Automatic Composition of Abstract Music by Distributed Agents

Max Ottesen[1]

[1]University of New Mexico, Albuquerque, NM 87131
mottese@unm.edu

Abstract

Since the early 1950's, interest in computer-generated music has been growing (Doornbusch, 2005). We decided to take the approach of generating music from an artificial life standpoint. We have done this by creating a set of rules for the Moveable Feast Machine, a cellular automata. By varying parameters such as how we treat dissonant sounds, what kind of chords we wish to produce, or the type of musical scale we want to work with, we can listen to the results change and improve over time.

Introduction

The purpose of this research was to look for new methods of computer-generating music. We decided to take an artificial life approach to the problem. Our model runs on the Moveable Feast Machine (MFM), a cellular automata with an emphasis on Robust-First Computing. For more information on the MFM, read Ackley and Cannon's paper (Ackley and Cannon, 2011). Current methods for computer-generating music include using mathematical models such as fractals (Hs and Hs, 1991), using grammars (Baroni et al., 1983), or using machine learning (M11, 2011). There are also evolutionary methods, which we decided to go with. The important difference to highlight is that current evolutionary methods require a human to act as the fitness function (Miranda, 2007), as it is difficult to capture the aesthetic qualities of music computationally. We demonstrate how low-level, autonomous agents can collectively write and rewrite note sequences with no human intervention based on simple elements of music theory such as *consonance*. A consonance is a combination of musical notes that are in harmony with each other due to the relationship between their frequencies.

Model Description

This model runs using a combination of three new elements: the Staff, the Note, and the Composer. Our model was designed to be able to run in the presence of DRegs, but for simplicity we did not include DRegs in our experiments.

Staff

The Staff element replicates itself to build a grid across the universe. Each Staff knows its position relative to the first Staff. This represents the musical staff in which Notes lie on. If a Staff atom sees that it is out of position, it will delete itself. The structure that the Staff element creates is self healing, so it can exist in the presence of DRegs.

Note

A Note represents a musical pitch. It computes what musical pitch it represents based on its position relative to the Staff elements around it. When the Note's behavior method is called, it checks around itself to verify that its stored pitch and position within the staff are the same. If it is not, it corrects itself by changing its stored pitch to match its position.

Composer

The Composer is our autonomous agent responsible for the writing and rewriting of note sequences. Composers diffuse around the MFM and can turn Res into Notes and more Composers. If a Composer is adjacent to a Note, it has a chance to move the Note based on the surrounding Notes and some internal state. There are three different versions of the Composer that we run against each other to compare the results. The first Composer simply moves Notes it is adjacent to around randomly with no regard for the surrounding Notes or its own internal state. The second Composer is trying to build any Triad it can. When evaluating a Note, it examines the Notes above and below to determine the Triad it could most easily make by moving the one Note it is currently working with. The third Composer is trying to construct a specific type of Triad. The Triad they are trying to build is stored internally. 7 different types of Triads can be made at the moment. We can make a C major, D minor, E minor, F major 6 4, G major 6 4, A minor 6 3, and B diminished 6 3. The reason for these triads is because we are working with only one octave and only on the notes in a C major scale. As Composers move into each others' event windows, they exchange information about what Triad they are trying to build. They then have a chance to change

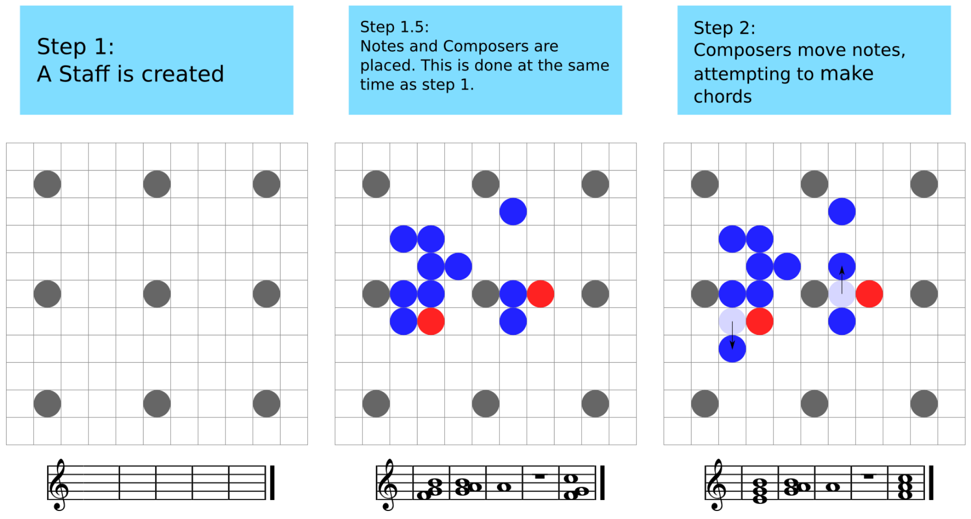
Step 1:
A Staff is created
Step 1.5:
Notes and Composers are placed. This is done at the same time as step 1.
Step 2:
Composers move notes, attempting to make chords

