The Spiral Model for Generative Harmony

Jackson Guy Spargur
Bard College

The Spiral Model for Generative Harmony

Senior Project Submitted to
The Division of Science, Mathematics, and Computing
of Bard College

by
Jackson Spargur

Annandale-on-Hudson, New York
May 2020

Acknowledgements

Thank you to my advisor, Kerri-Ann Norton, for helping me to make this a less painful process than it had any right to be, and to Matt Sargent, for his valuable guidance early in the project.

Table of Contents

Introduction

Music Theory

Before going into the details of this senior project, it will be necessary to lay the groundwork with some understanding of music theory, particularly the musical note system for identifying pitches, triadic harmony and interval relations for organizing notes that happen simultaneously, scales for organizing notes that occur sequentially, and keys for organizing notes and chords alike.

The pitches of the Western note system we are familiar with are identified by the letters of the alphabet between “A” and “G”: A, B, C, D, E, F, G. When a *pitch* becomes a *note* there is a subtle semantic change: a *pitch* is synonymous with a mathematical frequency, expressed in hertz. A sine wave oscillating at 440 hz. is equivalent to the pitch “A”. However, so is the sine wave oscillating at 220 hz., and 880 hz., and so on. A *note* represents a *pitch class* - the set of all pitches for which the next in the set has double the frequency of the previous. In common musical parlance these terms are often treated interchangeably, but I think for our purposes it is important to be specific.

There are two patterns predominant in organizing notes in the Western system, two kinds of *scale:* the *major* scale and the *minor* scale. A major or minor scale consists of seven notes in ascending order, and it begins on a particular note, for which it is named - for instance, the *C major scale* consists of C, D, E, F, G, A, B.

The relations, or *intervals* between pitches give scales their individual characters. The most important interval is the *octave,* which is the separation between the pitch “C”, and the pitch “C” which has double the frequency of the first “C”. The octave is so important because two notes an octave apart are at times almost imperceptibly different. If somebody with no training or familiarity with music was asked to sing a particular pitch, and that pitch was too high or low for them to sing comfortably, they would most likely be able to replicate the note in a different octave, possibly without realizing that they had just demonstrated an innate understanding of intervallic relations and pitch class. Such is the profound importance of the octave to music.

An important use of the octave concept is we can now think about pitches that are "within one octave". The seven notes of the C major scale are all within one octave, because the next note to follow the last "B" would be the "C" an octave above the beginning of the scale. Before introducing more intervals, two terms will be useful to know: the *whole* step and the *half step.* Whole steps and half steps are integral to the function of a scale, as between each sequential note in a scale is either a whole step or a half step, and as one can imagine a half step is smaller than a whole step - twice as small. In fact in Western theory there is no interval smaller than the half step, making it a very useful tool: every interval can be expressed in terms of half steps. The following are the remaining intervals within a one octave C major scale:

- A *major second* separates C and D, or two half steps
- A *major third* separates C and E, or four half steps
- A *perfect fourth* separates C and F, or five half steps
- A *perfect fifth* separates C and G, or seven half steps
- A *major sixth* separates C and A, or nine half steps
- A *major seventh* separates C and B, or 11 half steps

As one can perhaps guess from the necessity of terms like "major" and "perfect" preceding every interval, there are many named interval types, from *diminished,* to perfect, to minor and major, to *augmented*. It will not be necessary for our purposes to cover the details of this naming convention.

At this point we have seen seven notes, but in reality there are twelve, splitting the octave perfectly into equal parts. To get these additional five notes we need what are called *accidentals,* also known as "sharps" and "flats". Accidentals are essentially alterations to a letter note name. A "b" character after a note name makes that note lower by a half step, and a "#" character makes the altered note higher by a half step. A note that has been lowered by a "b" has "flat" added to its name, and a note that has been raised by a "#" has "sharp" added. For example, an "A" lowered by a half step is an "Ab", or "A flat", and likewise that note raised by a half step is an "A#" or "A sharp".

One of the most confusing concepts for new students of music theory is when to use sharps and when to use flats to alter a given note, as any note can theoretically be altered to be sharp or flat, and thus there are multiple ways to refer to the same pitch (e.g. Fb is the same as E, F is the same as E#, and F# is the same as Gb). It is often useful to think in terms of a scales, and particularly the fact that one would prefer not to have two of the same type of note in the same scale. For example, the sequence "A B B# D" would be better written as "A B C D". Another consideration is that one would rather not mix sharps and flats when possible, so for example a scale based on a sharped note should use only sharp or unaltered notes, and likewise for a scale based on a flatted note: it should use only flat or unaltered notes.

Now that we have all twelve notes to work with, we can look at the intervals within the counterpart to the C major scale: *C minor:*

- A *major second* separates C and D, or two half steps
- A *minor third* separates C and Eb, or three half steps
- A *perfect fourth* separates C and F, or five half steps
- A *perfect fifth* separates C and G, or seven half steps
- A *minor sixth* separates C and Ab, or eight half steps
- A *minor seventh* separates C and Bb, or ten half steps

It might seem at this stage as though we have covered an excessive amount of background information, having introduced these various structures for classifying and organizing pitches. What all of this is building towards, and what all of these structures have in common, is the answer to the question "why do some notes sound good together and some notes do not?" We have arrived at the notion of *consonance*, and by extension, *dissonance:* its opposite. Applying these terms is, to be clear, almost completely subjective. People have different ideas of which sounds they like to hear. However, within the Western system there is an agreed-upon hierarchy of sounds which, though possibly not representative of everyone's experience, is integral enough to the way we think about music and harmony that it works as a heuristic. From most to least consonant intervals:

- The octave/unison (e.g. C to C)
- The perfect fifth/perfect fourth (e.g. C to G, or G to C)
- The major third/minor sixth (e.g. C to E, or E to C)
- The minor third/major sixth (e.g. C to Eb, or Eb to C)

- The major second/minor seventh (e.g. C to D, or D to C)
- the minor second/major seventh (e.g. C to Db, or Db to C)
- The diminished fifth/augmented fourth (e.g. C to Gb, or Gb to C)

Note that this arrangement assigns each interval to a pair, such as the pairing of the perfect fifth and fourth. While there is a distinguishably different effect between, say, a perfect fourth and a perfect fifth, these pairs of intervals are often seen as equivalent in consonance, because if one of their constituent pitches were taken up or down the octave so that it became lower/higher than the other pitch, where before it was higher/lower, the identity of the interval would become that of its partner in the pair. A “C” below a “G” is a perfect fifth, and a “G” below a “C” is a perfect fourth.

Intervals are a foundational element to harmony, but when most people imagine harmony they think of chords, or simultaneities of three notes or more. Of course, every chord is comprised of at least three interval relations, and these interval relations are integral to the structure and function of chords. For our purposes we will only need to focus on a particular kind of chord called a triad: a three-note chord built from notes separated by intervals of a third. Triads are almost always used academically as the first introduction to chordal harmony, as they are the simplest way to represent chords. A triad can be any three unique notes together, as long as they can be organized in an orientation such that each note is separated from the next note by some kind of third interval. The triad is identified by the name of the note which sits at the bottom of this stack of thirds, which is known as the *root* of the triad, and a quality, such as major, minor, augmented or diminished. For example, a C major triad is “C E G”, and you can see that a major third separates “C” from “E”, and a minor third separates “E” from “G”. However, a C major triad can also be “G C E”, because if “G” were taken up the octave, it would again form that stack of thirds. This process of swapping the order of notes in a chord is called *inversion,* and it’s the same idea we invoked on the previous page with the pairs of intervals. Those pairs of intervals are inversions of one another, in the same way “G C E” or “E G C” are inversions of “C E G”.

Which quality a chord has is dependent on the types of third intervals it consists of:

- A major third under a minor third makes a major triad (ex. C E G)
- A minor third under a major third makes a minor triad (ex. C Eb G)
- A major third under a major third makes an augmented triad (ex. C E G#)
- A minor third under a minor third makes a diminished triad (ex. C Eb Gb)

Equally important as the third intervals contained within a triad is the type of fifth interval between the bottom and top note. Both major and minor triads are framed by a perfect fifth, which is the most consonant and stable interval aside from the octave or unison. This means that major and minor chords themselves have a stable, consonant sound. It is difficult to describe this idea of “stability” in concrete terms, but a major or a minor triad sounds complete in itself, and would not be out of place at the end of a piece of music. They have a “resolved” feeling. Augmented and diminished triads, which respectively are framed by augmented and diminished fifth intervals, both have a much more “unstable” sound, as if their effect is to imply another, more stable chord to follow which will resolve the instability. It would not generally be as musically satisfying to finish a piece of music on one of these triads, although this can be used to its own effect. It is far more common for this reason to hear major or minor chords than augmented or diminished chords.

The last and most broad organizational structure we need is the *key,* which can be thought of as the marriage of the scale and the chord. Keys, like scales and chords, are centered on a particular note, which is known as the *tonic* of the key. Other than this tonic note, there will be six other notes which are said to belong to this key, as well as seven triads. Keys can have the quality major or minor, which denotes whether a major or a minor scale is contained within the key. As this implies, any note which is in, for example, the C major scale is also in the key of C major, and likewise for the scale and key of C minor. We find the triads belonging to the key by creating a triad based on every degree in the key's scale, with the stipulation that every note in every triad also has to belong to this scale.

In C major, this would go as follows: the triad based on the tonic, which itself is also known as the tonic, would be "C E G", or a C major triad. Next in the scale is "D", and this triad would be "D F A", or a D minor triad. Next is "E", and "E G B" makes E minor. "F" makes "F A C" or F major, "G" makes "G B D" or G major, "A" makes "A C E" or A minor, and lastly "B" makes "B D F" which is a B diminished triad. These triads, which consist completely of notes in the key, are called *diatonic* triads or chords, and harmony which uses only these chords is called *diatonic harmony* - after the fact that the major and minor scales are sometimes also referred to as diatonic scales.

To reiterate, the key of C major consists of the notes of the C major scale, and also the chords of C major, D minor, E minor, F major, G major, A minor, and B diminished. As one can guess this pattern of major, minor, minor, major, major, minor, diminished is repeated in every major key. Because of this, music theorists use roman numerals to refer to the chords in a key, irrespective of the particular key tonic. The triad built on the first note is I, the second is ii, the third iii, followed by IV, V, vi, and vii^{o}. The capitalization is no mistake: major chords are represented by upper case numerals, minor chords by lower case numerals, and diminished chords are denoted by a lower case numeral followed by a small circle.

