combination of pitch, type, duration and offset;
- semitone.

The following is an example of a measure consisting of 6 elements represented at these six levels.

The **type** representation consist of the types of the measure's elements (note as N, chord as X and rest as R) in chronological order:

$$[X, X, N, X, X, N]$$

The **pitch** representation consist of the corresponding pitch(es) of the musical elements in the chronology they occur in the measure:

$$[G\sharp 3G\sharp 4, D3B2, C3, C5G4E\flat 4, E\flat 4G4C5, B4]$$

The **duration** representation is a list of the durations of the musical elements in the chronology they occur in the measure:

$$[half, quarter, eighth, eighth, eighth, eighth]$$

The **offset** representation is a list of the start time of every element that occurs in the measure. This list is also chronologically ordered:

$$[0.0, 1.0, 2.0, 2.5, 3.0, 3.5]$$

The **combine** representation is a combination of the types, pitches, durations and offsets representations:

$$[XG\sharp 3G\sharp 4h0.0, XD3B2q1.0, NC3e2.0,$$

$$XC5G4E\flat 4e2.5, XE\flat 4G4C5e3.0, NB4e3.5]$$

The **semitone** representation is a list of the semitones values of the intervals between the notes and chords.

$$[-6, -2, 24, -9, 8]$$

Note that rests will be skipped. Measures relations are defined by their similarity. This similarity is called the measure match rate. The

match rate is calculated by the Levenshtein distance [8] to get a ratio between 0 and 1 that represents the similarity between two measures. On figure 4, the 9th and 10th measure of the Godfather theme song are portrayed. The following is an example of match rates of the two measures.

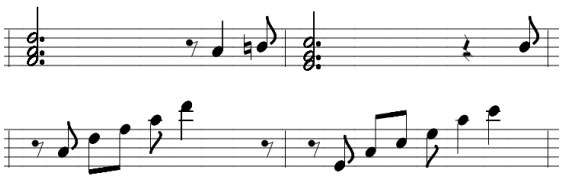

Figure 4: The 9th and 10th measure of the Godfather theme song.

TYPES

$A : [X, N, N, N, N, N, N, N]$

$B : [X, N, N, N, N, N, N, N]$

types match rate : 100%

PITCHES

$A : [E\flat 4 C5 G4, G2, C3, E\flat 3, G3, C4, E\flat 4, B\flat 4]$

$B : [D5 G\sharp 4 F4, C3, F3, G\sharp 3, C4, F4, G\sharp 4, B4]$

pitches match rate : 74%

DURATIONS

$A : [complex, eighth, eighth, eighth, eighth, quarter, quarter, eighth]$

$B : [complex, eighth, eighth, eighth, eighth, quarter, quarter, eighth]$

durations match rate : 100%

OFFSETS

$A : [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5]$

$B : [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5]$

offsets match rate : 100%

SEMITONES

$A : [-20, 5, 3, 4, 5, 3, 7]$

$B : [-26, 5, 3, 4, 5, 3, 3]$

semitones match rate : 91%

COMBINED

$A : [XE\flat 4 C5 G4 c0.0, NG2 e0.5, NC3 e1.0, NE\flat 3 e1.5, NG3 e2.0, NC4 q2.5, NE\flat 4 q3.0, NB\flat 4 e3.5]$

$B : [XD5 G\sharp 4 F4 c0.0, NC3 e0.5, NF3 e1.0, NG\sharp 3 e1.5, NC4 e2.0, NF4 q2.5, NG\sharp 4 q3.0, NB4 e3.5]$

combined match rate : 85%

The match rate defines the type of binding between two measures. Here, 4 types are defined: strong bindings, normal bindings, weak bindings, garbage bindings. These terms will be used further in this paper. A strong binding is of a stronger type than the normal binding, a normal binding is of a stronger type than the weak binding and a weak binding is of a stronger type than the garbage binding. Here, the criteria for each type of binding is defined. These criteria are determined subjectively and by intuition. As illustration, the first 16 measures of the Godfather theme song is used.
Strong bindings are bindings where repetition is very likely to be noticeable between the two measures. There exist a strong binding between two measures the following two rules are true.

1. At least three of the following criteria are true.
   (a) types match rate $> 80\%$
   (b) pitches match rate $> 80\%$
   (c) offsets match rate $> 85\%$
   (d) durations match rate $> 85\%$
   (e) semitones match rate $> 80\%$
2. (semitones match rate + combine match rate) / $2 \geq 80$ (= threshold)

Strong bindings of the godfather theme song are represented on a graph shown on figure 5. Each node is a measure marked with its position, each edge represents the matching rate between the measures. The measures that are not strongly bonded have no connection with each other. The graph representation has been used to determine the rules to classify the type of the relation between two measures.