Figure 1: Visual representation of how the model works

the type of Triad they are building to the type that the other Composer is building. This chance increases for every Composer they meet that has a different Triad. If a Composer meets another Composer with the same Triad, the chance to change Triads goes down, however the chance never goes to 0. There is also a very small chance that a composer will randomly change its Triad. The reason Composers have specific Triads they are trying to build is so that there is some consistency between all of the sounds in one area. This makes the music sound less random.

Results

Before we can pull any data from the model, we must first define a metric to measure our results with. We do this by comparing the frequencies of the notes that make up a chord. We do this by using a system call Just Intonation. To give an idea of how this system works, we can think of two notes being played together as a fraction. For example, Middle C can be denoted as 1/1. The fact that Middle C is 1/1 is arbitrary. We could have chosen D to be 1/1, or F# to be 1/1. These fractions are the ratios between the frequency of the fundamentals the pitch you are looking at and your reference pitch (Middle C, in this case). If your reference pitch vibrates at 200 cycles/second and your second pitch vibrates at 400 cycles/second, we can call this second pitch 2/1. Fractions over 1 are special because they are some number of octaves above our reference pitch. If we are working in a C major scale, we can define the notes as follow:

C	D	E	F	G	A	B
1/1	9/8	5/4	4/3	3/2	5/3	15/8

Table 1: Ratios for the C Major scale

To generalize this a bit more, we can assign whole number values to each of the notes in the scale so that we can compare any notes rather than just one note and the reference pitch. These numbers are called Naturals. They leave you with numbers you can create ratios with that you can use to measure how consonant a specific chord is relative to the scale you are in. The naturals for the C major scale are:

C	D	E	F	G	A	B
24	27	30	32	36	40	45

Table 2: Naturals for the C Major scale

Using these naturals, we can find a ratio between the naturals of all of the notes in a chord. For example, a C-E-G chord (a C major chord) could be called a 24:30:36 chord. This ratio simplifies down to 4:5:6. Another chord we could make might be a E-G-B chord (an E minor chord). In the context of our C major scale, this chord can be represented by the ratio 10:12:15. Now, to compare our C major chord and E minor chord, we will perform the sum

$$\sum_{i=1}^{n} \frac{1}{r_n} \tag{1}$$

Where n is how many numbers you have in your ratio and rn represents the nth number in your ratio. By performing this sum, we find our C major chord to get a score of 0.6167 and our E minor chord to get a score of 0.25. The higher the score, the more consonant our chord is in the context of our key. In the key of C major, we can see that the C major chord scores higher than the E minor chord. We can perform this type of scoring with chords that have any number of notes in