Roman numerals are used in the analysis of sequences of chords, or "chord progressions". Much could be said about chord progressions, but at a minimum we must cover the progression from V - I. The relationship between these two chords is the relationship between *tonic* and *dominant, dominant* being a word used to refer to the fifth degree of a scale, and this relationship is at the heart of Western tonal music. Tension and resolution, dissonance and consonance, stability and instability, and tonic and dominant all speak to the same idea. There are some sounds which have the effect of "home", and some which need to be "resolved" before the listener has the feeling of "home". The tonic is always "home", and the dominant always wants to be "resolved". This is a difficult effect to describe without an audio aid, but anybody who has grown up in a Western society is deeply familiar with this effect, whether they know it or not. It is how our music works. The reason it works is straightforward: a dominant chord contains the *leading tone,* which is the seventh degree of a scale. The leading tone is so named because it is a half step below the tonic, and because of the strong sense of "home" that the tonic possesses, the leading tone has a very strong pull towards the tonic. It is so close to the tonic, without being quite at the tonic, that our ears want it very badly to resolve upwards.

The term *dominant* refers specifically to the fifth scale degree, and the triad built on the fifth scale degree, but there are actually three diatonic triads which contain the leading tone of a given key scale. These triads are all known as *dominant-functioning,* or simply as having a "dominant function". In C major, these triads are G major, E minor, and B diminished. Not only do these triads all have the leading tone, in this case "B", in common, but E minor and B diminished both have two notes in common with G major: E minor has "G" and "B" in common, and B diminished as "B" and "D" in common. This further enhances the relatedness of these chords, and in fact chords with different functions can be related in this way as well. A key has not just a tonic, but also two other chords that can function as "tonic substitutes", because of the two notes they share with the tonic. In C major again, these would be A minor, which shares "C" and "E", and E minor, which shares "E" and "G". It is a strange fact that E minor, the iii chord, can function as both a dominant and a tonic substitute, but it just serves to show how much of music's effect is dependent on context.

The last group of chords, the *predominants,* are an intermediary step - tangibly different from the resolved sound of the tonics, but lacking the strong pull back to the tonic that the dominant chords possess. The predominant chords are those with two notes in common with the IV chord: the ii and the vi (another chord with multiple functions). Three of these chords might stand out at this point: the three chords to which all the others are compared. I is the tonic, while the triads with enough notes in common are tonic substitutes. V is the dominant, while the two triads that contain the leading tone and another note in common also possess a dominant function. Likewise, IV is in this view the "most important" predominant, and while I would argue this is true to a much smaller degree than that to which I and V are the "most important" tonic and dominant (for example, in the jazz tradition ii is clearly the predominant of choice), it is a useful simplification, and in practically every Western tonal style other than jazz it is certainly true. Therefore, it can be said in a way that the I, IV, and V chords together define a key.

All of this information pertains to the major keys. The minor keys have their own set of triads, built from notes diatonic to the minor scale. We'll walk through every triad of the key of C minor, in the same way we did C major. First is "C Eb G", which is the C minor triad. Next, "D F Ab", which is D diminished. Following this is "Eb G Bb", or Eb major; "F Ab C", or F minor; "G Bb D", or G minor; "Ab C Eb", or Ab major; and "Bb D F", or Bb major. This translates to roman numerals of i, ii^o, III, iv, v, VI, and VII. This clearly represents a much different pattern than the major keys, and one might wonder if the functions of all of the triads remain unchanged.

The answer is mostly yes, with some caveats. Notice that the one very important difference between the major and minor scales is that unlike the major scale, the minor scale has no leading tone. Whereas in the major scale the seventh scale degree is a half step lower than the tonic, in the minor scale it is a whole step lower, and as a result has a much weaker pull towards the tonic. Composers, for hundreds of years, since even before tonality became codified, have found ways around this problem - ways to tweak the scale to give it the tension and release inherent in the major scale. As a result of these "tweaks", theorists think of the minor scale as not a single scale, but three variations on a single scale. The first, default variation is known as the *natural minor* scale, and this is the minor scale we have used up until this point in the paper. The second variation of this scale is known as the *harmonic minor* scale, and it is the same as the natural minor except for the raised seventh degree, which changes the roman numerals to i, ii^o, III+ (the plus denotes augmented), iv, V, VI, and vii^o. One issue with this scale is that the raised seventh degree creates an augmented second interval between the sixth degree and seventh degree, which tends not to mesh well with the general mood of the minor key. As a result, this "scale" is really only used for harmonic purposes (as the name implies), in order to make use of a major V chord. The third variation, known as the *melodic minor,* solves this issue by raising the sixth scale degree as well. As the name suggests, this is the scale that composers, especially in the 17th and 18th centuries would use for melodies atop dominant chords in a minor key. This alteration leads to roman numerals of i, ii, III+, IV, V, vi^{o}, and vii^{o}. Note that the purpose of these three scales being merged under one umbrella is that composers generally choose the most useful elements of each of the three to use at any one time. So, for instance, at a given moment in a piece a composer might use the V of the melodic/harmonic minor, the raised sixth of the melodic minor, and then the next moment return immediately to the natural minor to make use of the less exotic major III chord.

With this, we have covered an adequate amount of theory to introduce the content of this project. On the topic of music theory there are always more details to be explored, and I have skipped or simplified many of them in the interest of efficiently covering the salient material. Before continuing onto other topics, a brief review of the pitch, the chord, and the key:

- **Pitches** are vibrational frequencies of air, which in Western music we have codified into twelve *pitch classes* or *notes,* based on the concept that a pitch doubled in frequency, or an octave above, represents the same note/a member of the same pitch class. For example, "A" = 220 hz., and "A" = 440 hz. are the same note, but different pitches.

- **A Chord** is, loosely, any simultaneity of pitches, although in this paper when I refer to a chord I will mean a *triad:* a chord which is made of three notes which can be arranged so as to create a stack of two *third* intervals, for example, “C E G”, but also “E C G”. Chords can have different *qualities,* such as *major* or *minor* based on the types of third intervals which compose them.
- **Keys,** in a way, are a unification of pitches and chords. A key can be said to be made up of the seven notes which constitute its *scale* (eg. the “C major scale” belongs to the “key of C major”), but a key is also a superstructure for the seven chords which are *rooted* on the seven notes in the key. For each note in the key’s scale, there is a triad: in the key of C Major the note C roots the chord C major, and then the note D roots the chord of D minor, and then E roots E Minor, etc., all the way up the scale to B and then beginning with C again. Another important concept is that each of these chords has a function within the key, and a name associated with that function: such as the *tonic*, denoted by the roman numeral I or i, *subdominant*, by iv or IV, or *dominant*, by v or V.

Generative Harmony

Generative music is a broad field, with connections to areas of study as disparate as computer science, mathematics, and linguistics. Which fields a generative music researcher finds themselves in contact with can have a lot to do with which aspect of music they are attempting to model. Linguistics, for instance, figures heavily into Fred Lerdahl and Ray Jackendoff's influential book *A Generative Theory of Tonal Music*, which aims to assemble a "grammar" for generating musical phrases, similar to the way that a grammar governs our language (Lerdahl). Their book is focused on the aspect of music, melody and phrase construction, which linguistics lends its insights to best.

Of course, a completely comprehensive and holistic approach to generating music would need to consider every aspect of music, like a human composer must. In this paper, I will focus on one piece of the puzzle, the process of generating harmony: creating rules by which multiple voices can sound simultaneously in a way that is familiar to the sensibilities of Western tonal music. My algorithm is concerned with decisions on the small scale only; it will not keep track of which part of a musical phrase or larger composition it is in at any particular step in its operation, but at each step it goes through a careful process to ensure that each voice is "performing" a note which harmonizes in the most ideal way with the other voices. While melody is not really a concern, something called *voice leading* is: my algorithm strives to ensure that each voice takes as smooth a path as possible, and moves in small steps, or jumps, or sometimes not at all. This not only creates a pleasing sound, but it is often the goal in real music as well - especially vocal music, as melodic lines with many large leaps are difficult for the human voice.

Before describing in detail the methods and results of my own approach to a generative harmonic system, I will introduce some prior research in this field, which was influential and in some cases instrumental to my work.

Realtime Generation of Harmonic Progressions Using Controlled Markov Selection

In this paper, authors Arne Eigenfeldt and Philippe Pasquier describe a method for generating harmony which uses a database of chords culled from jazz compositions by Miles Davis, Antonio Carlos Jobim, Wayne Shorter and Pat Metheny (Eigenfeldt 4), and Markov transition tables to generate a harmonic solution to a set of user-defined parameters. The user is presented with several Max/MSP function objects, allowing them to draw contours representing the levels of harmonic "complexity" and "tension", and the approximate contour

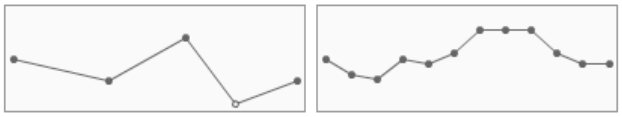
Figure 1: Max/MSP function objects (Eigenfeldt 9)

of the bass voice for the duration of the piece (Eigenfeldt 6). Once the user has specified their requirements, the system uses a "case-based system" to select from the transition tables of first, second, and third-order Markov chains the solution that best matches the user's requests of "complexity", "tension", and bass line (Eigenfeldt 3) - and the harmony is generated.

A first-order Markov system does two things: given a stream of input "events", it measures how often certain events lead to other certain events (for instance, if the stream is musical notes, how often the note "C" follows the note "D", etc.), and then it generates a "transition table", which contains the likelihood of each measured event being followed by each other measured event, based on how often it occurred in the stream. This means that if the user of a Markov system wanted to know a suitable note to follow the note "D", the Markov system would return one of the possibilities from the transition table, and it would be more likely to return one of the possibilities which was observed to occur more frequently. A second-order Markov system is the same, except that it "remembers" two events - so it decides, given two events in a row, which is a likely third to follow. A third-order Markov system does the same again, but it “remembers” one more event, and so on for higher orders.

In the authors’ system, the “events” fed into the Markov system are chords, which are “observed” as they occur in the database of real compositions. They represent these chords as sets of semitone distances from the root, so a major triad would be (0 4 7), because the major third is four semitones, and the perfect fifth seven semitones from the root. Each of these pitch sets occupies a unique index in a table, and chord events are referenced by this index and the bass movement from the previous chord as a positive or negative semitone value (Eigenfeldt 4).