Normal bindings are bindings where the repetitiveness of the two measures is less likely to be noticeable compared to strong between both measures. For two measures to be normally bonded they would have the same rules with different thresholds:

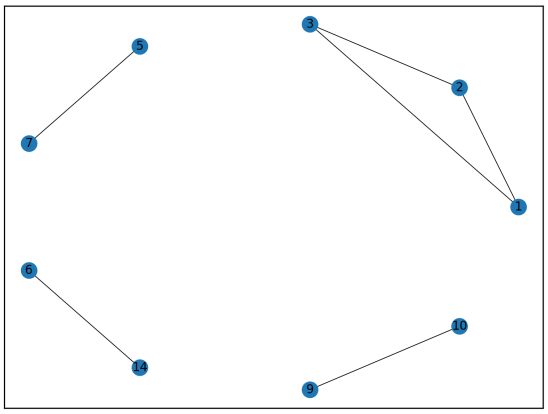

Figure 5: Graph representing strong bindings between measures in the Godfather theme song

1. At least three of the following criteria are true.
   (a) types match rate > 70%
   (b) pitches match rate > 70%
   (c) offsets match rate > 75%
   (d) durations match rate > 75%
   (e) semitones match rate > 70%
2. (semitones match rate + combine match rate) / 2 ≥ 70 (= threshold)

On figure 6, the normal bindings are visualized for the godfather theme song.

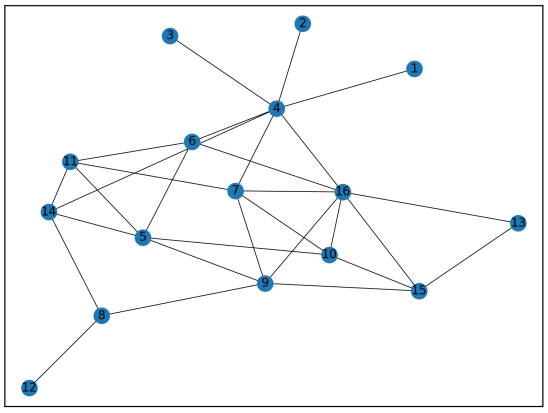

Figure 6: Graph representing normal bindings between measures: each node is a measure with its number inside, each edge represents the strength of matching rate between them.

As seen on the graph, there are more nodes and more edges on the normal bindings graph relative to the strong bindings graph on figure. This is interpretted as the structure of this song.

Furthermore, there are two binding types to bed discussed: weak bindings and garbage bindings. The weak bindings have as rules:

1. At least three of the following criteria are true.
   (a) types match rate > 55%
   (b) pitches match rate > 55%
   (c) offsets match rate > 60%
   (d) durations match rate > 60%
   (e) semitones match rate > 55%
2. (semitones match rate + combine match rate) / 2 ≥ 55 (= threshold)

The garbage has no rules They represent everything that is left over.

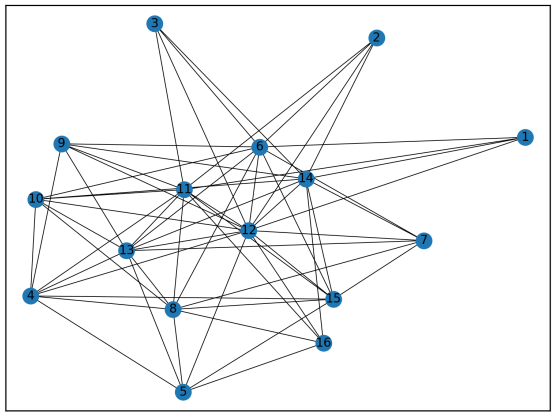

Figure 7: Graph representing weak bindings between measures: each node is a measure with its number inside, each edge represents the strength of matching rate between them.

On figure 7, it is shown that a lot of measures are weakly bonded with each other. These bindings have little resemblance with each other. On this graph, we see that all the measures have weak bindings with each other. This is a characteristic of the song that can be captured with these measure binding graphs. It indicates the resemblance of the measures across the song.

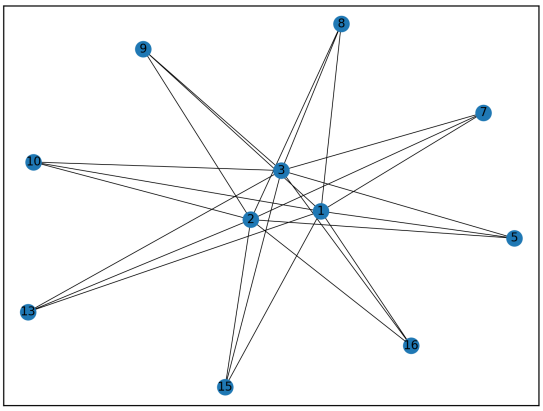

Figure 8: Graph representing garbage bindings between measures: each node is a measure with its number inside, each edge represents the strength of matching rate between them.

