Allegheny College
Department of Computer Science
&
Department of Music

Senior Thesis

Computer Aided Music Composition

by

Jacob Sutter

Allegheny College
Computer Science

Project Supervisors: **Dr. Janyl Jumadinova & Dr. Douglas Jurs**
Co-Supervisors: **Dr. Douglas Lumen & Dr. James Niblock**

December 10, 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The involvement of computers in the creation, performance, distribution, and analysis is most certainly something that most everyone is aware of regardless of their experience in using these technologies. Since the creation of the computer, musicians and others have found new and innovative ways to integrate technology into the field of music. As the computer continues to advance, we will find new methods of integration and ways to improve upon our existing means.

Currently, in the field of computer science, artificial intelligence is being heavily developed and researched. Its uses, implications, practicality, and future are an important topic of discussion in our lives today. Specifically related to the uses of AI, artists and musicians are examining ways in which it can be applied to their own processes. When the idea of computers creating art and music with computers was conceived, the field of AI was nonexistent. Today, the state of AI is much improved. It is now possible to train computers to create and develop original works.

1.1 Background

The earliest cited example of musical composition that was created using a computer was Hiller and Isaacson's *Illiac Suite* [6]. This work was composed in 1956 by Lejaren Hiller and Leonard Isaacson [1]. The following diagram shows the rules that were programmed into the system to guide the composition [10]. They built the system to follow an early format of strict counterpoint [10].

Figure 1.1: Diagram from Hiller and Isaacson's published paper on the Illiac Suite showing counterpoint rules [10]