Harmonic “complexity” is measured by the absolute sum of a chord’s semitone difference from a major triad. In a section of the composition where the user has drawn a peak in the “complexity” contour, chords will tend to have more notes, and have a more dissonant sound. Harmonic “tension” is measured by the number of common tones shared between adjacent chords in the piece. A moment in the piece which the user has drawn to be more “tense” will be one where successive chords to not share many notes in common (Eigenfeldt 8).

Put together, all of this leads to the generation of a chord progression which attempts to respond to a user outline of the emotional contour of the composition with a solution that follows trends observed in the jazz literature.

Towards a Mathematical Model of Tonality by Elaine Chew

Elaine Chew's dissertation *Towards a Mathematical Model of Tonality* presents a novel model, the "Spiral Array", for analyzing and generating Western tonal music. The Spiral Array consists of five rising concentric spirals occupying the same three-dimensional coordinate space, each spiral representing respectively pitches, major and minor chords, and major and minor keys. The purpose of the spirals is to allow these musical elements - pitches, chords, and keys - to be compared in a way that makes their distance in this three dimensional space analogous to their broadly perceived musical relatedness in the context of Western tonal music (Chew 3).

The spiral itself is based on the *Harmonic Network,* or *Tonnetz,* a development of the work of music theorist Hugo Riemann (Chew 33-34). The Harmonic Network arranges musical notes on a two-dimensional plane so that:

- the note to the right of a given note is higher by the interval of a fifth
- the note to the left is lower by a fifth
- the note above and to the right is higher by a major third
- the note below to the left is lower by a major third and
- the note above to the left is lower by a minor third
- and the note below to the right higher by a minor third.

The result of all of this is that those pitches which are separated by intervals of perfect fifths or major and minor thirds, i.e. the intervals most fundamental to Western tonal music, will in a graph theory sense be the most related pitches in this network.

The *Tonnetz* inspired and continues to inspire a large body of theoretical and computational work. Chew's model takes Riemann's idea one step further. Instead of pitches increasing horizontally to the right by a fifth on an infinite two dimensional grid, Chew takes this horizontal line of fifth relations and wraps it up and around in a cylindrical spiral. To preserve the spatial relationship of the two-dimensional Harmonic Network, every "node" on the spiral will be directly, vertically above a node which represents a pitch a major third lower, which means that there will be four nodes for every "turn" of the spiral. This arrangement, where the note directly above is a major third higher, is how the Network has sometimes been represented more recently, though it does differ from Riemann's original *Tonnetz,* which offsets major and minor third relations at diagonal angles.

Chew's Spiral Array has a crucial feature: when calibrated with the correct parameters it accurately reflects established hierarchies of pitch relationship. The intervals which are represented by the shortest distance on the Spiral Array are the perfect fifth/perfect fourth (these are equivalent), and the major third/minor sixth. The next-shortest interval distances are the minor third/major sixth, followed by major second/minor seventh, minor second/major seventh, and finally augmented fourth/diminished fifth (Chew 63).

This reimagining of the Harmonic Network into a three dimensional space carries another huge benefit. Chew outlines methods for using the locations of pitches in the Spiral Array to generate "centers of effect" - average locations between groups of given pitches. The center of effect for a note, the note a third above it, and the note a fifth above it will be essentially the location of the major triad, or three-note chord built on that note, and thus two spirals can be added, generated by finding the centers of effect for every major and minor triad. Chew then describes a method for doing the same with keys: by generating the center of effect of a triad, the triad rooted a fourth above it, and the triad rooted a fifth above it (in tonal theory these would be termed the tonic, subdominant, and dominant-functioning chords - the essential building blocks of a key), one can find the two final spirals, consisting of all the major and minor keys (Chew 58).

Flocking

Flocking was first outlined in Craig Reynolds' 1987 paper "Flocks, Herds, and Schools: A Distributed Behavior Model." His paper outlined a novel approach to simulating and digitally animating flocking, herding, and schooling behavior in animals: rather than painstakingly orchestrate the entire event, one could imbue each member of the flock, herd or school with the behaviors necessary for a flock to form organically through the interactions of the agents, or "boids", as he named them (Reynolds 2). He described three crucial behaviors:

- Collision Avoidance: boids should not collide.
- Velocity Matching: boids should match heading and speed with their nearest neighbors.
- Flock Centering: boids should be attracted to the center of the group of nearest boids (Reynolds 7).

With these behaviors implemented in each agent in one way or another, the flock/herd/school dynamics should occur in the interactions of the elements of the flock without needing to be orchestrated on a macro level, and it should work regardless of the size of the flock. Collision avoidance will take care of impending collisions, and velocity matching will tend to maintain separation, as speed and heading are matched with those boids closest and thus most likely to create a collision. Flock centering, the boids' attraction to their nearest neighbors, is the most subtle behavior. It allows such diverse benefits as boids on the outside of the flock being more strongly attracted to the center of the flock than those already in the center (due to the distribution of boids around them) and the natural parting of a flock around obstacles, as ultimately the boids have no allegiance to one larger entity, only those closest to them (Reynolds 8).

I originally intended to use flocking concepts in a much more literal sense than is now the case - having my own agents moving around a space with the classic visual effect of a moving flock or herd that identifies most flocking algorithms. I went in a different direction, but abstract ideas of the flocking tenets, such as collision avoidance, and even in a broader sense the idea of agents having an effect on other agents have always been at the heart of this project. Because of this, though my algorithm does not seem at first glance much at all like a flocking algorithm I will introduce one more piece of generative music research, which uses flocking concepts in a more overt fashion.

Generating Music from Flocking Dynamics

In their article *Generating Music from Flocking Dynamics,* Cristián Huepe, Marco Colasso, and Rodrigo F. Cádiz describe three approaches for applying flocking to music composition and synthesis (Huepe et al. 8). For all three approaches they used the Vicsek model (Vicsek et al.), a well-known minimal flocking implementation with a "noise" parameter that introduces states of order and disorder into the system - an easy analogy for the alternating states of consonance and dissonance in tonal music.

Their first approach they call the "direct approach", where the amplitude and frequency of a sine function are respectively mapped to the x coordinate and velocity of each agent on a two-dimensional plane (Huepe et al. 8). This approach provided an intuitive and straightforward way to map flocking concepts onto musical concepts. Ultimately, however, they found this approach to be more visually interesting than it was sonically interesting (Huepe et al. 10).

The second approach, called the "coupled oscillators approach", assigns a third variable, a sine function controlling sonic output, to each agent (Huepe et al. 10). This variable is affected not by the state of the individual agent but by interactions with other agents. Each agent is initialized with a "preferred frequency", and the actual frequency for each pitch at any given time is the weighted average of the preferred pitches within a certain range. The authors described this approach as more "cohesive", and in general creating a more subtle and interesting musical experience (Huepe et al. 11-12).

The final approach the authors cover is the "physical friction approach", which uses physics calculations to model the sound that agents in proximity to one another might make in a real physical environment, "rubbing together" as they moved past one another (Huepe et al. 13).

Sine tones are produced with a frequency controlled by the relative speed of the agents producing the sound, and an amplitude controlled by the number of agents in close proximity. Particles don’t need to actually collide to begin producing these sounds; all particles are always seen to be “in contact” with their closest neighbors, and thus producing a sound that reflects their relative velocity to these neighbors, with the overall amplitude of a cluster of neighbors proportional to its size. The authors noted that this approach produced “rich textures” (Huepe et al. 13-15).

Methods

The core of my project is a generative compositional model which uses agent interactions loosely inspired by flocking dynamics, and the work of Elaine Chew - particularly her Spiral Array model - to produce pitches in a manner sensitive to the guidelines of Western tonality. I use Chew's spiral as the environment in which the interactions occur, so that by comparison to the locations of the nodes on her spiral representing pitches, chords, and keys, I can analyze the movements of my agents through that space in terms of their movements through pitches, chords, and keys. In general during execution one Key Agent will take a random step in a direction, one Chord Agent will be aware of the location of the Key Agent (and thus what key it is "in") and take a step towards the location of the chord best suited to that key, and three pitch agents will be aware of the location of the chord agent, take a step each towards the pitches best suited to that chord, and also avoid choosing the same destinations. Here I describe the process in detail:

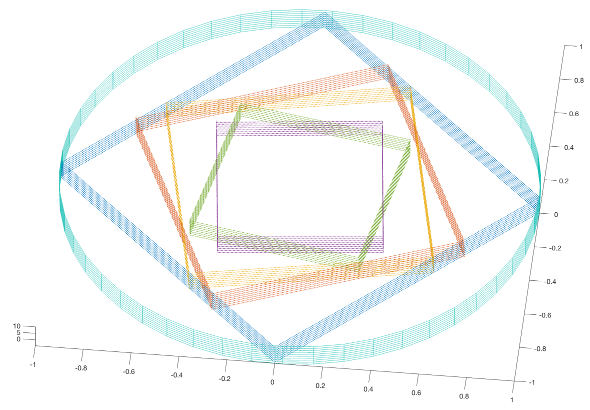

Figure 2: Top-down view of the Spiral Array, with the key spirals, purple and green, in the center; the chord spirals, orange and yellow, outside of that; the pitch spiral in blue outside of those; and the cyan cylindrical boundary

My model has two different modes of execution, which vary by a few different features but operate on the same core of code. I will describe one of them first, the "random key mode", as it is more complex, and then afterwards explain its differences to the other, the "chord sequence mode". I begin my generative model by setting the parameters of the Spiral Array model to Chew's specification: a radius of 1; a "height" of √(2/15) (height gained per quarter turn of the spiral); “alpha” of 1 (likelihood from 0 to 1 of major V chord instead of minor v chord in a minor key); “beta” of 1 (likelihood of minor iv chord instead of major IV chord in minor key); major chord pitch weights, major key chord weights, and minor key chord weights of [0.6025, 0.2930, 0.1145]; and minor chord pitch weights of [0.6011, 0.2121, 0.1868] (Chew 95-96). These parameters are passed to a function called Make_Spiral(), which plots the spiral as well as the boundary cylinder, and then returns five matrices of uniform length - each one containing the coordinates to, respectively, the pitches in the spiral, the major and minor chords in the spiral, and the major and minor keys in the spiral, spanning three octaves. As there are twelve pitches in each of these three octaves, each matrix has 36 columns of three-dimensional coordinates.

After this is done I initialize the Key Agent, the Chord Agent, and the three pitch agents, each represented by a small MATLAB table containing its current coordinates and the name of the key/chord/pitch to which it is currently closest. In this step I choose where exactly each agent begins. I initialize some additional matrices to keep track of the movement of the agents in different stages of the execution, and then move into the main loop. The following describes sequentially one iteration of this main loop, which can be run as many times as the user defines, depending on the desired length of the piece.

