Maximally Even Rhythms and Chord Coloring

Jake Kerrigan
COMP 163

December 2025

1 Introduction

This paper is a structured report on Godfried Toussaint's work *"Computational Geometric Aspects of Rhythm, Melody, and Voice-Leading."* The goal is to summarize the core geometric models used to measure rhythmic (and chordal) evenness and interpret their mathematical and musical meaning.

More broadly, this paper will explore how rhythms can be modeled using geometric and computational methods. A central idea is that rhythmic patterns can be represented as points arranged on a circle, allowing for the mathematical study of distance, symmetry, and evenness.

1.1 Rhythmic forms and the Con Clave Rhythm

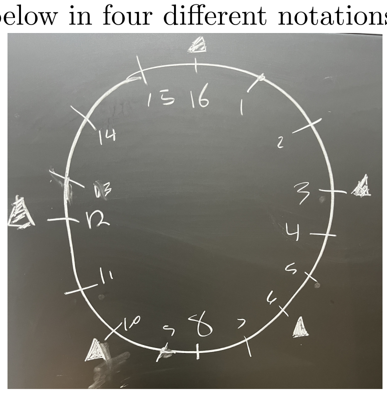
Different rhythm notations are used to describe the same rhythmic structure. There are nine introduced in the text, but the most useful include box-notation, binary sequences, interval-based, and geometric circular representations.

As a simplified example, if we consider just straight quarter notes in 4/4, what a layman can imagine as simply tapping their foot to a song, or as the monotone drive of the bass drum in most songs, it would be notated box notation as:

`[x...x...x...x...]`

Note the eveness of this rhythm, we'll revisit this shortly.

Godfried Toussaint introduces the Cuban *son* Clave rhythm as the motivating example. The *clave son* rhythm is represented below in four different notations.



- In geometric circular representation (pictured above)

- In box form: [x..x..x...x.x...] - each x represents a beat and each . represents a rest
- In binary form as: [1001001000101000] - analogous to box form.
- In interval-based form: [3, 3, 4, 2, 4] - represents the space of between each onset (including the onset itself)

We define its rhythm class as: R[5,16] which stipulates:

- 5 onsets
- 16 total pulses

Pulses are the total number of slots or spaces in the rhythm not occupied by an onset

Onsets are the number of beats or notes. We use the term onset because beats and notes always have a duration assigned to them in traditional musical notation, but for our purposes, we assume all beats or notes have no duration and only a starting point. This is made clear by the binary notation, where at each pulse in our space, there is either an onset (1) or a rest (0), and our array is of size # pulses (corresponding with the total time of the rhythm). As you'll see, the "onset" of each of the beats and not its duration is what we'll use to measure its geometric properties.

As a final piece of notation, we use the word clave to describe an abstract rhythm in the context of pulses and onsets.

2 Measures of Rhythmic Evenness

In order for a rhythm to provide drive and forward motion, as is necessary in dance, it must be even. In other words, the audience should not be left waiting for the next onset or barraged by a group of onsets close together. This section will explore three different methods of measuring the evenness of rhythms.

2.1 Maximally even rhythms

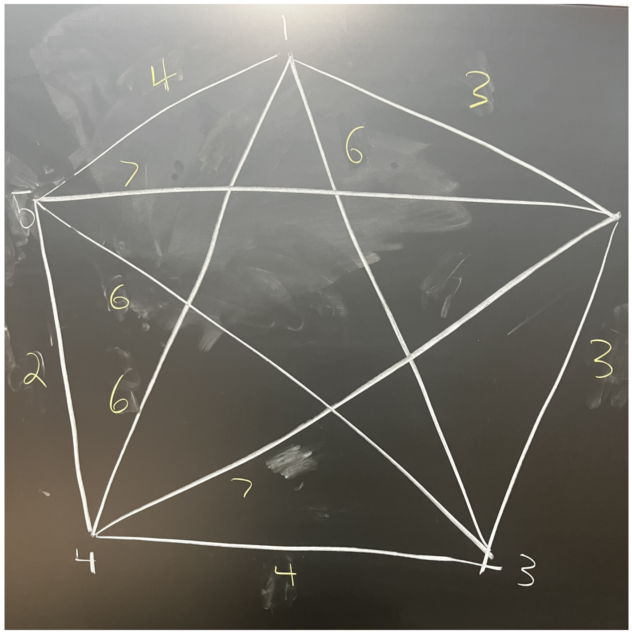
Regardless of measure, a maximally even rhythm is always the same. For example the first rhythm we considered is maximally even.

```
[x...x...x...x...]
```