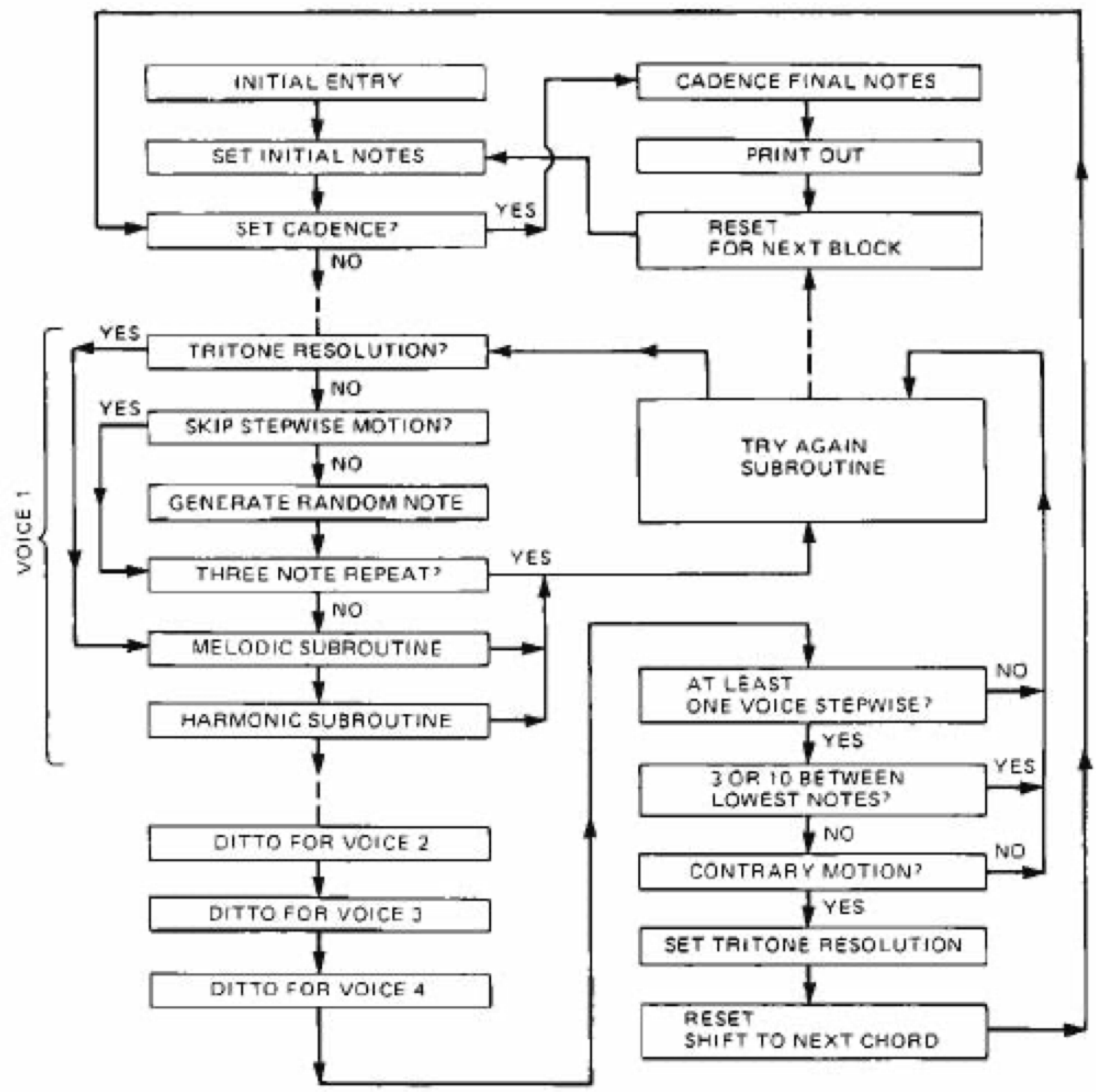


Figure 1.1
Experiment 2: Main routine for strict counterpoint.

An example of a later approach to computer music composition was Hiller and John Cage's HPSCHD in 1968 [1]. This work was composed by using computers that played Mozart's Musical Dice Game and altered the results using the rules of the Chinese oracle, I Ching [1].

The development of each of these works required extensive music composition and computer programming knowledge to complete. Lejaren Hiller was a composer by trade and Leonard Isaacson was a professor of computer science [1]. Neither of these qualifications are applicable to describe the general population of people that are interested in music. Music has an infinitely deep reaching appeal to practically all humans and even many animals.

With that being said, it only makes sense then that people who are not educated in either music or computer science should be provided with a way to create music. By not providing the means for these people to express themselves musically, we miss out on a large amount of creative output and new musical compositions. Rather than requiring these people to try to gain access to training, we can assist them in musical composition with computer based technology.

1.2 Motivation

Imagine that the piano was an instrument that could only be played if you were over seven feet tall. If the instrument were designed like this, only a very small portion of the population would be able to make use of it. The piano, however, is one of the most universally played instruments because it was not designed in a way that excludes lots of people from using it.

Currently, the tools that exist for computer assisted music composition are only accessible to musicians with high level computer science training [15]. All of them either have generally inaccessible command line interfaces or require the user to be trained in music composition to be able to understand or use the output of the program. Neither of these situation cater to the general person who might find some interest in being able to express themselves by composing with these tools.

Due to the research heavy focus around the creation of these tools, not a single one was designed to be placed in the hands of the average user. Someone who may have always wanted to create music might not have access to the training, schooling, tools, or opportunities that are currently required to be able to compose, record, or play music. Giving these people a tool that can help them through the composition process and assist them in notating their music is an important first step in being able to make creating music more accessible to the average person.

1.3 Goals of the Project

The goal of this project was to create a computer based musical composition tool that can be used by any person and does not require music or computer science knowledge. Through the careful design and testing of a user interface built over the high level functions of the composer, this project set out to create a tool that makes music composition accessible to everyone.

In order to make this a reality, the project was broken into three main components.

1. Computer Composer Development
2. User Interface Development
3. User Interface Study

1.3.1 Computer Composer Development

This part of the project consisted of developing the backend functions of the computer composer. Several different existing Python libraries were combined to create the composer and to process its output. The basic primary functions of the composer include melody generation, harmonization, key and mood analysis, and musical notation generation.

Each of these is a necessary part of the tool in order to make the composer into one that covers all of the bases of musical composition. After the generation of the musical output, the user is free to do what they would like in changing and altering it either with the composer or externally.

1.3.2 User Interface Development

This next part of the project consisted of building the interface that the user would use to interact with the composer. There are already several tools that perform some combination of the functions of the composer that were listed above, but none of them have this interface component.