First, the Key Agent takes a random step, which is then evaluated by *Key_Boundary()* to make sure it stays within the bounds of the spiral. If it does not, it simply takes a step in the opposite direction instead. The key location is updated, and then the Chord Agent moves according to the new key location, using the function *Chord_Heading()* to find the chord location closest (and therefore most consonant with) the current location of the Key Agent.

As a measure of harmonic consonance, I use another function, *Consonance(),* within *Chord_Heading()*. *Consonance()* in the context that it is used here (in my program it is defined generally) takes the coordinates of the Chord Agent, the coordinates of the Key Agent, and the two lists of all of the key locations in the Spiral Array. It determines what the current key is by finding the closest key location to the Key Agent's coordinates, and then returns the distance from the chord location to whichever of the three representations of that key in the array is closest. In other words, no matter which octave the Chord Agent is occupying in the array, it is always going to have the same relationship to a given key, which is exactly what we want from a measure of consonance and what a simple distance calculation would be lacking. After *Consonance()* has returned the consonance of every possible chord location, the chord location with the lowest (best) consonance is the new target. After this, it is time to move the pitch agents.

First, the function *Pitch_Heading()* finds a provisional new location for each of the three pitch agents, and in doing so it functions almost identically to *Chord_Heading()* - except that for pitches, consonance is defined in relation to the Chord Agent, and it is also not the only factor in determining the new target. I also calculate the "voice leading distance" for each pitch agent, which I define as the distance in terms of steps on a musical scale from the current pitch location to each other pitch location. The "voice leading distance" from a C0 to another C0 is zero, from C0 to C#0 is 1, from C0 to D1 is 14, etc. I use the function *Smooth_Vl()* to calculate this measure for every potential pitch target, and then I incorporate one last factor: repeats of the same note are tracked in the variable *pitch_holds* later in the execution, and this variable is passed to *Pitch_Heading().* Sometimes one note can remain the most attractive choice for a particular voice for an excessive period of time, so the variable *pitch_holds* keeps track of how long each agent has remained on the same pitch. Once a voice moves, the tally is reset for that voice. These three variables are combined in the following formula:

$$priority = ((consonance^{7} / 150) + 0.5) + (((voice\_leading^{4} / 60) + 0.5) / 11.83) + (pitch\_holds^{3} * 0.002)$$

I arrived at the formula by simple trial and error. I knew that the formula needed to make some subtle but important choices, so it was imperative that I get the balance between consonance and voice leading just right, but unfortunately in real practice they are completely abstract and subjective concepts from which no mathematical precepts can be derived. I made two tables, the one for major chords reproduced on the next page, in order to see the effect that different coefficients and exponents would have on note choices. Each column represents a consonance value, as well as the interval from the root of the chord that would carry that consonance value. For instance, column four has the consonance value 1.25, which is how consonant, for instance, an E is on top of a C major chord in my model. Each row represents a voice leading distance, so I included 36 rows - for the full range of zero to 35 semitones that it is possible for two agents to be separated by in my model. Pitch holds are a dampening factor which I calibrated after the rest of the formula was solidified, so they were not considered in this table.

Once I had the table, I could see the choices that the formula would make under different circumstances. How large of a leap, for instance, would the formula allow to happen in order to place a note on the root of the chord, rather than remain on the major 6th? The answer, as it turns out, is three semitones, or a minor third. Would the model rather leap a fifth to a major third, or move a half step to an augmented fourth? The answer is leap a fifth. If I ever did not like the answer that the formula gave me during this process, I would tweak it, and see if it produced the

	0.5 (root)	0.75	1 (P5)	1.25 (M3)	1.5 (P4)	1.75 (M6, m3, m6, M2, M7)	2 (m7)	2.25	2.5 (m2, +4)
0	0.542	0.543	0.549	0.574	0.656	0.877	1.396	2.488	4.611
1	0.544	0.545	0.550	0.575	0.658	0.879	1.397	2.490	4.613
2	0.565	0.566	0.571	0.597	0.679	0.900	1.418	2.511	4.634
3	0.656	0.657	0.663	0.688	0.770	0.991	1.510	2.603	4.725
4	0.903	0.904	0.910	0.935	1.017	1.238	1.756	2.849	4.972
5	1.423	1.424	1.429	1.455	1.537	1.758	2.276	3.369	5.492
6	2.368	2.369	2.375	2.400	2.482	2.703	3.221	4.314	6.437
7	3.925	3.926	3.932	3.957	4.039	4.260	4.778	5.871	7.994
8	6.313	6.314	6.320	6.345	6.427	6.648	7.166	8.259	10.382
9	9.786	9.787	9.792	9.818	9.900	10.121	10.639	11.732	13.855
10	14.631	14.632	14.637	14.663	14.745	14.966	15.484	16.577	18.700
11	21.169	21.170	21.176	21.201	21.283	21.504	22.023	23.115	25.238
12	29.756	29.757	29.763	29.788	29.870	30.091	30.609	31.702	33.825
13	40.780	40.781	40.787	40.812	40.894	41.115	41.634	42.727	44.849
14	54.665	54.665	54.671	54.696	54.778	55.000	55.518	56.611	58.734
15	71.865	71.866	71.872	71.897	71.979	72.200	72.719	73.811	75.934
16	92.873	92.873	92.879	92.904	92.986	93.208	93.726	94.819	96.942
17	118.211	118.212	118.217	118.242	118.325	118.546	119.064	120.157	122.280
18	148.437	148.438	148.444	148.469	148.551	148.773	149.291	150.384	152.506
19	184.145	184.146	184.151	184.176	184.259	184.480	184.998	186.091	188.214
20	225.958	225.959	225.965	225.990	226.072	226.293	226.811	227.904	230.027
21	274.536	274.537	274.543	274.568	274.650	274.871	275.390	276.483	278.605
22	330.573	330.574	330.580	330.605	330.687	330.908	331.427	332.519	334.642
23	394.796	394.796	394.802	394.827	394.909	395.131	395.649	396.742	398.865
24	467.964	467.965	467.971	467.996	468.078	468.299	468.817	469.910	472.033
25	550.873	550.874	550.880	550.905	550.987	551.208	551.727	552.820	554.942
26	644.352	644.353	644.358	644.384	644.466	644.687	645.205	646.298	648.421
27	749.262	749.263	749.268	749.293	749.376	749.597	750.115	751.208	753.331
28	866.499	866.500	866.506	866.531	866.613	866.834	867.352	868.445	870.568
29	996.993	996.994	997.000	997.025	997.107	997.328	997.847	998.940	1001.062
30	1141.709	1141.710	1141.715	1141.741	1141.823	1142.044	1142.562	1143.655	1145.778
31	1301.643	1301.643	1301.649	1301.674	1301.756	1301.978	1302.496	1303.589	1305.712
32	1477.826	1477.827	1477.833	1477.858	1477.940	1478.161	1478.679	1479.772	1481.895
33	1671.324	1671.325	1671.331	1671.356	1671.438	1671.659	1672.178	1673.270	1675.393
34	1883.236	1883.237	1883.243	1883.268	1883.350	1883.571	1884.089	1885.182	1887.305
35	2114.694	2114.695	2114.701	2114.726	2114.808	2115.029	2115.547	2116.640	2118.763

Figure 3: Table representing note choice formula for major chords:
*priority = ((consonance^7 / 150) + 0.5) + (((voice_leading^4 / 60) + 0.5) / 11.83) + (pitch_holds^3 * 0.002)*

desired result. Eventually I arrived at something which is by nature imperfect, but I believe an acceptable heuristic for the balance a composer must strike between ideal note choice and maintaining good voice leading.

After the provisional targets are found for each of the pitch agents, the function enters a loop for the purpose of checking the pitch agents for any unacceptable movement, such as two pitch agents occupying the same note in the same octave, or parallel fifths or octaves (when two notes a fifth apart move to two new notes which are also a fifth apart, or the same for two notes an octave apart). Parallel fifths and octaves were extremely uncommon in a certain style and era of music, so much so that when learning music theory they are often thought of as “off-limits”. My reasoning for disallowing them is slightly more pragmatic.

The less important reason, and this relates to the reason they were so rarely used in Baroque-era Classical music, is that they can sound a little bland, and contrary motion (one voice moving up another moving down) is almost always preferable. More importantly, if ever pitch agents in my model find themselves in octaves, they will without fail make the same decisions from that point forward. They are essentially occupying the same note, and therefore every potential note destination is now an equivalent distance for both agents. Two pitch agents mirroring one another perfectly is not something that would sound interesting. The same logic applies to notes in unison, except that in real three part harmony voices are almost never allowed to be in unison. As the model aims to create the richest harmony possible, I decided to disallow unisons completely - rather than allowing unisons but not parallel unison movement, as I do for octaves and fifths.

A 1x3 array called *choice_array* is maintained for the three pitch agents, and it is this variable that keeps track of the provisional targets. The value in each of the three columns represents the place on the three sorted lists of targets which each of the three pitch agents will choose. If the first pitch agent and the second pitch agent are found to be in unison, for instance, the second pitch agent may move from its first to its second choice, which will represent a new, slightly less desirable pitch target. The checks for unisons are made in the function *Pitch_Flock()* (a nod to the collision avoidance aspect of flocking), and the checks for parallel fifths and octaves are made in the function *P_Octave_Fix()*. If any of the choices needs to be changed, this correction loop will happen again, in case the change has created any new issues. Once a loop happens with no corrections, the *choice_array* is used to index the target arrays for each agent, and they each have their new destinations.

This entire sequence, including all five agents, repeats as many times as the user has defined, and finally the path of each agent through time is plotted, either in an animated or static fashion, again according to the user's specification. I earlier mentioned an alternate generation mode called the "chord sequence mode", and I will explain that now. This mode exists for the purposes of bypassing the key agents and chord agents, in order to generate a harmonization of a sequence of chords that already exists. One could use any existing song as the input for this mode, as long as one knows its chord progression and can represent it as a string array of chord names. This mode requires almost no modification from the random key mode; the biggest difference is simply that the corresponding step in the input chord sequence is used as a substitute for any chord location that might have been chosen as part of the random key process. From that point forward, the executions are identical.

All that remains, for either mode, is to create the MIDI output, and for that the function *MidiOut()* is called. *MidiOut()* is built to work with two MIDI tools for MATLAB written by Ken Schutte: *matrix2midi()* and *writemidi()*. Those two functions convert an input matrix into MIDI information, and then write that MIDI information to a .mid file - but in order to use them, I need to translate my program's output into a matrix that can be read by *matrix2midi()*. *MidiOut()* fulfills that purpose, and then executes Schutte's code.