On figure 8, it is visible that 11 out of the 16 measures have garbage bindings with other measures and 5 measures have some level of bindings. This is a way to analyze a song and to compare it with other songs if they meet the same condition. This allows our model to look how the measures are bonded to each other and if it is possible to have measures that have a lot or have a few garbage bindings. The relationship between measures captures characteristic of a song that can be compared with other songs and from this comparison a score can be calculated that represents the distance between the two compared songs.

5.1.6.1 Measure bindings rate

The measure bindings rating function calculates a score based on the amount of strong, normal, weak and garbage bindings between measures. The concept that is being rated is the correctness of the rates in which the bindings of measures occur. Notice that the order does not have an effect on the score. For M measures there are $\frac{M(M-1)}{2}$ bindings, if s measures are strong bindings, the strong bindings rate would be equal to $\frac{2s}{M(M-1)}$. The following sub-functions are used calculate a score for this rating function:

$$StrongBindingsRate(x) = \frac{2s}{M(M-1)}$$

with s the number of strong bindings and x a song;

$$NormalBindingsRate(x) = \frac{2n}{M(M-1)}$$

with n the number of normal bindings and x a song;

$$WeakBindingsRate(x) = \frac{2w}{M(M-1)}$$

with w the number of weak bindings and x a song;

$$GarbageBindingsRate(x) = \frac{2g}{M(M-1)}$$

with g the number of garbage bindings and x a song.

The rating function will use the sub-functions to compare the measure bindings rates to of a master song m to the candidate song x. The rating function is defined as followed:

$$MeasureBindingsScore(x) = abs(StrongBindingsRate(x), StrongBindingsRate(m)) + abs(NormalBindingsRate(x), NormalBindingsRate(m)) + abs(WeakBindingsRate(x), WeakBindingsRate(m)) + abs(GarbageBindingsRate(x), GarbageBindingsRate(m))$$

5.1.6.2 Measures types rate

This rating is based on the measure itself instead of the bindings. Every measure can be categorized by its best possible measure binding. On figure 5, the measures 1,2,3,5,7,6,9,10 and 14 have at least one strong binding. The strong measure rate of a song x is equal to the number of measures with its strongest binding type strong s divided by the number of measures M:

$$StrongMeasureRate(x) = \frac{s}{M}$$

In this example, there are 9 measures that have as their strongest binding a strong binding type thus $StrongMeasureRate$ of the Godfather song is $\frac{9}{16} = 0.5625$.

Similarly, the followind sub-functions are defined for the other types of the measures. The normal measure rate of a song x is equal to the number of measures with as strongest bindings type normal n divided by the number of measures M:

$$NormalMeasureRate(x) = \frac{n}{M}$$

In the Godfather theme song, there are 7 measures out of 16 measures that have as

their strongest binding a normal binding, thus $NormalMeasureRate$ of the Godfather song is $\frac{7}{16} = 0.4375$.

The weak measure rate of a song x is equal to the number of measures with as strongest bindings type weak w divided by the number of measures M:

$$WeakMeasureRate(x) = \frac{w}{M}$$

In the Godfather theme song, there are 0 measures that have as their strongest binding a weak binding thus $WeakMeasureRate = 0.0$.

The garbage measure rate of a song x is equal to the number of measures with as strongest bindings type garbage g divided by the number of measures M:

$$GarbageMeasureRate(x) = \frac{g}{M}$$

In the Godfather theme song, there are 0 measures that have as their strongest binding a garbage binding thus $GarbageMeasureRate = 0.0$.

Having a $WeakMeasureRate = 0.0$ and a $GarbageMeasureRate = 0.0$ means that all the measures have some sort of similarity with each other. There cannot be an outcast of a measure that has nothing in common with any of the measures, not according to the Godfather theme song.

The rating function will use these sub-functions to compare the measure types rates of a master song m to the candidate song x. The rating function is defined as followed:

$$\begin{aligned} &MeasureTypesScore(x) = \\ &abs(StrongTypeRate(x) - StrongTypeRate(m)) \\ &+abs(NormalTypeRate(x) - NormalTypeRate(m)) \\ &+abs(WeakTypeRate(x) - WeakTypeRate(m)) \\ &+abs(GarbageTypeRate(x) - GarbageTypeRate(m)) \end{aligned}$$

5.2 Measure structure and patterns

In this section, multiple rating functions are discussed that compare the structure of the measures of a master song m with a candidate song x. For every measure of a song, its similarity to the other measures is calculated. For example: for an 8 measure long song, the first measure's type similarity with the other measures can look like the following:

$$[77, 85, 89, 85, 85, 77, 83]$$

Notice that the size of the list is 7 since the first measure is not compare to itself. The first element in the list 77 is a rate that represents the similarity between our first measure with the second, the second element in the list 85 is a rate that represents the similarity between our first measure with the third and so on. Notice that this list only represents the relations of measure one with the others and not all measures with each other.

