Algorithmic Arrangement for Close Jazz Harmonies

Abstract

Since certain jazz harmonies follow very specific rules for construction, we can algorithmically produce the midi of a harmonized melody given input parameters of melody, chords, harmony type, and key.

Overview

A musical arrangement usually starts with an existing melody, song, or composition, and produces a new composition which changes certain elements of the original, such as the instrumentation, key, and even harmonies and sometimes the very melody itself. A standard form is the Theme and Variations, in which the composer starts with a tune, the theme, and then generates variations that might vary in key, tempo, time signature, harmony, or other features.

Arrangements often change the instrumentation of a piece. For example, Liszt created virtuoso piano versions of the nine Beethoven symphonies. He wanted to be able to perform the symphonies on his own. It is common to create an arrangement of an existing piece to suit the structure of a specific musical ensemble.

The repertoire of a cappella singing is largely derived or arranged from music that was not originally meant for a cappella groups. Creating new arrangements is thus the primary way of expanding the a cappella repertoire.

One style of a cappella arrangements is close jazz harmony. In this project, I automated that process. The input to the program is a melody, associated chords, and the output would be a vocal arrangement (or realized chords a pianist might use).

The normal process of arranging and harmonizing a song is laborious and time consuming. However, this project automates the process based on common jazz harmony, theory, and style conventions. I created an algorithm for arranging a few close jazz harmonies and implement it using Euterpea.

Harmony Types

The program deals with three main categories of harmony: 4 way close (otherwise known as barbershop), drop 2, and chordal. Both 4 way close, and drop 2 harmonize melodies note by note based on the scale degree of each note. They harmonize 2 different types of scales, bebop scale which contains an additional sharp 5, and dominant bebop scale which contains an additional flat 7. With the simple major bebop scale, they assign a major I6 chord to scale degrees 1, 3, 5, and 6, and a dim7 chord to the remaining scale degrees. With the dominant major bebop scale, they
assign a major I7 chord to scale degrees 1, 3, 5, and flat 7, and a dim7 chord to the remaining scale degrees. 4 way close simply takes the closest harmonies below the melody note, while drop 2 puts the 2\textsuperscript{nd} harmony from the top an octave down.

Below are sample C major and A minor scales harmonized by the program in the 4 main styles by the automatic arranging program.

And the dominant scales:
So with the two scales, and two melody harmonizing types, we already have 4 harmony types. For each of those we perform substitutions, replacing one harmony note of each chord with a note a major second or so different, giving us 8 harmonies. Additionally we deal with non scale degrees by giving them no harmonies.

Below are sample C major and A minor scales harmonized by the program using substitution in the 4 main styles by the automatic arranging program.
<Drop 2 harmonization of A minor scale with substitutions>

And the dominant scales with substitutions:

<4 way close dominant harmonization of C major scale with substitutions>

<4 way close dominant harmonization of A minor scale with substitution>

<Drop 2 dominant harmonization of C major scale with substitutions>

<Drop 2 dominant harmonization of A minor scale with substitutions>

The final harmony category, chordal, simply takes a list of chords, realizes them, and puts the result parallel to the given melody.

Here is a sample of all quarter, all C tonic chords built of all the possible qualities using the chordal function:

Levels of Harmonizing

In the program, there are 3 levels of harmonizing functions. The top level defines a switch on the harmony type and passes the melody down. The second level defines a switch on Euterpea’s music constructors (and controls the frequency of random substitutions for substitution harmony types). Finally, the bottom level defines a switch on the scale degree, which is finally fully harmonized.
The chordal harmonizing has a slightly different format. There is a top level switch to it, but it leaves the melody at the top level to be parallelized with the chords constructed at lower levels. In the subsequent levels, a list of chords is taken in, which are then built at the bottom level.

**Voice Leading: Evaluation**

In order to improve any potential arrangement, we would have to address the problem of voice leading. Traditional voice leading evaluations look at how perfect intervals are arrived at. If they are arrived at by parallel or similar motion, it is considered bad voice leading. I created a function to put a voice leading score to a set of 4 part harmonies.

However, revoicing to improve is a more amorphous, ill defined task. It would require a framework to search for and replace bad voice leading, and each replacement affects the voice leading to and from that note. Consequently, it proved too large for the scope of this project.

**Musical User Interface**

The MUI allows specification of input and saving a midi of the harmonized result. Specifically, you can enter notes of a melody and chords via two textboxes, select harmony type, key, and mode, and specify a filename to save a midi as.

See below a screenshot of the interface.
Alternately, it can be controlled from the command line, directly calling the “harmonize” function. For example

```
harmonize [C Maj qn] (c 5 qn) (C, Major) FourWayClose 0
harmonize [C Maj qn] (c 5 qn) (C, Major) Chordal 0
```

**Use Cases**
This program could be used to produce vocal arrangements, though it is more reasonable to generate just a first draft than a finished product. Additionally, a pianist might use the program to realize chords or visualize a drop 2 or 4 way close harmonization.

Acknowledgements

I would like to thank Professor Paul Hudak, Dan Winograd-Cort, and Donya Quick for their help on this project.

References