First, *MidiOut()* initializes the empty MIDI matrix. In the MIDI matrix every row represents a note event, meaning there are three for every iteration of the main loop (one for each pitch agent), and every column represents information about that note event. Column one represents what track the note event belongs to: each third of the matrix belongs to a pitch agent, so each third will be assigned its own track. I assign every row to be in MIDI channel one and at volume 120; that information belongs to columns two and four respectively. Column three is the pitch information, which I find by mapping the pitch and octave name for the pitch agent (for instance, "B1") to the MIDI notes between 48 and 83, which represent the pitches between "C0" and "B2", the extent of my three octave range.

Note that there is a standard notation for pitches which uses this letter + number notation, and that I am not adhering to it, though my notation appears identical. For instance, in the standard notation MIDI 48, or 130.81 hz., is represented by "C3" - but in my program's notation it is "C0". I do this for the reason that a C0 in the standard notation represents the frequency at 16 hz., which is lower even than most commercial speakers can reproduce. As I wanted to loosely match the range of my output to the range of the human voice, C3 is a much more reasonable lower limit. However, I thought it might be confusing to start my indexing from 3, so I decided, for my program's purposes alone, to refer to C3 as "C0", C4 as "C1", and so on.

Column five represents the start time of the note, and column six the end time, as multiples of whatever note duration I choose (I have chosen 0.5, which represents a quarter note in Schutte's MIDI implementation, so every increment of time is expressed as a positive integer multiple of 0.5). I decided to translate every sequential repetition of a pitch to a hold of that pitch, which involves adjusting the end time of a held note to the point at which *MidiOut()* detects that note has changed. Because this effectively nullifies any note event that is simply a sequential repetition of a previous pitch, the final step is to delete those rows that represent those nullified note events, for they are not needed and interfere in the proper parsing of this matrix.

The final output of my model is a .mid file, which I open in the music notation program Finale.

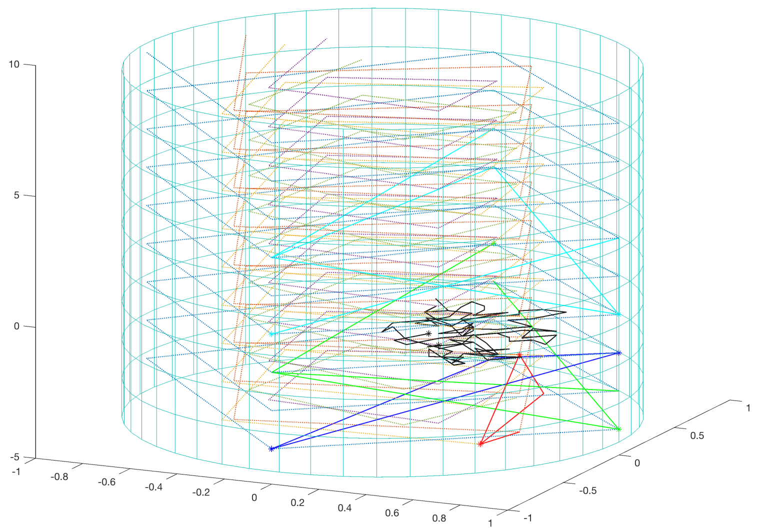

Figure 4: Visual output of a completed 100-step run of the Random Key model - Key Agent in black, Chord Agent in red, Pitch Agents in blue, green, and cyan.

Results

To illustrate numerically the output of my model, and the effect that the different elements of my model have on the output, I have generated several metrics tracking the behavior of the pitch, chord, and key agents. Because the model governing the initial pitch decisions is a linear combination of the output of my *Consonance()* and *Smooth_Vl()* functions, I run the metrics on three different models: the "combined" model, which is the full model; the "consonance" model, which makes pitch decisions solely on the consonance of the prospective pitches; and the "voice leading" model, which optimizes parsimony of voice leading in its decisions. These two alternative models are useful in the extremity of their results; not as viable alternatives to the combined model. The Consonance Model cares only for consonance, so it will move to whichever note is most consonant, even if that note is a huge, unperformable leap away from the previous. The Voice Leading model likewise cares only for how little it moves, and will remain in the same spot until the safeguard I wrote ensuring voices do eventually move tells it to.

Below is an explanation of the metrics:

- **Consonance per Step:** Tallies the total average consonance between the three pitch agents and the chord agent at each step (and also between the Chord Agent and the Key Agent, if there is a key agent involved) and divides at the end by the number of steps. As "consonance" is no different than distance between agents per Elaine Chew's Spiral model, a "lower" consonance value means the agents were more related, and therefore more consonant.
- **Voice Leading Distance per Step:** Measures the total average semitones traveled of the three pitch agents at each step, and divides by the total steps.

- **Length:** Measures the total average distance traveled by the three pitch agents, the total distance traveled by the Chord Agent, and the total distance traveled by the Key Agent if there is one.
- **Speed:** The Length metric divided by the number of steps.
- **Displacement:** Measures the distance between the last and the first location for each agent, with the pitch agents averaged together.
- **Directionality Ratio:** Represents Displacement divided by Length, in other words what percentage of the total distance traveled by the agents was in a consistent direction away from the origin.
- **Isotropy:** Distance traveled by each agent at each step (or in the pitch agents' case, average distance traveled by the three) is separated into one of twenty-seven bins, representing each of the twenty-seven possible directions for three-dimensional movement. An agent could move back, left, and down; back, left, and level; back, left and up; back, straight, and down; stay completely still; etc. Once the run is complete, the largest-valued bin is divided by the total number of bins which had value added to them over the course of the run, and then this value is divided by the Length metric, and that is the Isotropy value. Isotropy measures how dispersed the movement of an agent is; whether it moved in many directions over the course of its run or stuck to a few. Ideologically it is similar to Directionality Ratio, and under certain circumstances the output could be similar. It does, however, measure a subtle difference. Take for example an agent which moves away from the origin 10 units of distance on a straight line, and then back to the origin 9 units on the same line. The

DR would be quite low at *1 / 19 ≈ .05*, as the agent traveled 19 units but was displaced only 1, but the Isotropy would be much higher at *10 / 2 / 19 ≈ .26*, given that the agent only traveled two different directions. A benefit to these two metrics is that they both have a range of possible values from NaN (division by zero) to 1, so they can be directly compared in this way.

- **Fifth Displacement per Step:** Measures the average distance traveled on the circle of fifths by the three pitch agents per step, and the distance traveled by the Chord Agent per step. Acts in a sense as a "displacement", as travel can be interpreted as either up or down in fifths (C to G would be "up one fifth" or "down eleven fifths"), but whichever distance is smaller is chosen as the interpretation. So C to B is up five fifths, not down seven, and likewise C to Db is down five, not up seven.
- **Semitone Displacement per Step:** The same as Fifth Displacement, but using semitone distance instead of distance on the circle of fifths, and only considering the averages of the three pitch agents. It is also similar to Voice Leading Distance per Step, but different in that VLDpS measured the semitone distance as an absolute value, whereas SDpS cares in which direction the travel occurred. In this case it is not necessary to make a determination on how to interpret the direction of travel, as the model provides explicit directions for which pitch in which octave to move to, and thus the direction of movement is given by the relationship between the current pitch and the chosen pitch target.

In order to calculate the metrics, each model ("combined", "consonance", and "voice leading") is run simultaneously, using the same starting pitches and the same key and chord agents to ensure the comparison is as accurate as possible. The metrics are then averaged over 10 runs, with random starting locations for each agent chosen each time. To show the effect of different run sizes on the metrics, I run the model up to ten steps, twenty-five steps, fifty steps, 100 steps, and 250 steps.

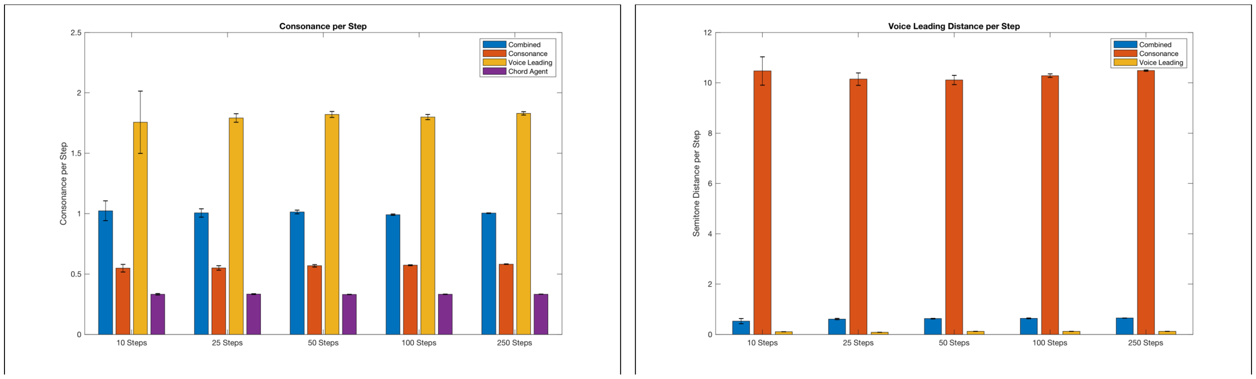

Figure 5. Consonance (left) and Voice Leading (right) metrics (error bars represent 95% confidence intervals)

These two figures function as much as a sanity check as anything else - but it's plain to see that the three models (and the Chord Agent) are working as intended in regards to these metrics. The Consonance Model runs the lowest (best) consonance values, save only for the Chord Agent, which in effect uses a consonance-only model of its own. The spirals containing the chord locations are closer to the key location spirals than the pitch location spiral, which I suspect is the only reason the chord agent has the best consonance performance. The Voice Leading Model has the highest consonance values, owing to the fact that it instructs its pitch agents to stay still whenever they can, despite what the chord and key agent may be doing. It is interesting to note the error bars on the ten-step run of the Consonance metric: there is the smallest bit of variance there, owing to the fact that the starting locations for pitches and chords and keys are randomized, and are sometimes liable to harmonize coincidentally. The more steps in the run, the more likely that even on these runs which begin on a consonant note for the Voice Leading Model, the key and chord agents will go elsewhere and the pitch agents will continue to stay put, creating dissonance.