For the master song Godfather theme song, the first measure's type similarity with the other measures will look like the following:

$$[100, 100, 77, 68, 73, 68, 80]$$

By using the Wasserstein distance to compare these two lists, the order of the list will not be taken into account. Only the quality of the relations will be rated. Since the Wasserstein distance gives an absolute value, it needs to be normalized. The normalization factor is calculated by the distance of the master song's list to an artificial list that is the furthest away from the master. In our example, the following list will be used:

$$[0, 0, 0, 0, 0, 0, 0]$$

The maximum distance of the first element 100 is 0, the maximum distance of the second element 100 is 0, the maximum distance of the third element 77 is 0 and so on. This is called the complement relation for a measure.

For a candidate song x with M measures where x_i is the ith measure's type relations list of x, for a master song m with M measures where m_i is the ith measure's type relations list of m and z_i the complement relation of m_i, the rating function that calculates a score based on the structure of the types is

$$\frac{1}{M} \sum_{i=1}^{M} \frac{WD(\ x_i, m_i)}{WD(\ z_i,\ m_i)}$$

Alongside the types relations, the offsets, pitches, durations and semitones relations can also be calculated resulting in a total of 5 rating functions.

5.2.1 Type distribution rating

This fitness function compares the occurrence rates of the different types of elements and compares them with the occurrence rate of the master. The sub-functions are used:

$$Rate_{Notes}(x) = \frac{Number\ of\ notes\ in\ x}{Total\ number\ of\ elements\ in\ x}$$

$$Rate_{Chords}(x) = \frac{Number\ of\ chords\ in\ x}{Total\ number\ of\ elements\ in\ x}$$

$$Rate_{Rests}(x) = \frac{Number\ of\ rests\ in\ x}{Total\ number\ of\ elements\ in\ x}$$

The type distribution score will be calculated with the following rating function. The type distribution rating of a song x with master m is calculated by the following formula:

$$TypeDistributionScore(x) =$$
$$abs(Rate_{Notes}((x)-$$
$$Rate_{Notes}((m)$$
$$+abs(Rate_{Chords}(x)-$$
$$Rate_{Chords}(m))$$
$$+abs(Rate_{Rests}(x)-$$
$$Rate_{Rests}(m))$$

5.2.2 Absolute rating functions

The following rating functions compare the master song directyl with the candidate song. These rating functions do not compare the values calculated within the songs, here they calculate this value by directly comparing the songs with each other. For example, instead of comparing the values received from comparing elements from the same song with the master, the absolute rating functions directly compare elements from the song to the master.

5.2.2.1 Element count rating

This rating function calculates a score based on the number of elements that appear in a song compared to the master song. This rating function is calculated by the following equation:

$$\frac{abs(E_x - E_m)}{E_x}$$

with E_x and E_m the number of elements of the candidate song x and master song m respectively.

5.2.2.2 Absolute rhythm rating

This rating functions compares the rhythm of the candidate song x with the master song m. A rhythm list is created by combining durations and the offsets representations of the song. For example the following rhythm string list:

$$['0.0whole','0.5eighth','1.0quarter','1.5quarter',$$
$$'1.7516th', ..., '16.5quarter','16.7516th']$$

The Absolute rhythm matching score is calculated by using the Levenshtein distance to match the string rhythm string list of the candidate song x with the rhythm string list of the master song m.

5.2.2.3 Absolute types rating

This rating functions uses the string list that consists out of the types (notes, chords and rests) that occur in the song in chronological order, for example:

$$['N','N','N','N','N','N','N','X', ...'N']$$

The absolute types rating is calculated by using the Levenshtein distance between this types string list of the candidate song x with the types string list of the master song g.

5.2.2.4 Absolute interval distance rating

This rating function uses the Wasserstein distance when comparing the semitones of the intervals. By creating a list of interval semitone values, the Wasserstein distance can be calculated between the list of the candidate song x and the master song m. Such a list would look like the following:

$$[7, -7, 5, -5, 7, ..., 5, 2, 3, -12, 9, 3, 2, -5, -2, 7]$$

The order in which these intervals occur does not effect the Wasserstein metric. For the normalisation factor, the Wasserstein distance between the master song and the following list is used:

$$[0]$$

The Interval Distance Rating between a song x and a master song is:

$$IntervalDistance(x) =$$
$$\frac{WD(Distribution\ of\ song's\ intervals, Distribution\ of\ masters's\ inte}{WD([0], Distribution\ of\ masters's\ intervals)}$$

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.

[8] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press, 2009.