In order to meet the overall goal and motivation for this project, the interface had to be designed in a way that could, in essence, explain itself. Rather than having long lists of instructions and using lost of music theory terminology, the interface has to present itself in a way that each element explains its own function in a completely visual way. This is where the bulk of the work of the project was done.

1.3.3 User Interface Study

Following the development of the composer and the UI, it was necessary to test the effectiveness of the UI and if it met the idea behind the project through a user study. Users with varying degrees of musical knowledge were asked about their experiences through interacting with the tool. The overall success of the interface design was gauged through a series of survey questions.

After getting feedback from users of the tool, some revisions were performed to the interface to clear up uncertainties and confusions that the users had. This is an important step in UI development since not every user of the interface is going to see it or interact with it in the same way.

1.4 Thesis Outline

The following outlines what is contained within this thesis document.

1.4.1 Introduction

This section of the document provides the situating of the work within the disciplines of music and computer science. It contains details about the history of computer composed music and some background into the associated problems. It also discusses the motivation for the project and its overall goals.

1.4.2 Related Work

This section of the thesis document provides a discussion into the other published works related to computer composed music. It discusses the details and ideas used in these projects as well as some of their shortcomings as related to the motivation behind this project.

1.4.3 Method of Approach

This section details the approach taken to solve the problem discussed during the introduction. It explains the composer development, UI development, and the user study. It further discusses the reasons for these parts of the project and what was entailed in their completion.

1.4.4 Experimental Results

This section of the document provides a discussion of the results of the user study and the overall success and failures of the project. It goes through the users' responses to the questions in the user interface study and what their responses mean for the evaluation of the interface.

1.4.5 Discussion and Future Work

This section of the thesis document discusses the overall impact of this project and what might be done in the future to improve it. Neither of these are limited to strictly the research based impact of this project as the goal is to move the tool beyond the lab and into the average user's hands.

Chapter 2

Related Work

As the field of AI has grown, more and more researchers are looking into the extent of the capabilities of AI. The creation of music with computers has gained increased attention within the last decade, however, this does not mean that these tools have gained popularity among users. Quite a few of these composers that have been developed, but only a very small number of them have seen industry use and almost none of them have been used outside of research or these niche areas in industry.

The reason for this primarily is that the focus of researchers when creating these tools is how advanced the composer can be and what new ways can AI be employed to create more original music. Their focus is not at all on how these tools can be made it a way that they might find a larger user base or even a user base at all. This limits what might come of these technologies now and in the future.

Setting this aside, however, the researchers that have developed these composers have created some very impressive tools capable of creating very convincing music. This project aims more to guide the user through the process of composing music which is different than majority of the research projects in this area that focus on creating a composer that can write its own music, but the writing still contains an important discussion of how computers can be used to create music.

2.1 AI Powered Composers

The following works discuss several different AI based composers that were created to generate their own music. For the purpose of this thesis, they will be analyzed for their effectiveness at composing as well as their shortcomings as tools that only have limited usability. All of these have proven to be effective at creating music, but none of them have shown popularity as music composition tools.

2.1.1 FlowComposer

Of the existing AI composers, FlowComposer is the only one that has seen use in industry [14]. An album that was recorded consisting of only music created with

FlowComposer was released in 2017 [14]. This is also one of the only tools that has an actual visual user interface and is not strictly accessed through the command line. It is currently being developed as part of Sony CSL and Flow Machines, but is not available to the public for use [14]. An audio example of what FlowComposer is capable of producing is available on their website, but access to the tool itself is not provided [2].

FlowComposer was designed to create new songs automatically in any style or can generate the style based on user provided parameters and input [14]. This tool is capable of re-harmonization (taking an existing melody and create new harmonies in different styles), variation (taking the melody and introducing variations into the pitch content and rhythm), and rendering (playing back a given score as if it were being performed) [2].

The musical output of this tool is a lead sheet that is scored for a full band. The music that FlowComposer produces fits into the category of popular songs (e.g. Pop, Jazz, Rock, R & B, etc.). It comes out fully notated and able to be performed or recorded. It is a very high quality tool and very capable at writing songs, but due to its research based nature, is kept from the public. It is highly likely also that this tool would be very expensive if it were available to the public. Maybe in the future, this will be a tool that is made available, but it will doubtfully ever be accessible to the average person due to the potential price.

