Lab: 12-Tone Music Theory

CSI 3305: Introduction to Computational Thinking

February 9, 2011

1 Introduction

The Tonal System of music was the prevailing method of music composition throughout the period from around 1650-1900, and was used by such artists as J.S. Bach, Mozart, Beethoven, Chopin, and Brahms, among others, to create some of the timeless masterworks that we continue to listen to and appreciate today.

Throughout the 19th century, however, composers became increasingly interested in both personal expression and musical programs emphasizing extra-musical importance. These aesthetic interests manifested themselves in an increased use of dissonance and a gradual breakdown in the syntax of the tonal system.

By the early 20th century, many composers had moved so far in the direction of chromaticism that the pitch hierarchies that were the hallmark of the tonal system were no longer in operation. In fact, many of the basic mechanics of the tonal system such as key signatures and major and minor scales and triads were no longer found in the music of these composers.

Even the term "dissonance" with its pejorative connotations was dispensed with. Now, all intervals and pitch collections were considered consonant, albeit, some more consonant than others. In a sense, anything was now possible. A composer no longer had to adhere to a well-established musical language, but could create his or her own language, independent of what his or her contemporaries were doing. Composers were now free to explore new territory, not only in the domain of pitch relationships (melody and harmony), but in areas such as rhythm, timbre, and form as well.

As a result, the older methods of viewing and analyzing music no longer applied to this new "post-tonal" style. Therefore, music theorists had to devise new analytical tools for describing and discussing the music. One of the most successful approaches developed is known as "set theory." This technique views collections of discrete pitch classes of various cardinalities and provides a means of cataloging them and subsequently comparing them with other collections. By analyzing and comparing the sets used in a given composition, theorists can gain valuable insight into the piece and the composer's compositional method. The "set" is a collection of unordered pitch-classes[1] and may be presented as either a melodic or harmonic entity in a given composition. Theorists typically look at certain "promising-looking" pitch collections within a composition, determine the set-class[2] that the given collection belongs to, and then look for other pitch collections that

[1] A "pitch-class" accounts for all of the pitches of the same alphabet name and enharmonic spellings, regardless of register (octave position). Therefore, all C#;s and Db's belong to the same pitch-class.

[2] A "set-class" refers to all pitch-class sets that share the same prime form. Prime forms are expressed in integer notation (mod. 12, since there are 12 pitch-classes in the chromatic scale) and are transposed to begin with 0. Therefore, the set (3, 4, 8) has a prime form of [0,1,5]. We get this simply by "transposing" the set to 0 (T_9). Likewise, the set (5, 9, 10) also has a prime form of [0,1,5]. To understand this, we must accept the concept of inversional equivalence, which states that a set-type and its inversion are the same due to the preservation of intervallic content under inversion. Therefore, if we invert (5, 9, 10) about the zero axis (T_0I), we get the inversionally equivalent set, (7, 3, 2). This is then transposed (T_10) to get (5, 1, 0), and then reordered into its prime form beginning on 0: [0,1,5]. By reducing a pitch-class set to its prime form one can easily categorize set-classes.

are members of the same set-class. This takes a fair amount of time since a given set-class can appear in many different guises. The reason for this is due to several factors including transpositional equivalence, inversional equivalence, and the fact that the set members may appear in any order, not to mention that the composer is likely using a given set both melodically (linearly) and harmonically (vertically) within a piece.

A computer program that enables a theorist to quickly identify a given set-class and its various iterations throughout a composition would be an extremely useful tool. If a theorist believed that the set-class [0,1,4][3] were significant within a piece and could then simply type in the given set-class and have the program search for all iterations of [0,1,4] throughout the composition, the theorist would likely be able to quickly tell whether or not the given set-class was important to the construction of the work. If it were determined that the set-class was not that important based perhaps on a limited number of occurrences, then the theorist could simply search for iterations of another set-class type.

If a program were to search for occurrences of [0,1,4] in the linear domain, then it would need to search through segments of pitch classes. For example, a given melodic segment when notated in numeric notation (C=0, C#/Db=1, D=2, etc.), may appear as 4, 1, 0, 9, 5, 6. The first three pitches of the segment are clearly members of the unordered set-class [0,1,4]. However, the segment 1, 0, 9 is also a member of this set-class due to both transpositional and inversional equivalence. A third iteration of the [0,1,4] set-class is found in the segment 9, 5, 6.

The program would also need to search through vertical simultaneities. For example, a chord containing the pitch classes 10, 5, 2, 9, 6, 1, contains numerous occurrences of [0,1,4]: (10,9,6), (10,9,1), (10,2,1), (5,2,6), (5,2,1), (5,9,6). What this analysis reveals to the theorist is that each member of the simultaneity belongs to three different [0,1,4] groupings. This information would likely influence how the analyst would consider other simultaneities within the piece.

Another possibility is that a set-class may be partitioned among instrumental voices. Therefore, one pitch may be found in the violin, another in the clarinet, and a third in the cello, none of which are played simultaneously. The program should also have the ability to search for and illuminate set iterations that are not exclusively linear or vertical.

Such a computer program would provide the music theorist with a powerful tool for analyzing the music of the post-tonal era. Not only would it quickly reveal the frequency with which a certain set-class occurs within a piece, but it would also reveal to the theorist where these sets occur and if and how they are interlocked and related, information that may reveal a great deal about the compositional method behind the piece.

2 Problem Statement

Your task is to build a set-class analysis tool for midi files. Your program will take in a midi file as input, look through the file for occurrences of triad set-classes and report these (with their locations) to the user. Your program should also be capable of accepting a prime form set-class and reporting where it occurs in the piece.

3 Tools

You will be using the Python programming language and Python interpreter for this lab. The code will be written using a text editor, and the interpreter will be called using the command line.

[3] Brackets [ ] are used to denote the prime forms of unordered set-classes, and parentheses are used to denote simple pitch-class sets.

4 Programming

4.1 Midi Parser and Prime Form Calculator

A prime form calculator, which takes as input a set of notes and returns its prime form, is provided for you as is a simple midi parser that takes in a midi file and calculates note events (where they occur and their duration) and midi tempo data. These are available on the course website. Using these scripts, you will be able to select a group of notes from the parsed midi data, submit them to the prime form calculator, and determine which prime form set-class they belong to.

Part of your task for this lab is determining a strategy for searching a given work for both linear and vertical occurrences of set-classes. Given the tools available to you (such as note data), create a strategy for searching groups of notes for set-class occurrences.

As a help in this lab, you are given a midi.py file, which contains code showing how to access the note data, the tempo data and the prime form calculator. The note data is contained as tuples in a python list, with the following form:

```
(track_index, channel_index, pitch, velocity, keyDownTime, keyUpTime)
```

The pitch info denotes the note, and the `keyDownTime` and `keyUpTime` allow you to know when a note starts and ends, so that you can determine note overlap.

To call the program, using an input file and a prime form set-class to search for, you use the following command:

```
midi.py -i music.mid  -p[1,0,5]
```

The "-i" option allows you to input a midi file, in this case music.mid. The "-p" option allows you to search for the prime form [1,0,5].

5 Questions

1. Describe your strategy for finding prominent set-classes within a work.
2. What are some advantages to using a computer to accomplish this task?
3. What advantages (if any) would a human have in performing this same task?