Resurrection

Dirk Schlingmann

Math/CS Division, University of South Carolina Upstate; schlingm@uscupstate.edu

Abstract

In this paper, I will investigate how mathematics and computing can be helpful in creating new interesting algorithmic music that is based on the music data of great composers.

Introduction

Computers, the Internet, and technology have advanced the way we learn and create. Playing sounds, displaying graphics, viewing moving images, and programming solutions have enhanced conceptual understanding and creative endeavors. With computing power, we can tackle real data and real problems. In this paper, I try to show how algorithmic music programming can provide musical solutions that are based on mathematical concepts and that expand our ideas of music.

Algorithmic Music

Music is an artistic composition of sounds. We often use instruments to create musical sounds, which are represented by notes in a musical score. If we want different instruments to collaborate on a piece of music, the instruments need to be designed in such a way that they produce sounds that are pleasing together. During the evolution of music, humans have recognized that certain ratios of frequencies of tones (pitches) produce favorable consonant sounds (chords). If the pitches (frequencies) adhere to a rigid common musical tone or tuning system, a whole orchestra with many different instruments and many different notes interacting with each other can play beautiful music together. Such a system is the 12-tone equal temperament, which is widely used in the Western world of today. With the introduction of computers and synthesizers, a Musical Instrument Digital Interface (MIDI) was engineered for this 12-tone equal temperament. MIDI has enhanced music and facilitated communication between MIDI devices. For a more detailed introduction to MIDI, please consult the information at reference [2].

In terms of mathematics, a music score is a list of notes, which are the instructions on how to perform the musical piece. A note is determined by its musical elements, which are its time-measure (the time measured in seconds when the note is turned on and the time when it is turned off), its pitch (frequency), the instrument (timbre), and the volume (dynamics or loudness). Each music note in a song can be described using the following notation or data structure: note = {{time-on, time-off}, pitch, instrument, volume}. In music, we use the term *duration* for the difference between time-off and time-on (duration = time-off – time-on). A song is now the set {note[1], note[2], …, note[n]} of finitely many notes, where note[i] = {{time-on[i], time-off[i]}, pitch[i], instrument[i], volume[i]}. This set of notes is stored in a MIDI file.

song = {
{{time-on[1], time-off[1]}, pitch[1], instrument[1], volume[1]},
...,
{{time-on[n], time-off[n]}, pitch[n], instrument[n], volume[n]}
}.

In mathematics, we prefer numbers instead of music notation. For example, a standard grand piano has 88 keys (A0 through C8 in music notation) with corresponding MIDI numbers 21 through 108. Please note that the piano has the MIDI number 1, and the volume is a number between 0.0 and 1.0, where 0.0 is

silence and 1.0 is maximum volume. For a more detailed list of MIDI instruments and their MIDI numbers, please consult reference [3]. Using MIDI numbers, here are the first two and last two notes of the 1637 notes of Beethoven's *Adagio* from his famous *Pathetique* sonata.

```
adagio =
{
    {{0.0, 1.07083}, 56, 1, 0.27451}, {{0.0, 2.14375}, 44, 1, 0.321569},
    …,
    {{292.038, 297.494}, 48, 1, 0.196078}, {{292.038, 297.494}, 56, 1, 0.196078}
}.
```

If we look more closely at the above MIDI data, we realize that a computer and a computer program might be helpful or even much more efficient in creating a set of individual notes and its associated music. The parameters for pitch, time-on, time-off, instrument, and volume can originate from mathematical objects, abstract concepts, or any appropriate data coming from almost anything imaginable. A computer algorithm (a carefully written computer program) can produce music that goes beyond traditional music. Music created by such computer algorithms is called algorithmic music. Algorithmic music is not limited to the MIDI environment. We can program our own unique sounds of any frequency and any timbre. Algorithmic music opens a door into a new world of music (references [4], [5], and [6]).

Resurrecting Beethoven

Mathematics, computing, and algorithmic music can also assist us in answering questions like "Can we create new interesting music that is based on the music data of great composers, for example, Beethoven's piano sonatas?" Here is one of my attempts to answer this question.

The idea behind my attempt is to use Beethoven's most popular time-ons, most popular durations, most popular pitches, and most popular volumes in certain time intervals. The instrument will be the piano. To learn about Beethoven, we need to study and analyze his work. For example, Beethoven wrote 32 piano sonatas. To create algorithmic music based on Beethoven's piano sonatas, we collect, arrange, and sort the MIDI data of all Beethoven piano sonatas. A complete list of MIDI files for all 32 piano sonatas can be found at *Kunst der Fuge* (reference [1]). First, we make sure that all sonatas start at time 0.0. To create algorithmic music that is more appealing, we transpose all Beethoven sonatas into the same key, for example C Major. Next, we put the notes of each individual piano sonata into a rigid order following these two rules. Notes with an earlier time-on are listed before notes with a later time-on. If notes share the same time-on value, then notes with a lower pitch are listed before notes with a higher pitch. Then we create a spreadsheet-like layout of rows and columns where each row represents all the ordered MIDI notes of a piano sonata. If we introduce the notation, note[i, j] = {{t-on[i, j], t-off[i, j], p[i, j], instr[i, j], v[i, j]} to determine the j-th note of piano sonata i, then the described spreadsheet is presented in Table 1.

