How to run the program

This software was created using Python 3.8 with two additional libraries: music 21 and mido. The first one allows to determine a song’s tonality, the second one parses midi format (for input files)

You would notice two limitations at the program's beginning where you may modify the names of the input and created output files.

The average number of algorithm iterations required to get a suitable version is around 400. The generation takes between three and five seconds to finish.

Algorithm description

After determining the song's tone, we must determine the accompaniment's size. We want to play a chord for each semi-tact in our situation. Then, we append the best-generated accompaniment from the most recent generation to the original MIDI file using our method. We must establish some restrictions and the genetic algorithm's parameters in order to do this.

Chromosomes

We must define our individuals before we can build the algorithm. In this instance, the chromosome is the chord-based accompaniment.

The accompaniments are random chords for the first population. By utilizing a random number to represent the primary note and a predetermined pattern to the chord, we may generate such a random chord (major, minor, sus2, sus4, major-inverse, minor-inverse, and double-inverse). An array of numbers that reflect the indents from the main note is used to define this pattern.

We employ 128 chromosomes, which is a rather small amount, for the first generation. We choose this number because experiments have shown that, although a larger number of chromosomes may result in the algorithm being produced in fewer generations, the effort involved in creating those generations will be slower.

Fitness function

We need a decent fitness function to rate our accompaniments in order to discover the optimal sound algorithm. In this instance, the circle of fifth is used to evaluate the accompaniments. First, we give it the lowest rating possible by assigning 1 point to each chord, and then we reward each chord that fits by lowering this rating.

To do this, we compare each chord of the generated accompaniment to previously calculated consonant chords for the song. Therefore, we reward the chromosome by raising its rank by 0.5 if the generated chord is in the song's consonant chords.

Then, if it is in consonant chords, we determine if the music is pausing at this point. If so, we additionally lower the rating by 0.5 as any chord from the consonant array would fit it. However, if the original MIDI contains a note in this semi-tact, we check to see if it is in the chord and deduct 0.5 points from the rating if it is. The note is fine if it isn't in a chord or consonant chords, which implies that any consonant chord would be fine. However, we also lower the rating by 0.5 in this case.

In other instances, we leave the accompaniment's rating alone. After that, we rank order the population. The accompaniments that received the lowest ratings in this case are the most appropriate ones.

Mutation

We choose a predetermined number of chromosomes from the population, and we substitute a random chord generated in the same manner as the initial population for one of each accompaniment's genes. In our code, 50% of the chromosomes are altered, but we never alter more than one chord at a time. It enables us to get the best outcome when only one or two chords in the accompaniment need to be changed, as mutating more genes could lead to less desirable results.

Selection

By ranking the population and choosing the chromosomes with the lowest ratings, our program chooses 25% of the best-rated chromosomes. After choosing them, we transfer them to the new array,

and then we use crossover for these survivors to produce the new population.

Crossover

In our case, we use one-point crossover, wherein two parents' genes are switched for two children at a random location at one point.

Break Criteria

Our population's chromosomes are rated, and we would prefer to halt our program in the following two circumstances:

1. The chromosome receives a grade of 0. Since the rating at the beginning is equal to the number of chords, each fitted chord lowers the rating by one, a rating of zero indicates that all chords in the accompaniment are appropriate.

2. More iterations of the population have been run than have been stopped. This number, in our situation, equals 5000. If there are too many populations, we choose at least one less-than-ideal solution.