2.1.2 BachProp

BachProp is a neural composer algorithm that was created to be able to compose new music in any style [5]. It was trained initially on the chorales of J. S. Bach, but is able to compose in any style when given appropriate training data [5]. The data is fed to the composer through MIDI files which are mathematically normalized before being translated into probability data that the algorithm uses to predict melodic direction and shape [5]. For the evaluation of BachProp, audiences were asked to rate several string quartets that it composed after it was trained on string quartets by Haydn and Mozart [5]. Based on the results of the surveys, the music was not only well received, but it was also quite convincingly in the appropriate style and character [5].

The process of actually performing crowd based musical validation is quite important here. Rather than the researchers who developed the composer rating the music, they had general audiences perform the reviews to better understand the broad appeal of the music. In this project, a similar crowd based reviewing of the interface was performed. Participants individually judged their created music in their own terms so that their personal success was measured to be able to determine if the user interface was able to guide them through the composition process.

BachProp has also been used and evaluated in the composition of music in a number of other styles [4]. The method of musical representation used by BachProp has proven to be more effective at capturing and later translating the information from the provided training scores than other algorithmic composers [4]. By normalizing how the musical data is stored, this tool is able to gather and retain more detail from the score than other composers [4]. This is what makes BachProp more effective at emulating a given style and producing more convincing musical output [4].

In order for the several Python libraries that were used to create the composer to cooperate, the musical data for this project was also represented in a normalized manner. Like BachProp, the data was represented as MIDI events to provide a standard for communication and output. This also means that at any step in the composition process the data could be imported into any standard music notation program and used externally to the composer as MIDI is the standard for digital music notation representation and playback.

The largest shortcoming of BachProp is that it was not designed in a way that can be used without extensive music or computer science knowledge. This means that is highly unlikely that it would ever be picked up by someone who simply wanted to play around with it or experiment with it. There is an extensive setup involved and it cannot be used by any simple means. However, BachProp is able to produce high quality music in any style and not just popular music as FlowComposer does.

2.1.3 DeepBach

DeepBach is a graphical model for musical composition that is designed to be effective at producing polyphonic music in four part hymn format [8]. This is yet another composer that was trained on the music of J. S. Bach in order to teach it the mechanics of composition and in this case, to teach it chorales specifically [8]. This composer is able to produce convincing Bach chorale style pieces that are driven by user parameters and input [8]. The following are examples of output composed by DeepBach.

Figure 2.1: Scores from Hadjeres's published paper showing DeepBach output [8]

(a) (b) (c)

Figure 8. Examples produced using DeepBach as an interactive composition tool. Examples (a) and (b) share the same metadata.

These scores by DeepBach demonstrate very convincing imitations of Bach chorales. Upon analysis, the composer has effectively followed the rules of this type of counterpoint as well as captured the style of Bach. It is important to note that the first two examples share the same user input data and parameters. This shows that there are at least several different possibilities for harmonization and development of the melodic line.

DeepBach does provide integration with the notation program MuseScore [8]. This means that there is a way in which people who are familiar with notation software could use this composer. While this is better than many of the others, it still requires a fair bit of prior knowledge. Additionally, this composer is not capable of operating in more than four part chorale style. The potential uses of this composer are very limited due to this fact.

Chapter 3

Method of Approach

This section has been broken down into three parts that each detail a different segment of the project that was completed. The first section describes the process and tools needed to develop the composer, the second section describes the process and ideas behind the development of the user interface, and the third section describes how the user interface study was designed and conducted.

3.1 Computer Composer Development

While having the functioning composer is an integral part of this project, the most important work here was done on the development of the user interface as this is where other composers in the past have been lacking. This being said, rather than building the composer from scratch, several Python tools and libraries were employed to created the backend of the tool.

None of these tools on their own were capable of forming the functioning composer. Several of them have very advanced sets of functions, but these were not all of what was required to build the composer. To complete the tool for this project, the output of each of these libraries was normalized and then transferred from one to another to reach the final output. The following table lists all of the tools that were used and their specific functions within the composer.

