Music Generation Project 1 Report

Arseny Savchenko

Innopolis University

3 July, 2023

I Introduction

This report aimed to provide an overview of the project, analyze strong points and what can be further improved in the current state of the development. The goal of the project is to generate MIDI files, containing both melody and accompaniment, with the algorithms that do not depend on the results of human work.

Generated .mid files, as well as the source code, can be found on my GitHub. To build and run the project, Rust compiler is required.

II Algorithms

Project uses multiple algorithms to generate melodies. All of them in some sense helped to come up with the next ones, so they are worth to be mentioned. Previous algorithms and strategies, that were implemented and used, can still be found in the source code, history of commits may include additional examples of each algorithm that was used.

A. Previous ideas

Markov chains and analysis of the MIDI files: First algorithm is related to the Markov chains and MIDI parsers. Next Rust crate was chosen to read existing MIDI files and produce new ones. Implementation of this algorithm can be found here. Markov chains based analyzer (constructs map of probabilities) is located here. MIDI parsing is here. Algorithm was abandoned, because it was hard to distinguish melody notes from chord notes, so the parsed results were too unstable. However, it can be further used in the next projects.

Markov chains and generation of notes' parameters: Idea of the previous method showed that relation on a dataset of MIDI files was not acceptable. However, the idea of Markov chains was further integrated to control randomization. Tonic note was generated with random parameters (length, velocity and delay) from the generated key. Then, 3 types of events were chosen: 1) Increase (length, velocity or pitch) 2) Decrease 3) Repeat. Events were produced with the random sequence of digits of PI number. If event was chosen by the digit, then probabilities of other 2 events were increased. Algorithm was deprecated because

it was producing too unpredictable melodies.

Even and odd melodies: The main problem of first two methods is uncontrolled randomization: even if notes are on scale, they still placed too randomly. To control this randomization and not limit it completely, I've decided to arpeggiate melodies with the even number of notes. For odd melodies - create melodies with length 3, based on the previous idea, then arpeggiate other two even parts and shuffle all 3 of them. Implementation can be found here. Idea was abandoned because melody's rhythm was too unstable.

B. Current algorithm

Current algorithm is still based on the idea of arpeggiation, however, it limits the randomization of tempo to produce something acceptable in 90% of tries. Algorithm separates bar on 16 parts, creates a tonic note with a zero delay and a length equal to 1/16 of bar. Then it travels across all 16 parts of the bar and tries to put a note with the same length that is located closely to the current one (difference in pitch is not greater than 2 notes in scale). Algorithm limits the distance between two notes: only the single delay 3/16 is allowed, delays greater than 3/16 are not allowed. Implementation can be found here.

Generated lead helps to construct the harmony. All generated melodies in the project written in Melodic Minor scale with Aeolian mode, so specific chords can be chosen. Algorithm travels through the generated lead and either puts new chord or extends last one with the length and the delay of the current note. Implementation can be found here. Finally, algorithm composes notes by converting data types to MIDI messages.

III Analysis of the result. Further improvements

Even though algorithm produces listenable results (and sometimes even good ones) and most generated melodies are unique, results are limited with the rhythm. To improve the result and extend it with other ideas, first of all, other scales and modes can be chosen. Besides, algorithm highly relates on the arpeggiation, producing a lot of notes in the single bar. So, other patterns to generate tempo can be chosen. Finally, algorithm to create chord progressions can be improved and extended, as it is very limited right now.