Table 1: *Organization of the Notes of Beethoven's 32 Piano Sonatas (after transposition into C Major)*

	1st note	2nd note	…
sonata 1	{{0.0, 0.165593}, 52, 1, 0.266667}	{{0.256417, 0.420606, 57, 1, 0.282353}	…
sonata 2	{{0.0, 0.110294}, 53, 1, 0.470588}	{0.0, 0.110294}, 65, 1, 0.470588}	…
…	…	…	…
sonata i	{{t-on[i, 1], t-off[i, 1}, p[i, 1], 1, v[i, 1]}	{{t-on[i, 2], t-off[i, 2}, p[i, 2], 1, v[i, 2]}	…
…	…	…	…
sonata 32	{{0.0, 0.184606}, 48, 1, 0.737255}	{0.0, 0.184606}, 60, 1, 0.737255}	…

To create new algorithmic music based on Beethoven's most popular time-ons, most popular durations, most popular pitches, and most popular volumes used in his 32 piano sonatas, we partition the notes of all piano sonatas into disjointed sets based on time intervals. For example, the first set contains all the MIDI notes of all piano sonatas for which the time-on data (real number) falls into the time interval [0, 4) seconds. The second set contains all the MIDI notes of all piano sonatas for which the time-on data falls into the time interval [4, 8) seconds, and so on. We make sure that the last time interval ([412, 416)) does contain the minimum time length of all Beethoven sonatas, which is 413.908 seconds. For each such set, we compute separately the most used time-ons, the most used durations, the most used pitches, and the most used volumes. The instrument will be the piano.

For example, the most used time-ons for the interval [0, 4) are

{0., 3.22474, 0.255348, 0.892239, 1.10962, 1.26264, 1.54601, 1.76659, 1.98801, 2.44255, 0.220588, 0.353771, 1.84616, 2.0361, 2.07691, 2.29142, 2.875, 1.36586, 1.94684, 2.29798, 2.55489, 2.9198, 2.99606, 3.01975, 3.44032, 3.82882, 3.94501, 0.108707, 0.200018, 0.288461, 0.389386, …},

the most used durations for the interval [0, 4) are

{0.1948, 0.149358, 0.0925932, 0.149388, 0.14423, 0.149998, 0.288461, 0.14423, 0.220588, 0.16419, 0.168523, 0.19483, 0.10715, 0.162175, 0.288461, 0.103458, 0.162145, 0.165532, 0.184606, 0.396496, 0.12714, 0.149998, 0.203375, 0.203406, 0.217383, 0.221412, 0.32432, 0.396466, 0.436384, 0.454543, 0.454542, …},

the most used pitches for the interval [0, 4) are

{48, 52, 55, 60, 36, 43, 64, 57, 62, 59, 67, 53, 72, 50, 45, 69, 47, 65, 40, 24, 35, 76, 71, 41, 33, 31, 44, 68, 74, 77, 38, …},

and the most used volumes for the interval [0, 4) are

{0.345098, 0.305882, 0.360784, 0.321569, 0.337255, 0.282353, 0.290196, 0.352941, 0.313725, 0.298039, 0.376471, 0.27451, 0.329412, 0.368627, 0.384314, 0.235294, 0.407843, 0.243137, 0.4, 0.392157, 0.25098, 0.596078, 0.227451, 0.266667, 0.454902, 0.439216, 0.588235, 0.219608, 0.415686, 0.470588, 0.619608, …}.

Please note that some of the numbers might have been rounded and show only up to six important digits to improve readability.

Now, we create the notes of the new algorithmic music in the following way. We put together the most used time-on with the most used duration, the most used pitch, and the most used volume, which is {{0., 0.1948}, 48, Piano, 0.345098}. Please note that time-on plus duration is equal to time-off (time-off = time-on + duration). Next, we put together the second most used time-on with the second most used duration, the second most used pitch, and the second most used volume, which is {3.22474, 3.22474 + 0.149358}, 52, Piano, 0.305882}. We continue in this way. Out of these newly created notes, we pick the first n ones, where n is the rounded average number of notes played in each time interval based on all 32 Beethoven sonatas. For example, Beethoven used on average 31 notes in the time interval [0, 4), 32 notes in the time interval [4, 8), and 30 notes in the time interval [412, 416). We end up with the following notes for the time intervals [0,4) and [412, 416):

{

{{0., 0.1948}, 48, Piano, 0.345098}, {{3.22474, 3.3741}, 52, Piano, 0.305882}, {{0.255348, 0.347942}, 55, Piano, 0.360784},

{{0.892239, 1.04163}, 60, Piano, 0.321569}, {{1.10962, 1.25385}, 36, Piano, 0.337255}, {{1.26264, 1.41264}, 43, Piano, 0.282353},

{{1.54601, 1.83447}, 64, Piano, 0.290196}, {{1.76659, 1.91082}, 57, Piano, 0.352941}, {{1.98801, 2.20859}, 62, Piano, 0.313725},

{{2.44255, 2.60674}, 59, Piano, 0.298039}, {{0.220588, 0.389111}, 67, Piano, 0.376471}, {{0.353771, 0.548601}, 53, Piano, 0.27451},

{{1.84616, 1.95331}, 72, Piano, 0.329412}, {{2.0361, 2.19828}, 50, Piano, 0.368627}, {{2.07691, 2.36537}, 45, Piano, 0.384314},

{{2.29142, 2.39488}, 69, Piano, 0.235294}, {{2.875, 3.03714}, 47, Piano, 0.407843}, {{1.36586, 1.53139}, 65, Piano, 0.243137},

{{1.94684, 2.13144}, 40, Piano, 0.4}, {{2.29798, 2.69448}, 24, Piano, 0.392157}, {{2.55489, 2.68203}, 35, Piano, 0.25098},

{{2.9198, 3.0698}, 76, Piano, 0.596078}, {{2.99606, 3.19944}, 71, Piano, 0.227451}, {{3.01975, 3.22315}, 41, Piano, 0.266667},

{{3.44032, 3.6577}, 33, Piano, 0.454902}, {{3.82882, 4.05023}, 31, Piano, 0.439216}, {{3.94501, 4.26933}, 44, Piano, 0.588235},

{{0.108707, 0.505173}, 68, Piano, 0.219608}, {{0.200018, 0.636402}, 74, Piano, 0.415686}, {{0.288461, 0.743004}, 77, Piano, 0.470588},

{{0.389386, 0.843928}, 38, Piano, 0.619608}

},

{

{{412.719, 412.8}, 60, Piano, 0.27451}, {{415.508, 415.643}, 76, Piano, 0.352941}, {{412.014, 412.095}, 55, Piano, 0.321569},
{{413.665, 413.765}, 48, Piano, 0.337255}, {{414.945, 415.057}, 52, Piano, 0.368627}, {{415.827, 416.022}, 62, Piano, 0.360784},
{{415.854, 416.124}, 59, Piano, 0.329412}, {{412.266, 412.336}, 57, Piano, 0.345098}, {{412.772, 412.922}, 74, Piano, 0.313725},
{{413.075, 413.216}, 72, Piano, 0.376471}, {{413.283, 413.372}, 43, Piano, 0.392157}, {{413.34, 413.41}, 64, Piano, 0.384314},
{{414.863, 415.012}, 50, Piano, 0.462745}, {{415.073, 415.222}, 53, Piano, 0.478431}, {{412.017, 412.111}, 67, Piano, 0.290196},
{{412.274, 412.355}, 36, Piano, 0.305882}, {{412.35, 412.499}, 40, Piano, 0.486275}, {{412.397, 412.65}, 45, Piano, 0.611765},
{{412.397, 412.545}, 44, Piano, 0.447059}, {{412.426, 413.426}, 54, Piano, 0.4}, {{412.538, 412.627}, 56, Piano, 0.415686},
{{412.601, 412.702}, 47, Piano, 0.470588}, {{412.727, 413.029}, 81, Piano, 0.596078}, {{412.778, 413.475}, 65, Piano, 0.439216},
{{412.959, 413.708}, 69, Piano, 0.454902}, {{413.104, 413.19}, 71, Piano, 0.282353}, {{413.224, 413.478}, 58, Piano, 0.431373},
{{413.274, 413.575}, 61, Piano, 0.54902}, {{413.293, 413.734},49, Piano, 0.243137}, {{413.301, 413.983}, 38, Piano, 0.266667}

}.

Please open the supplemental MIDI-file *ResurrectingBeethovenS01-S32_4s_CMajor.mid* to listen to this newly created algorithmic music. In the second supplemental MIDI-file *ResurrectingBeethovenS01-S32_4s_8duration_CMajor.mid*, I increased the duration by a factor of 8. This version is more appealing to me.

Summary and Conclusions

In this paper, I showed one of my attempts to create new interesting algorithmic music that is based on the music data of great composers. I believe this music has its place in the world of applications of mathematical ideas.

References

[1] *Kunst der Fuge*. https://www.kunstderfuge.com/.

[2] MIDI Association. https://www.midi.org/.

[3] MIDI Association. General MIDI 1 Sound Set. https://www.midi.org/specifications-old/item/gm-level-1-sound-set.

[4] D. Schlingmann, *Mean Beethoven* https://archive.bridgesmathart.org/2022/bridges2022-449.html?fbclid=IwAR0tmvFIZjbfKtywGx_CmpK5GIBzK1vodzN52h81tx--Wt6pW3_6X1Cn4WM#gsc.tab=0.

[5] D. Schlingmann. Algorithmic music albums. https://open.spotify.com/artist/5iZrQGW6LFvuJ760K9UmEc/discography/album.

[6] D. Schlingmann. *Music via Math*. https://www.amazon.com/Music-via-Math-Dirk-Schlingmann-ebook/dp/B07VTR4DKX/.