However, as all onsets are equally spaced, without any other layering, this rhythm will not be interesting to listen to. To make maximal even rhythms less predictable, we consider rhythmic classes R[X,Y] where X and Y are coprime. Under such constraints, there is no trivial maximal even rhythm; rather, distinct rhythmic structures arise by trying to maximize evenness.

2.2 Sum of Geodesic Evenness

The first measure of evens measures the distance between each onset using geodesics on the circle they're plotted on. There are $\binom{n}{2}$ measurements, where n is the number of onsets in the rhythm. The distance between two onsets is the shortest path along the circle between the two onsets. The *con clave* the rhythm has a geodesic evenness value of 48. Calculation depicted below:



There are $\binom{5}{2} = 10$ geodesics to calculate, displayed by using a weighted graph where nodes (labeled by index in white) represent onsets and edges (weighted in yellow) represent geodesic distances between nodes. The resulting evenness score is: $2+3+3+4+4+6+6+6+7+7 = 48$

This measure, however, doesn't seem to capture all evenness in a rhythm. For example, the *Rumba clave* and the *Bossa-Nova clave* both also have evenness values of 48 under geodeic measurements. But the *Bossa-Nova* clave is clearly more even than the *Rumba* clave. and *Son* clave.

- Son:

  `[x..x..x...x.x...]` $= 48$

- Rumba:

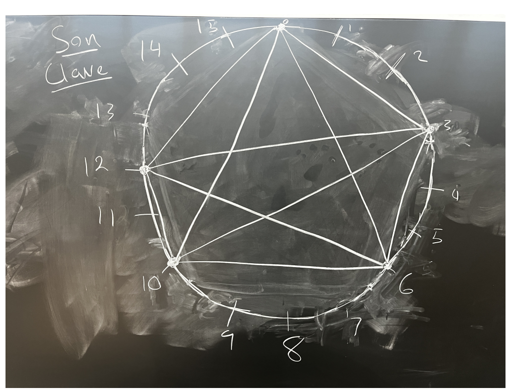
  `[x..x...x..x.x...]` $= 48$

- Bossa:

  `[x..x..x...x..x..]` $= 48$

2.3 Sum of Distances Evenness

Our second metric of evenness is similar to the geodesic evenness but more discriminant. Instead of considering the length of the geodesic between each pair of onsets, we'll consider the actual Euclidean distance between onsets in geometric form. The lengths are depicted below.



- Bossa:

$$[\texttt{x..x..x...x..x..}] = 15.3252$$

- Son:

$$[\texttt{x..x..x...x.x...}] = 15.2825$$

- Rumba:

$$[\texttt{x..x...x..x.x...}] = 15.2115$$

- Shiko:

$$[\texttt{x...x.x...x.x...}] = 15.1644$$

- Gahu:

$$[\texttt{x..x..x...x...x.}] = 15.1361$$

- Soukous:

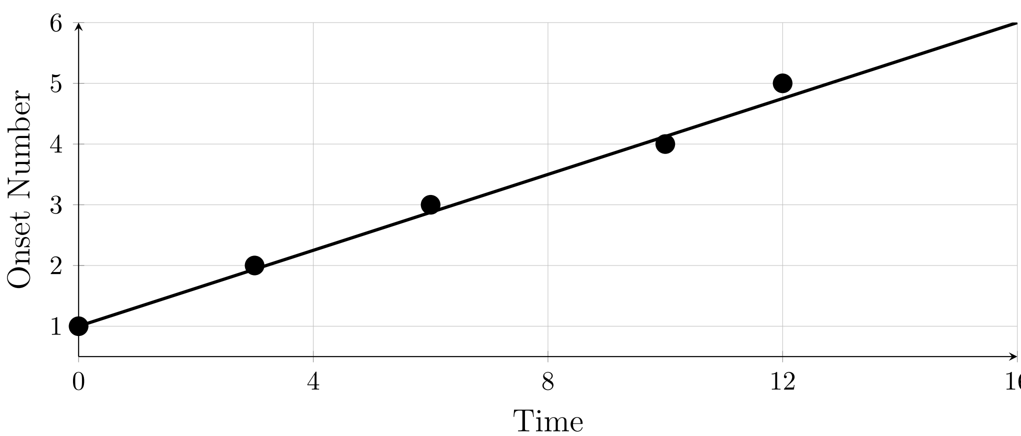
$$[\texttt{x..x..x...xx....}] = 15.0096$$

These numbers align much better with our intuition. The *bossa* clave, which seemed the most even, is maximal under this evenness metric.

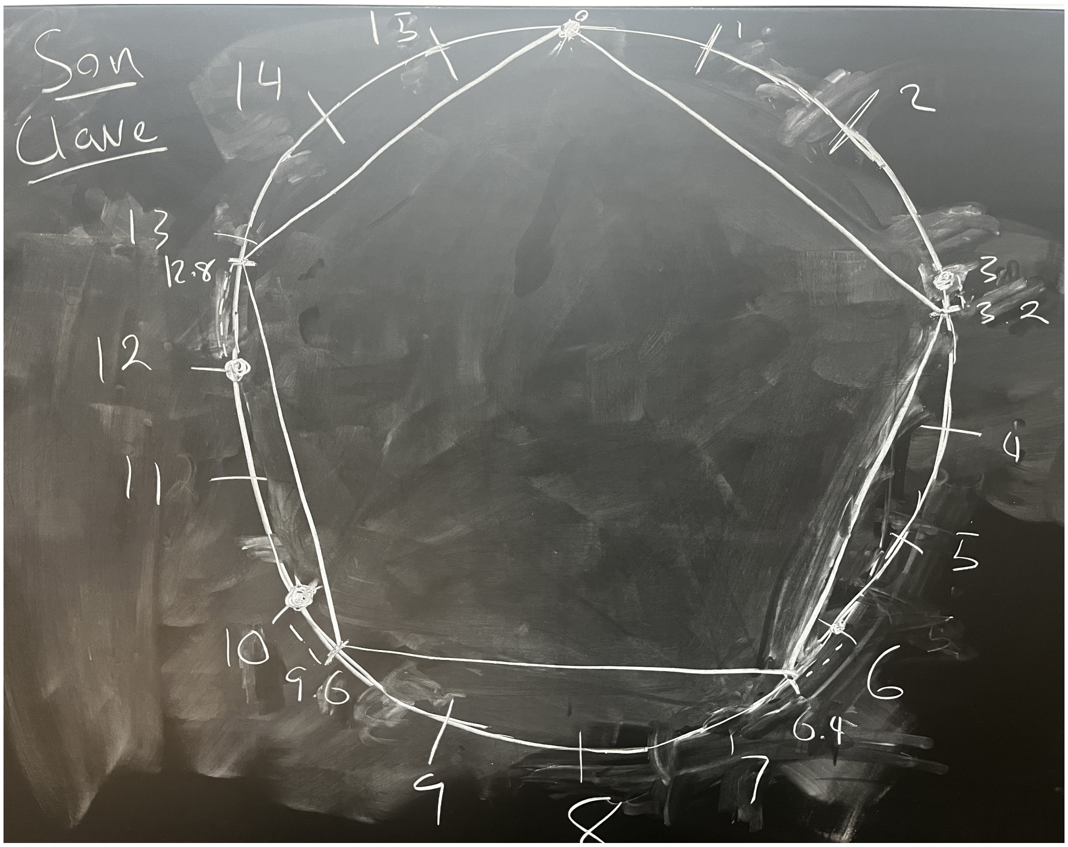
Another example the *Bembé* clave [x.x.xx.x.x.x], is maximally even among all 7 onset 12 pulse rhythms. However, this metric is time-intensive to calculate. Worst case it takes $O(n + k^2)$ time where n is the number of pulses (time to read in rhythm) and k is the number of onsets. This means determining the maximal rhythm will be hard for rhythm classes with many onsets and pulses, and if extrapolating to calculate evenness for the distributions of markers in DNA sequences, for example, both k and n are in the thousands.