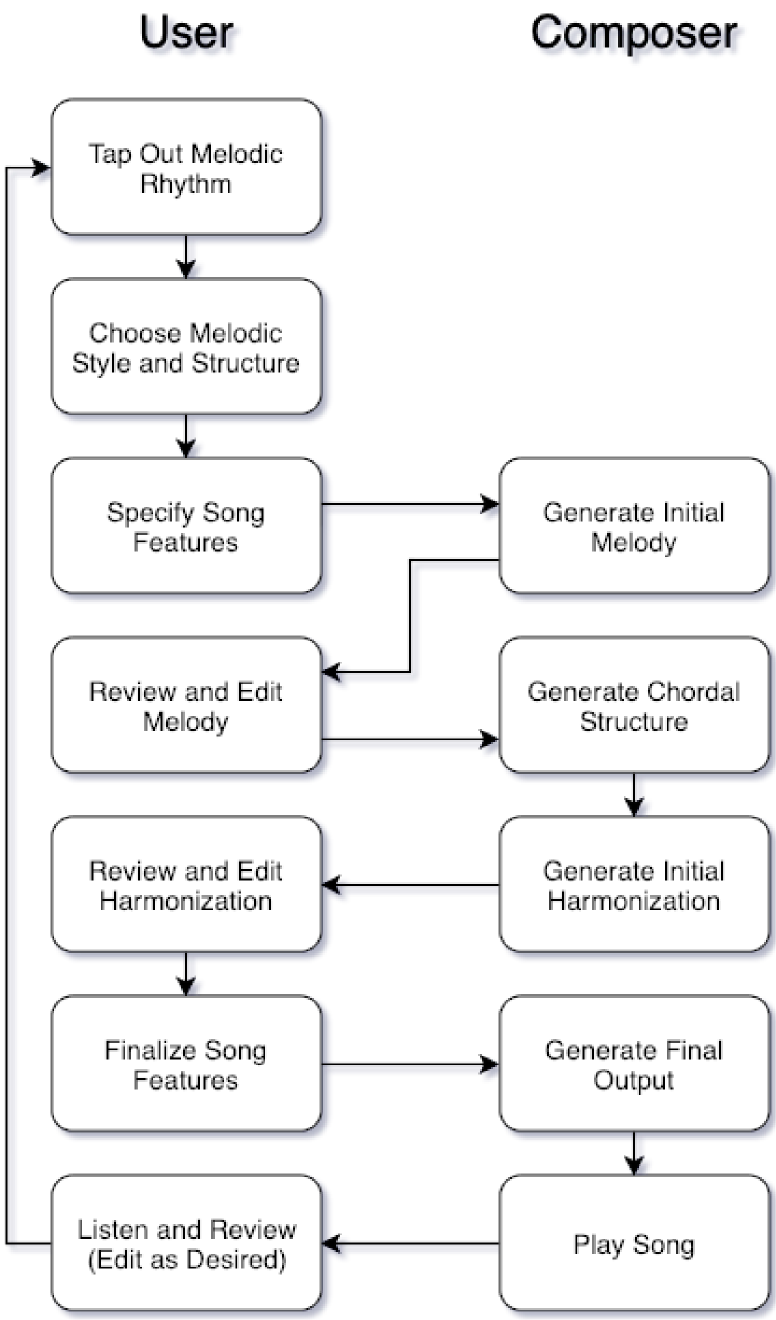
Table 3.1: Python tools and libraries and their uses in this project

music21	Melody Generation, Harmonization
MIDI Miner	Tonal Analysis, Key Analysis, Mood Analysis
MIDI Musical Accompaniment	Accompaniment Generation
Kodou	Notation
mingus	Interface Link
LilyPond	Notation Output
Improviser	Part Generation

3.1.1 Main Composer Functions

There are several important functions that any of these computer composers should have. This includes: melody generation, harmonization, key and metric analysis, orchestration, and notation. The following chart demonstrates the process taken to compose using this tool.

Figure 3.1: Composition process flowchart



Melody Generation

During this initial part of the composition process, the computer assists the user in creating a "good" melody. This involves determining the contour, rhythm, conjunction and disjunctions, and pitches of the melody. After the melody is generated, the computer determines if it is able to harmonize the melody. If it is not, it goes back and guide the user through the problematic areas.