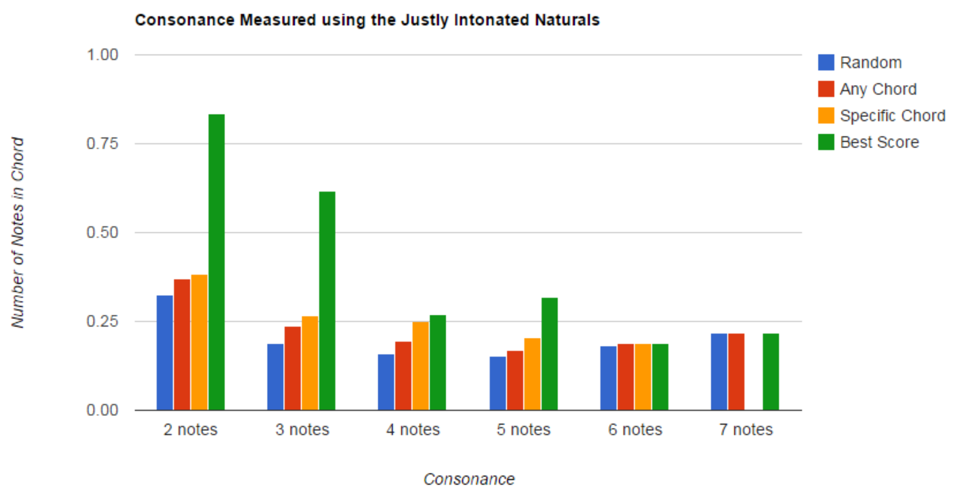

Figure 2: Consonance measured using the Justly Intonated Naturals

them, but it is important to note that with our equation, we can only compare chords with the same number of notes. We must also keep in mind that the scores we compute are only valid in our chosen key. A C major chord will get a different score if its evaluated in a different context. Now that we have a way to measure how consonant the music we produce is, we will compare 4 different methods of arranging notes. We will apply our equation to every chord and then average the scores of chords with the same number of notes.

Ill discuss the methods slightly out of order. The green bars represent the best possible score a chord with a given number of notes can get. These were calculated by hand. The first method the Composers used was to randomly move Notes around. The second method the Composers used was to attempt to make any triad. They would look around and try to make a triad with the lowest note it could see. Our third and final method was to have the Composers try to make specific chords. WIth this method, they will always move notes that arent in the specific chord they are making. If they are unable to move a note into their chord, they will delete the note. Keep in mind that the chord that each Composer is trying to make can change. Composers talk with each other and try to make the same chord as their neighbors, but they have a chance to randomly change their chord. This leads to clusters of Composers around the MFM with each cluster making a different chord. One interesting thing to notice is the lack of 7 note chords with the Specific Chord method. This is due to the fact that our experiments were run on the small, 2x2 version of the MFM. Since there tends to be clumps of Composers all making a specific chord and we’re running in a small environment, we only ever see 2 clumps for a maximum of 6 notes in any given column. We have shown that our model does increase the average consonance according to our chosen metric, but how do the results sound? We performed an informal survey where we ask participants to listen to 4 samples of music and say which one they like the most. Each one of the samples was generated using a different one of the 4 methods we discussed earlier. The participants of the survey were not randomly chosen. I asked friends, family, and classmates to take the survey and then pass it on to their own friends, family, and classmates. The results are shown in figure 3.

Discussion

Although we can measure the consonance of a piece of music and try to determine how good it is, it is hard to say anything certain since music is so subjective. There are many factors that influence what kind of music you like including culture, friends, or instruments you play. Because of this, we only refer to the consonance of a piece of music rather than how good it is. We can see that the randomly generated music and the hand-made, highest scoring music were both extremely unpopular with people. The music generated by the Composers trying to make any triad was liked slightly more, but not much. We can see that the music generated with our third method where the Composers aggressively form a specific triad was the most liked by far. Perhaps a more helpful way of looking at the survey results would be to rename the x-axis to overall consonance. This way, we can see that as the overall consonance of the music increases, the percentage of people that like that music goes up for a while, but then starts to drop off as the music gets to the point of best possible consonance. As pointed out in the Results section, its interesting to note that there are no 7 note chords with our Specific Chord method. This is due to the fact that Notes are always moved into a chord or deleted if they cant be put into a chord. Since we ran our experiments on the small version of the MFM, the areas

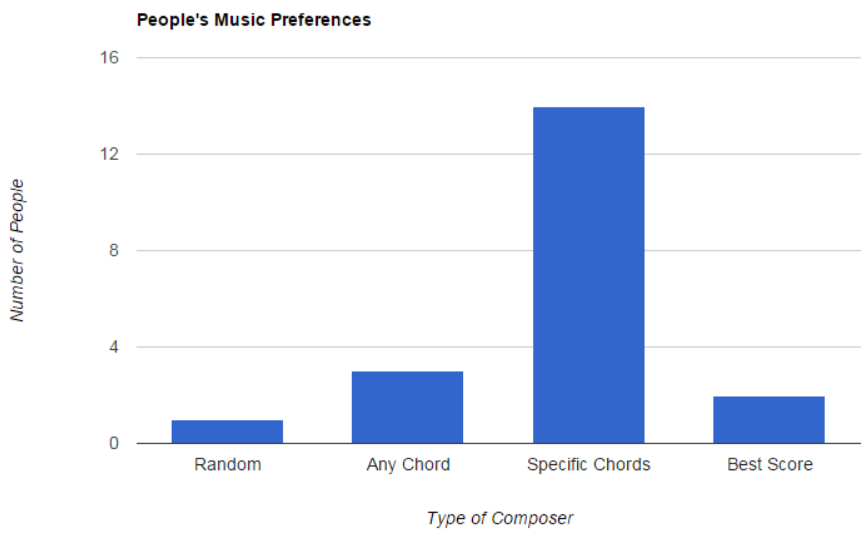

Figure 3: Survey asking which Composer produced the most likeable music

that clusters of Composers end up making chords in take up a very large vertical portion of the MFM. Because of this, its fairly typical to only see about 2 different chords being made in any given vertical space. With 3 notes per chord and 2 chords per vertical space, we usually see a maximum of 6 notes being played at once. Since we decided to make our model for the MFM, we ended up having to make a few very large sacrifices. The biggest sacrifice being the quality of the music we were able to produce. If we look at man made music, we can easily see that there is a lot of structure in it. We see melodies that work towards some type of climax and then ending resolution. We see harmonies supporting and sometimes mimicking the melody. We see clear sections that support or might even elaborate on other sections in the piece. In the MFM, we have no way to look at the big picture. Scope is severely limited (we can only see 4 spaces away from any given point) and this presents a very large challenge for us to overcome. If we want to make music that resembles something a human might make, how can we get past the fact that we can only make extremely small portions of the overall piece work together? One possible way might be to make more elements that represent the melody or different harmonies. They would then need to be able to communicate with each other beyond the scope of an event window. We might be able to add in some type of behavior to the Staff element that tries to facilitate this long distance communication. What it really comes down to is getting past one of the important parts of the MFM: no global state. Since we did not want to try and get past this important part of the MFM, we decided to use triads to try and make pleasing music. Triads are a good approach because they are very compact (thus able to be created and maintained within an event window) and usually sound good together. Most music simply takes some progression of triads and then elaborates on them. Lots of popular music today does not even elaborate on their triad progression. They will simply repeat the chords over and over while singing over it. We believe that this is the best approach to take when trying to generate music within the MFM.

Conclusions

By comparing our model to a baseline of randomly generated music, we can conclude that our model succeeds in creating relatively pleasing music. Another interesting result of studying this model is how overall consonance affects peoples liking of music. We can see that people obviously dont like randomly generated music. The more interesting result is that people dont like perfectly consonant music. There is some point in the middle wheres peoples liking peaks and then starts to decline. We did not have enough samples of music with differing scores be find this point with more accuracy, but future work could look for a more accurate estimate of this point. It might be interesting to measure existing pieces of music using this scale and see where they score.

Acknowledgements

I would like to thank Dave Ackley for his work developing the Movable Feast Machine and for offering his class on Artificial Life.

References

(2011). Unsupervised analysis and generation of audio percussion sequences. In Ystad, S., Aramaki, M., Kronland-Martinet, R., and Jensen, K., editors, *Exploring Music Contents*, volume 6684 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Ackley, D. H. and Cannon, D. C. (2011). Pursue robust indefinite scalability. In *Proc. HotOS XIII*, Napa Valley, California, USA. USENIX Association.

Baroni, M., Maguire, S., and Drabkin, W. (1983). The concept of musical grammar. *Music Analysis*, 2(2):pp. 175–208.

Doornbusch, P. (2005). *The music of CSIRAC : Australia's first computer music*. Common Ground Pub, Australia.

Hs, K. J. and Hs, A. (1991). Self-similarity of the ”1/f noise” called music. *Proceedings of the National Academy of Sciences*, 88(8):3507–3509.

Miranda, E. (2007). *Evolutionary computer music*. Springer, London.