The Voice Leading metrics are even more extreme than the Consonance metrics, with the Consonance model far outstripping all others in Voice Leading Distance per Step. The explanation for this is simply that Consonance is not affected by octave, and if the same note in a different octave happens to fall to the top of the list of potential targets, even though all three notes are essentially in a tie for first, the model will choose to jump the octave, adding 12 or even 24 semitones to the tally. This would of course be undesirable in the Combined Model, but the Consonance and Voice Leading models are meant to possess extreme behavior.

Between Length and Speed (Figure 6, on the next page) one can see the same relationships between models and agents hold, being normalized in Speed by the number of steps. Predictably, the Consonance Model moves far further than the rest, for the same reasons covered in the discussion of the Voice Leading metric. Following that discussion as well, Voice Leading moves the least of the pitch agent models, while Combined is in the middle. The Key Agent moves the least overall, owing both to the fact that its step sizes are smaller than even the smallest step the Chord or pitch agents could take, and that its Speed is consistent, on account of the fact that takes a step of very similar length every time.

The Chord Agent is interesting in that in the ten-step runs its Speed varies from slightly above the Speed of the Combined Model to near the speed of the Voice Leading Model, per the confidence intervals, but as the runs get larger its mean shrinks along with the error bars, past the Voice Leading Model and nearing the Key Agent's tiny speed. The explanation for this is

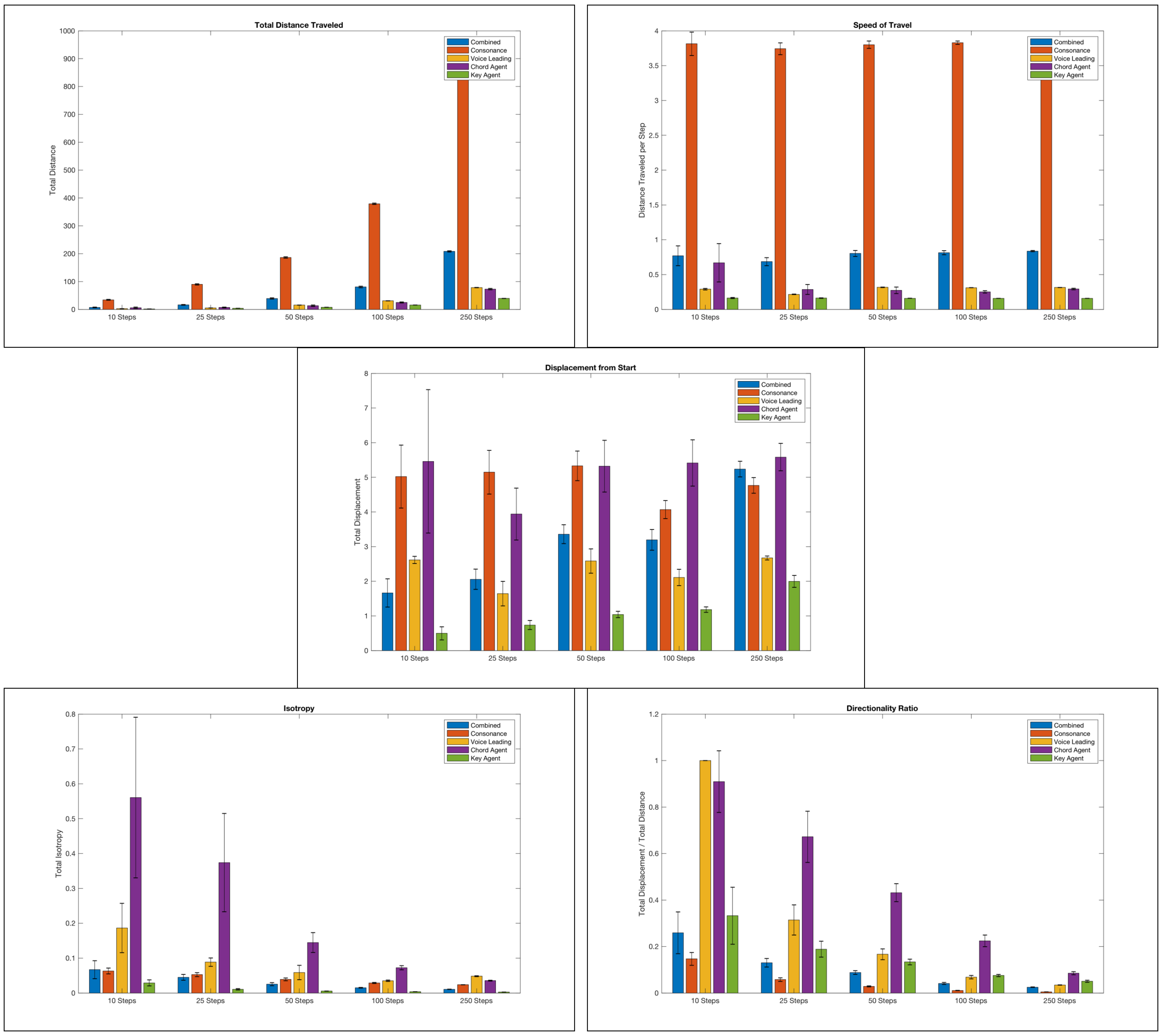

Figure 6. Length metric (top left), Speed metric (top right), Displacement metric (middle), Isotropy metric (bottom left), Directionality Ratio metric (bottom right) - on random key run (error bars represent 95% confidence intervals)

straightforward at the ten-step run: the Chord Agent's random starting location will at times clash with the Key Agent's, requiring a large shift in such cases. Other starting locations may require no movement at all. My belief as to why the mean tends to shrink over time, informed by observation of many runs, is that it is due to the inevitable periods of stagnation that the Chord Agent will encounter, as the Key Agent wanders but does not enter any new key areas, leaving the Chord Agent with no better option than to remain at its current location. If the run does not begin with such a period, then the larger the run is, the more likely it is that one or more will occur. In other words, dramatic movement on the part of the Chord Agent is most likely to occur near the beginning of the run, and as the runs get larger, this beginning period means less and less to the average speed of the agent. Of course, in this particular 250-step run the mean of the Chord Agent's speed is actually slightly higher than that of the 100-step run, so this is not always strictly true.

The Displacement graph shows a lot of noise, but some sense can be made of it. The Key Agent clearly displaces further the longer it is allowed to run, and if one looks at the rest of the agents on the whole, it appears that they largely do as well - with the possible exceptions of the Consonance and Voice Leading models. The Chord Agent and the Consonance Model can be expected to perform similarly here, as both are subject to the same chance that at any point they will suddenly leap the octave in their blind quest for consonance. Both show volatility in their confidence intervals, which are far larger than those of the other models. In general the Consonance Model/Chord Agent are likely to displace the most, followed by the Combined Model, the Voice Leading Model, and the Key Agent, a hierarchy consistent with other metrics.

The Isotropy and Directionality Ratio graphs show some notable differences. No agent on any run managed to get to an Isotropy of 1 (though judging by the error bars, the Chord Agent on a ten-step run may have been very close) but the Voice Leading Model apparently had a DR of 1 on every single run (with the Chord Agent mean near 1 DR as well). This is due in part to the way the different metrics combine the actions of the three pitch agents constituent to each model. Imagine that all three pitch agents over the course of a run move the same amount, each in a different direction. The DR will be 1, as the respective displacements and lengths will be added up, and these two sums divided, which must be equal and therefore divide to 1. The Isotropy will be less than 1, as each separate direction will be placed in a separate bin - meaning the maximum bin will be divided by 3, and then that number by the total length. Despite this difference, in general it seems the Isotropy and DR of both the Voice Leading Model and Chord Agent are fairly similar in contour, and while the Voice Leading Model for the above reason has much lower Isotropy numbers, the Chord Agent's are only slightly smaller.

The rest of the models/agents have lower Isotropy than DR values as well, but the Consonance Model is notable in that at higher step-size runs its DR becomes lower than its Isotropy. Its DR is consistently below that of the Combined Model and Key Agent, but this does not hold true for its Isotropy. If one looks at the graph of Displacement, and then the graph of Length, the DR graph makes a lot of sense. DR = Displacement / Length, and as Displacement is practically a wash in terms of the differences between models/agents, the relationships between these models/agents in DR look something like an inversion of how they look in Length, with models/agents that traveled less distance having higher DR values, and vice versa. With this information, the question becomes why the Consonance Model has a higher Isotropy than expected relative to the other pitch models. I believe the answer to this is simply that the pitches in the Consonance Model are moving further at each step. It is an interesting twist, as right next to it on the graph is the Voice Leading Model, which has an even higher Isotropy value despite hardly moving at all. In the case of Voice Leading, the fact that it only moves a few times means the little movement it has is very concentrated. In the case of Consonance, its movement is varied, but its max bin is going to be fairly high, so its Isotropy will be on the higher side as well.

The Combined Model, a happy medium moving moderately in every way, winds up with the lowest Isotropy score among the pitch models, approaching even the randomly-moving Key Agent.

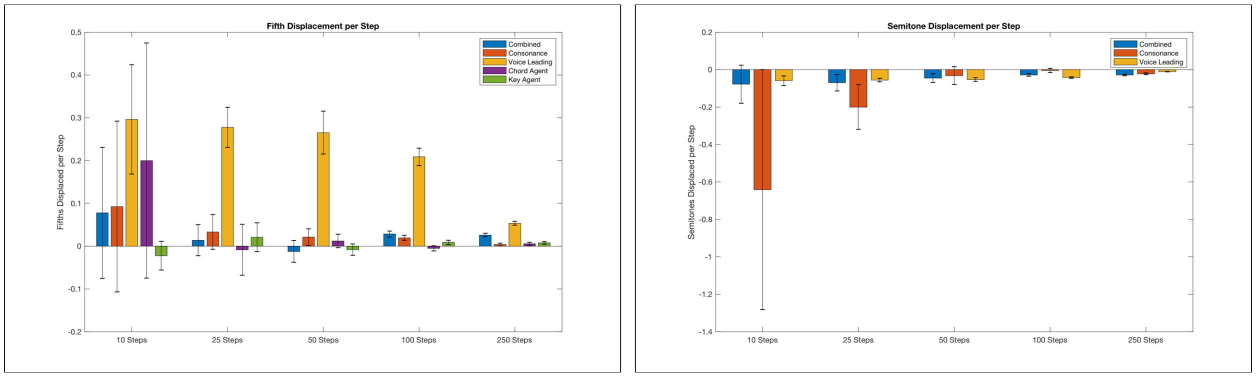

Figure 7. Fifth Displacement (left) and Semitone Displacement (right) metrics (error bars represent 95% confidence intervals)

The Fifth Displacement plot looks almost (with the exception of the Voice Leading Model) as one would hope it would, given the randomness underlying this model. There is some volatility in the smaller runs, and by the largest run the displacement of almost all of them is at or very near zero. The Voice Leading Model is a clear outlier. The only real reason that the VL Model would have to tell one of its pitch agents to move is that the agent had already held the note for as many steps as it is allowed. When that happens, it will be made to target its second choice, which will be either a half step below, or a half step above, as those are the two smallest movements it can make. What do these options look like in terms of fifth displacement? Take a move from C to Db: up one half step. One would need to move up in fifths C to G to D to A to E to B to F# to C#, which is the same as Db, for a total of +7. This could be better expressed, however, as moving down five fifths: C to F to Bb to Eb to Ab to Db. So the metric would interpret an upward movement of a half step as a displacement of -5. To move down one half step is just the opposite: in fifths up from C is C to G to D to A to E to B, for a displacement of +5.