2.4 Linear Regression Evenness

A more straightforward and easier to calculate measure of evenness of a rhythm is the linear regression evenness method! We consider the # pulses X # onsets, graph, where each onset is plotted on which pulse it is played, and its height is its index (the first onset has height 1, and so on).



Maximally even rhythms will deviate as little as possible from the regression line. But how do we measure deviation? We could choose vertical, horizontal, or orthogonal distance to the line. As we're measuring a deviation in time, the most fitting metric would be horizontal. Under this metric, we can imagine our regression line as a normal k-polygon in the n-pulse cyclic lattice. We can then calculate how much our rhythm deviates by calculating the geodesic distance of each onset to the nearest vertex of the normal polygon. The calculation is pictured below using the normal pentagon for the Son Clave rhythm:



Adding up the distances depicted by the dotted lines, we calculate the *son* clave's evenness score as: $0.2 + 0.4 + 0.4 + 0.8 = 1.8$

This simplified metric makes a similar distinction as our Sum of Distance Evenness metric discussed in the previous section (2.3), but in linear time.

- Bossa:

  `[x..x..x...x..x..]` $= 1.2$

- Son:

  `[x..x..x...x.x...]` $= 1.8$

- Rumba:

  `[x..x...x..x.x...]` $= 2.0$

- Shiko:

  `[x...x.x...x.x...]` $= 2.4$

- Gahu:

  `[x..x..x...x...x.]` $= 2.2$

- Soukous:

  `[x..x..x...xx....]` $= 2.8$

3 Maximally Even Chord Coloring Implementation

All methods we used for measurements or rhythms can be used for evenness of melodies or chords as well; in fact, many of the methods showcased were originally developed to analyze melodies and chords but have been recently readapted for rhythms. From a musician's standpoint, these metrics can be used as a compositional technique (finding maximum evenness). For my prpoject, I built a program to automatically find maximally even chords (using Linear Regression Evenness) for a given pulse count and onset count. Musicians, however, often have root notes of a chord they will always use or that fit well into the song. My implementation allows the user to dictate certain pitches to be included in their chord and adds in notes (from a given mode) to "color" the chord.

3.1 Functionality

The problem of coloring a chord as evenly as possible while using a base set of notes is solved using dynamic programming. We assign each vertex from the normal polygon to a note in the coloring, and try to minimize this distance. A definition of the subproblem is as follows:

$$color(v, b, c) = \begin{cases} c & \text{if } v = 0 \\ c + b[0] & \text{if } v[0] < b \\ \min\{c + b[0], c + v[0]\} & \text{if } v[0] > b \end{cases}$$

Intuitively, we assigned each vertex to a note in the colored chord. If the normal vertex we're trying to assign is further in our circle representation than the smallest base note, any maximally even chord must assign that vertex to that node. If not, we try assigning the closet note in our mode to the vertex and not assigning the base and take the minimum of the two.

3.2 User Guide

The user inputs which notes they want included in their base chord, what mode they want to select notes from, and how many total onsets (notes) they want in the chord. The implementation then adds color notes to the chord to increase its total number of notes while making it maximally even.

Submitted Files:

- proto.py - the first version of chord colorer. Base chord, onsets, pulses, and mode are all set as variables in the code and upon running the script outputs the colored chord in the terminal.
- server.py - Provides the same functionality as proto.py but hosts a webserver using index.html that allows the user to enter the Base chord, onsets, mode, and outputs the chord in the geometric circulate form overlayed over the normal plygon. After running open up http://localhost:8000 in a webrowser of your choice.

4 References

- Toussaint, G. (2009). *Computational Geometric Aspects of Rhythm, Melody, and Voice-Leading.* Computational Geometry: Theory and Applications.