

text1
text2

Abstract

```
graph TD; Init[Initialisation] --> Pop[Population]; Pop --> Term[Termination]; Pop --> Off[Offspring]; Off -- "Survivor Selection" --> FFO[Fittest Offspring]; FFO -- "Mutate" --> Pop;
```

The flowchart illustrates the Genetic Algorithm process. It begins with 'Initialisation' (red box), which leads to 'Population' (blue box). From 'Population', the process can lead to 'Termination' (red box) or 'Offspring' (blue box). 'Offspring' leads to 'Fittest Offspring' (blue box) via 'Survivor Selection'. 'Fittest Offspring' then leads back to 'Population' via 'Mutate'. 'Population' also leads to 'Offspring' via 'Crossover'.

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

1 Introduction

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 [5]. 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

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.

2.1 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.

2.2 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.

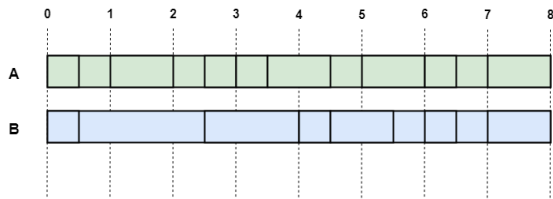


Figure 2: Graphical display of two parents A and B. Each rectangle represents a note, chord or rest with their corresponding length. The horizontal axis represents time.

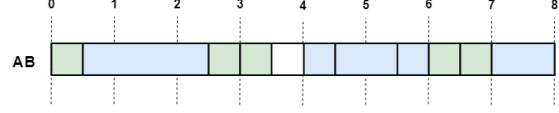


Figure 3: The child generated from the parents A and B.

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.

2.3 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 gives a certain **score** for a particular **concept** of the song. We clarify with an example: the tendency to follow a musical scale within the composition can be such a concept. 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.

$$S(x) = difference(f(x), optimal score) * S_{weight}$$

2.3.1 The master song

The fitness function calculates a score song based on the master. The master song is set during the initialisation process and it controls the population by defining the rules. We can rate candidate songs based on the master song in two ways:

- absolute comparison: this is where we compare the elements of the master song directly with the candidate song.
- relative comparison: this is where we compare the relative structure of the master song directly with the candidate song.

After

man

results

Discussion

concluded

We would like to extend our sincerest gratitude
and appreciation to everyone who helped make
this project a possibility.Acknowledgements Acknow-
ledgements Acknowledgements Acknowl-
edgements Acknowledgements Acknowledge-
ments Acknowledgements Acknowledgements

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