All of this is simply to articulate the real mystery, which is why, when the Voice Leading Model is required to settle on its second choice, it so often opts to move down rather than up a half step. Rather than solve this mystery, I bring the reader's attention to the Semitone Displacement plot, which ought at this point to be scratching heads for a similar reason. Why the bias towards downward motion? I can unfortunately offer no solution to that mystery other than confirming that the list of targets for the pitch models do indeed always appear to be sorted with the lower note higher up, given that all else is equal. I can only conclude from this that the issue comes down to a quirk of the MATLAB sort() function.

I presented these metrics on the random key mode of the model as a sort of "control", given that the Key Agent moves randomly, and the Chord Agent mirrors exactly the Key Agent. I now show the same metrics on another mode of operation for my model, wherein the Key Agent is bypassed completely, and the Chord Agent is simply an array of chords taken as input, with the pitch agents acting as usual. The chord sequences I used represent a small selection of some different kinds of music: the changes to the jazz standard "Giant Steps" by John Coltrane, 80's pop creation "Africa" by Toto, and the traditional spiritual "Amazing Grace".

To make the figures, I again took the average of ten runs each, with again randomized starting locations for the pitch agents. Obviously the starting chord is not randomized, and there is only one run size considered; the size of the particular sequence of chords used as input. In the previous section of analysis it was the inner workings of the model under scrutiny, with the "music" being a randomized baseline. In this mode some insight may be gained through the model's process about the music itself.

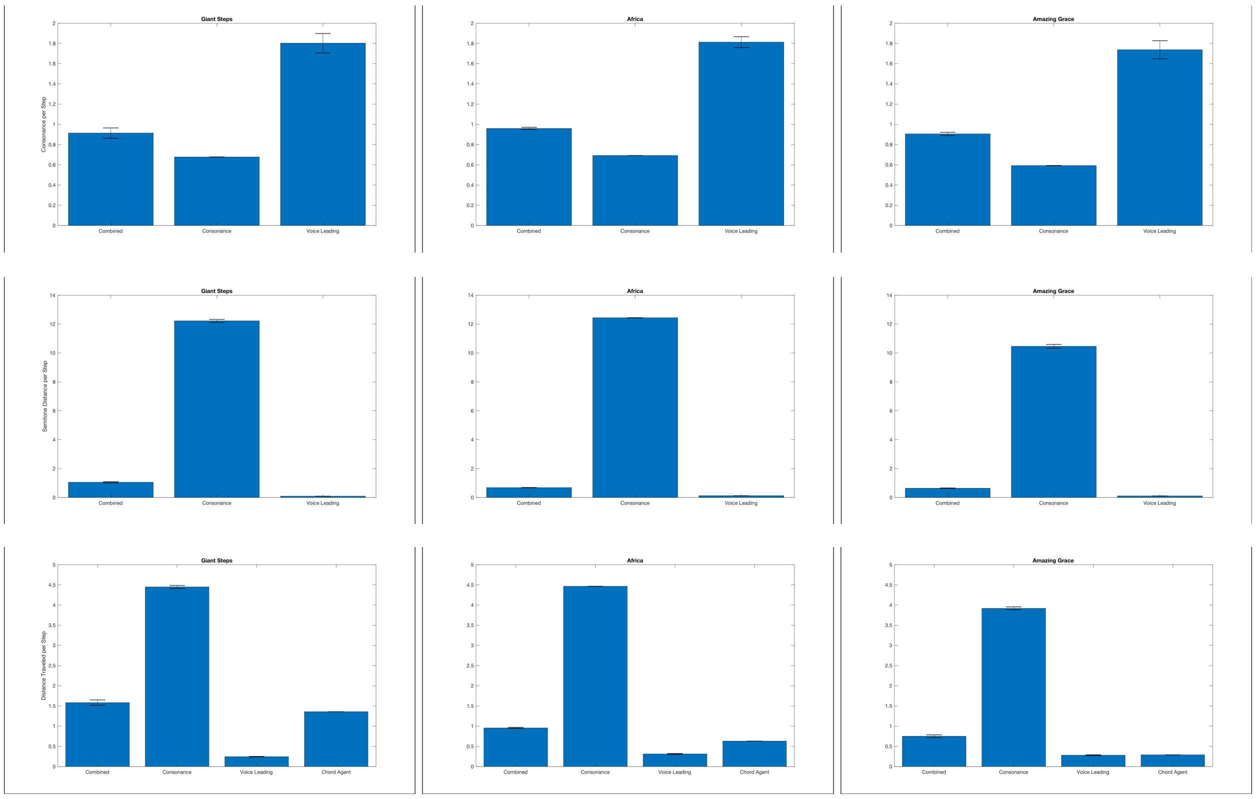

Figure 8. Consonance per Step (top), Voice Leading Distance per Step (middle), and Speed (bottom) for all three pieces (error bars represent 95% confidence intervals)

Here we see all of the relationships we expect to see - the Voice Leading Model scoring poorly in terms of Consonance per Step and the Consonance Model scoring poorly in terms of Voice Leading Distance per Step. One thing to note is the Voice Leading Distance per Step of the Consonance Model - it is lower on Amazing Grace than it is on Giant Steps or Africa. One possible reason for this is that the chords move more slowly in Amazing Grace (the *harmonic rhythm* is slower), as we can see reflected in the Speed metric. The slower harmonic rhythm means more sequences of repeated chords, allowing the Consonance Model to stay put, and improving its Voice Leading performance. Of course, because of another MATLAB sorting quirk, it’s equally liable to jump the octave unnecessarily.

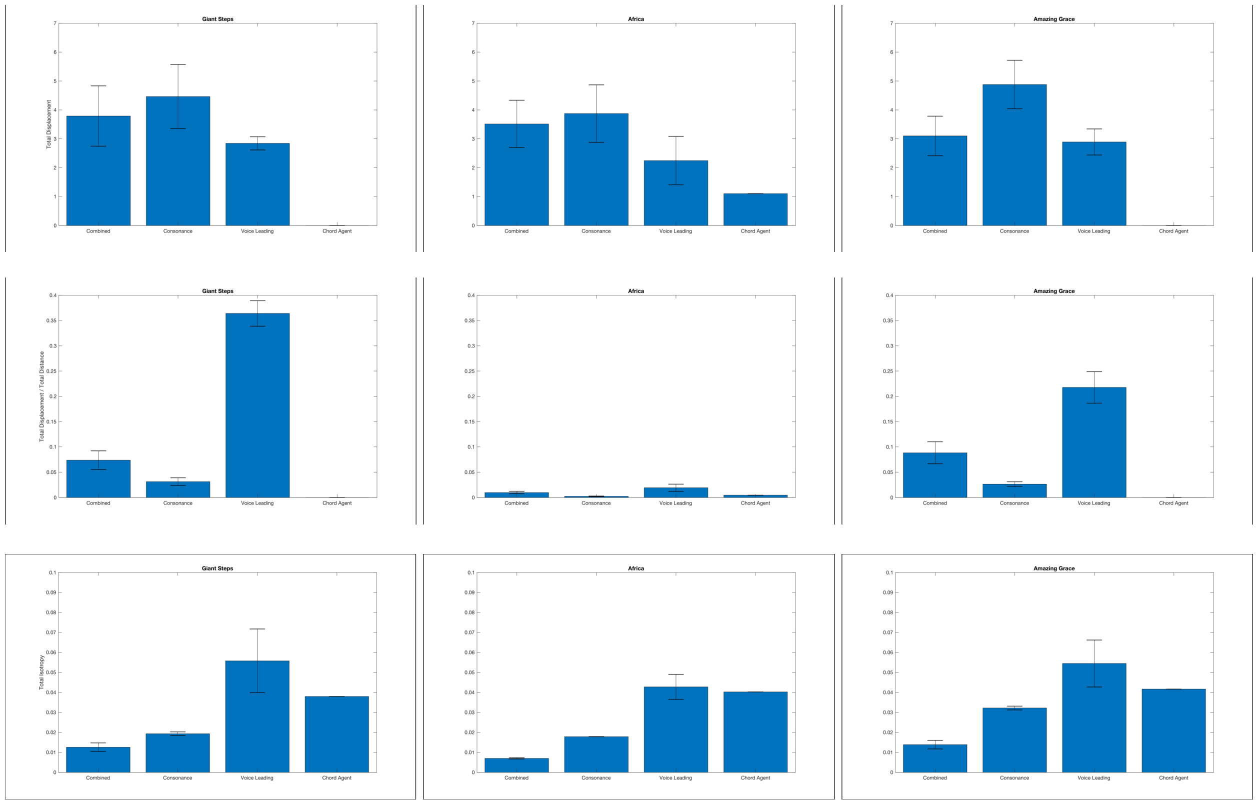

Figure 9. Displacement (top), Directionality Ratio (middle), and Isotropy (bottom) for all three pieces (error bars represent 95% confidence intervals)

The Directionality Ratio for Africa is much lower than for the other two songs for all three pitch models, which must nearly all be due to the fact that Africa is by far the longest of the three songs, and so has the greatest number of steps - thus, as the Speeds are comparable between the three songs, the longest Length (remember: DR is Displacement / Length). We can also see that in general the Displacement metrics for the three songs are very comparable, further confirming this. The Spiral Array is a relatively confined space for these agents to travel in; in an unrestricted system we might expect the Displacement to grow the longer the model runs, but in this case it has a ceiling.

Partly because of this fact, I find that as a measure of the general “dispersion” of the movement of the agents, Isotropy yields more nuanced results. For one, we can see how the chord agents compare to the other agents for every song, whereas because Giant Steps and Amazing Grace begin and end on the same chord, the Displacement for those two songs is zero, making the DR in turn zero. The differences are very small, but Amazing Grace seems to have the highest Isotropy for its chord agent, followed by Africa’s and then followed by Giant Steps’. I would have expected this result, as Amazing Grace has fewer chords, and as a result fewer directions to travel. This effect is what largely what drives the Consonance model pitch agents as well, as they will move to the most consonant note they can as soon as the chord changes.

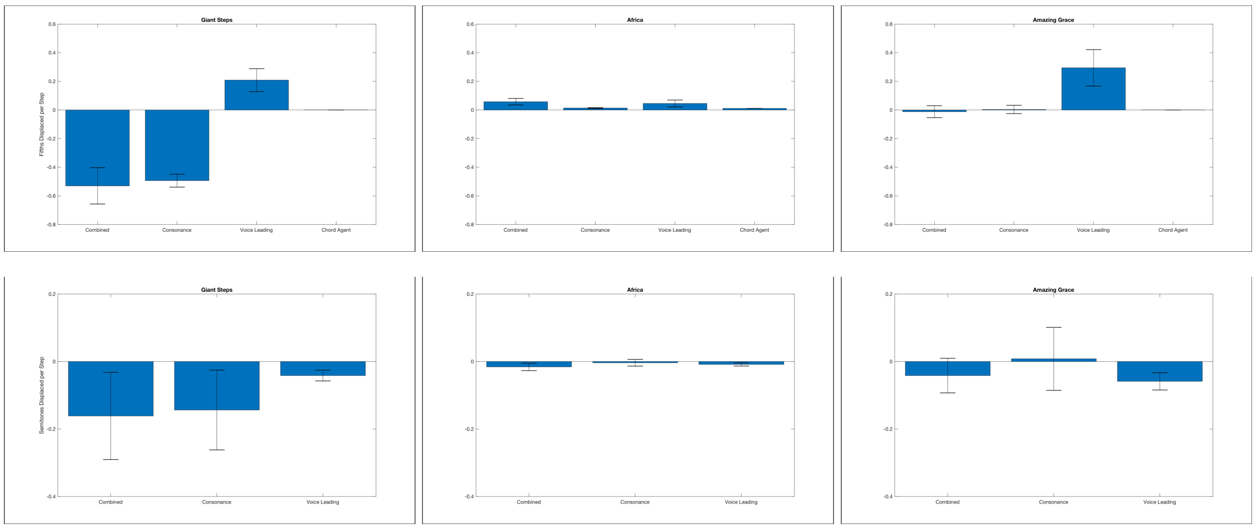

Figure 10. Fifth Displacement per Step (top), and Semitone Displacement per Step (bottom) for all three pieces (error bars represent 95% confidence intervals)

I’ve already covered the fact that because of MATLAB’s sorting, the model is slightly biased downwards in terms of semitone movement, and as a result upwards in terms of fifth movement. We can see this fact somewhat borne out here, but it is interesting to see how many cases in which the results overcame the bias. The chord agents in Giant Steps and Amazing Grace both wound up with a FD of zero, which is somewhat surprising intrinsically, but does make musical sense. Amazing Grace is heavily centered around the I, IV, and V chords, and there is an inherent symmetry there, in that moving up a fourth is the same thing as moving down a fifth, and vice versa. Giant Steps contains another level of symmetry - segmenting the octave into three equal parts by stepping between chords a major third apart. The Consonance and Combined model metrics for Giant Steps run strongly counter to the Fifth Displacement bias, though they are also the most extremely negative on Giant Steps for Semitone Displacement. One can really only guess why this might be, given the complexity of the pitch agent dynamics, but one possibility may be that there is a lot of downward whole step motion, which would be a negative semitone displacement but also a negative fifth displacement.

Discussion

In this project I created an agent-based model for the generation of tonal harmony, based partially on an analytical model created by Elaine Chew. For the purpose of analysis, I then created two variations on the model, such as the Voice Leading and Consonance models, which each act on a separate facet of my full model. The Consonance Model optimizes Consonance, without consideration for Voice Leading Distance, and the Voice Leading Model does the opposite.

The first goal in analyzing the findings of my project was to verify some basic assumptions about these models, such as the fact that the Consonance model does indeed optimize Consonance, and the same for the Voice Leading Model and metric. Many of the figures I presented above served mainly to confirm these assumptions: that the Consonance Model has the best performance in that metric, and likewise for the Voice Leading Model, with the Combined Model somewhere in between. While they did serve that purpose, they also gave me an appreciation for the Combined Model - how it operates with subtlety, and with relatively simple forces at work.

The Consonance Model represents one of these forces: while it does produce harmonious, consonant sounds, it does so with no consideration whatsoever for how the individual voices move, leading to many bizarre melodic leaps which no human could perform. The other force, the Voice Leading Model, is concerned with what the Consonance model ignores: how far in terms of musical note distances (not distance within the model's coordinate space) each voice travels. Ideally, each voice wouldn't move at all, so that's exactly what the Voice Leading Model does - until safeguards to prevent overlong holds force the voices to move. The metrics made clear the extent of the extreme, uncompromising behavior of these two models. Neither are successful on their own, but combined in just the right way, they temper each other in a way that produces a reasonable approximation of what a real composer might do, under certain circumstances.

Another promising insight came with the Isotropy metric: in every Isotropy figure, the Combined Model always had the best performance of any pitch model, losing only to the randomly-moving Key Agent. This means the model does well in avoiding repetitive decisions, which one can only imagine would have a positive impact on how interesting the musical results will be.

One last fact that the results revealed, one which I hoped would be the case, was that the figures generated from the chord sequence mode contained some musical insights into the songs themselves, and not just the model. For instance, the Speed of the Chord Agent can be used to infer the harmonic rhythm of the piece, and the Isotropy measure can be an indicator of the variety of chords used in the piece, both of which came from my analysis of the Amazing Grace chord sequence. I picked that song primarily as a stylistic contrast to the other two musical inputs, one being a complex jazz standard and the other a lengthy pop song. I inadvertently created another important contrast: Amazing Grace has a very simple chord progression, as is typical of its idiom, and the chords change fairly slowly. Both Africa and Giant Steps use a larger variety of chords, and the chords change more rapidly - often every two beats, whereas in Amazing Grace it is rare for the chords to change more often than every three beats.

The Speed metric of the Chord Agent reflects the rate of chord change in the piece for obvious reasons: whenever the Chord Agent moves, the chord has changed, and a piece with rapidly changing chords will have an often-moving chord agent. If I wanted in the future to design a metric specifically for this purpose I would make one small adjustment. The Speed metric is the Length metric divided by total steps, so it takes into account the distance the Chord agent traveled, not only the fact that it moved. In other words, chord agents which move to far-flung chord locations at each step will appear "faster" than chord agents which move just as often but to closely-related chords. This may bear further investigation, but it only confuses the question of how frequently chords move, so a measure which simply tracks how often the Chord Agent moves would be more appropriate.

As for the variety of chords, the Isotropy of the Chord Agent serves as a decent measure, because it measures the dispersion of the agent's movement. A large amount of movement in one direction will yield the highest possible value, and a small amount of movement in many directions will yield smaller values. If a song only contains three or four chords, as does Amazing Grace, there is going to be a fairy low limit to how many different directions of motion there can be for the Chord Agent. Although this does work in the case of my test inputs, one can imagine easily how using this metric for this purpose could have its flaws. Compare, for instance, a piece which moves to three chords in sequence, each a major third higher than the last - to a piece which moves between three different chords in sequence: up a fifth, down a major second, and up a fifth. They both have the same variety, but the piece moving in major thirds will have the maximum Isotropy, 1, as major-third relations in the Spiral Array are unique in that they are vertical on the z axis. This is an extreme example, but it demonstrates how the relative positions of the chord locations can have unforeseen consequences.

There are many ways my model could be improved in the future, but there is in particular one set of improvements I've already begun plans to implement in the near future. While there may always be room to improve the expressiveness of generative musical models like mine, for the time being I am relatively pleased with what I was able to do with the pitch agents in my model. I think they largely make good and realistic decisions. The Key Agent and Chord Agent in their current state are useful as a baseline for the purposes of generating figures and analyzing the behavior of the pitch agents, but I always intended them to have a more nuanced process themselves, and simply ran out of time.

I plan to implement a system whereby just as the Chord Agent responds to the Key Agent, and the pitch agents respond to the Chord Agent and one another, so the Chord Agent will respond to the pitch agents, and the Key Agent to the Chord Agent. The Chord Agent will still choose a chord that is as consonant as possible with the key, but it will also take into consideration voice leading relationships between potential chord targets and the current location of the pitch agents. This will help to create pleasing melodic lines, while at the same time keeping the chord to choices which work well within the key.

The Key Agent for its part will stay still, but it will be meanwhile keeping track of the most recent sequence of chords that the chord agent has chosen, looking for a sequence of chords which could be interpreted as something called a "pivot". Pivot chords are a consecutive sequence of chords which can be interpreted as inhabiting two different keys. For instance, "Em - G - C" could be "i - III - VI" in E minor, but it could also be "iii - V - I" in C major. They are called pivot chords because they are used to "pivot" between two different keys. They create an ambiguous harmonic moment, which composers take advantage of to change keys in such a way that the listener may not even notice the shift.

If the key agent detects such a sequence, it will change to the other key to which the pivot sequence could be interpreted as belonging. This change will then of course reflect in the chord agent's choices, as it must now choose chords that work well with this new key, and so on down the line to the pitch agents. I believe this will make the output of my model many times more interesting, and give it much more creative and expressive potential.

Bibliography

Chew, Elaine. “Towards a Mathematical Model Of Tonality.” *Massachusetts Institute of Technology*, 2000.

Eigenfeldt, Arne & Pasquier, Philippe. “Realtime Generation of Harmonic Progressions Using Constrained Markov Selection.” *Proceedings of the International Conference on Computational Creativity,* 2010.

Huepe, Cristián, et al. “Generating Music from Flocking Dynamics.” *Controls and Art*, Jan. 2014, pp. 155–179.

Lerdahl, Fred, and Ray Jackendoff. *A Generative Theory of Tonal Music.* The MIT Press, 2017.

Reynolds, Craig W. “Flocks, Herds and Schools: A Distributed Behavioral Model.” *Computer Graphics (Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '87)*, vol. 21, no. 4, July 1987, pp. 25–34.

Vicsek, Tamás, et al. “Novel Type of Phase Transition in a System of Self-Driven Particles.” *Physical Review Letters*, vol. 75, no. 6, 1995, pp. 1226–1229.