The user also has the opportunity to change the melody if they do not like the way that it sounds or are not satisfied with what they have created. The computer also provides tips on how to make better melodies and makes suggestions based on the melody that has been generated.

Harmonization

Next, the computer takes the melody that it has determined that it can harmonize and performs an initial harmonization. By default, this follows the standard progression of chords in western music as displayed below.

Figure 3.2: Standard major key harmonic progression

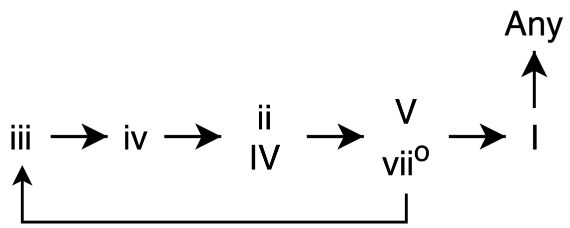
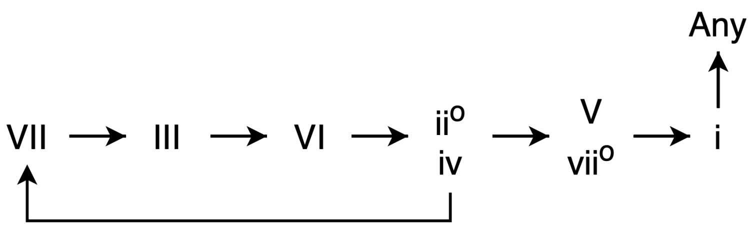


Figure 3.3: Standard minor key harmonic progression



The user is presented with a listing of the chords created during the harmonization and has a change to hear the progression and choose an alternative chord that still fits the melody. The user also has the option to create their own progression without the rules of western progression in place.

Key and Metric Analysis

The next step in the composition process metric and key analysis of the melody and harmony that were previously generated. This step is important for adding additional parts during orchestration and the final notation step. For the purpose of sound to the user, this does not matter, but it is require to be able to notate the composition.

This step also contains mood analysis for the purpose of providing the user with an idea of how what they are composing might feel. This provides them with a general affect of the piece and gives them some information as to what sound set of instruments to pick from.

Orchestration

Notation

3.1.2 Tools

The following is a descriptive list of the tools that were used in the creation of this composer.

- MIDI Miner
  - This is a Python library that is capable of analyzing music's tonal tension and classifying parts into different musical segments. This is used in the composition process to determine the mood and structure of the musical output.
- MIDI Musical Accompaniment
  - This is a tool, written in Python, that creates musical accompaniment after being provided with instructions in a text file. This tool is used to generate additional parts automatically based on the lines that the user creates. The instruction file is be populated by user choices such as style and tempo and by the composer's analysis with details such as key and structure.
- Kodou
  - Kodou is a Python package that does algorithmic music notation. This was helpful to be able to take the music created by the tool and output it to a file in standard music notation so that it can be taken and read by another musician.
- mingus
  - This is a tool that is capable of a number of useful functions for this composer. Generally, it is a music theory and notation package for Python, but more specifically, it can help build editors, educational music theory tools, and other music processing applications.
- LilyPond
  - LilyPond Is a program that can interface with a Python program in order to output music notation files. Kodou and mingus both use LilyPond for their notation output functions.

- music21
  - This is a set of tools that was developed by MIT for music theory, music computation, and generative composition. music21 contains the central set of functions that power the composer.
- Improviser
  - This is Python based software that was created to perform real-time music generation. Improviser is used in conjunction with music21 to perform the bulk of the back-end work when generating the compositions.

Bibliography

[1] Computer music. https://www.britannica.com/art/electronic-music/Computer-music, February 2018. Accessed: 2019-12-01.

[2] Flow Machines – AI music–making. https://www.flow-machines.com, 2018. Accessed: 2019-11-10.

[3] Aucouturier, J.-J., Defreville, B., and Pachet, F. The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. *Journal of the Acoustical Society of America 122*, 2 (2007), 881–891.

[4] Colombo, F., Brea, J., and Gerstner, W. Learning to Generate Music with BachProp. *CoRR abs/1812.06669* (2018).

[5] