Building Algorithmic Software for Musical Harmony

Objectives

- Implement algorithmic process of musical harmonization
- Develop platform for exploration of alternate harmonic theories

Resources

Software
- Max 8 audiovisual programming environment
- JavaScript

Graph Theory
- Note options are vertices, intervallic displacements between options are edges
- Dijkstra’s Shortest Path First algorithm

Counterpoint
- Gradus ad Parnassum (1725) by Johann Joseph Fux

Methods

The final prototype system consists of an adaptation of Dijkstra’s Shortest Path First algorithm implemented in a JavaScript module embedded in a Max 8 patcher. The module has a three-step processing method:

1. Generate the list of note options based on allowed vertical intervals above the input cantus firmus (a provided unchangeable melody)
2. Calculate the intervals between each pair of note options, setting prohibited motions to infinity and modifying others for exceptions and preferences
3. Run the Shortest Path First algorithm to maximize stepwise and oblique motion

The cantus firmus is input as a list of scale indices, and the counterpoint is output as a list of chromatic semitone offsets. Both lines are sent to a playback engine and a musical notation display.

Results

The prototype system successfully generates a musical line that obeys the constraints of counterpoint and sufficiently harmonizes the cantus firmus. However, the output often lacks musical interest and contour, since it is chosen to minimize motion. It also responds unpredictably to inputs that do not begin and end on the tonic or do not end by step, so all inputs must be chosen to ensure compatibility.

Conclusion

This project sets the foundation for further explorations of musical graph theory and algorithmically generated harmony that will expand to more complex textures.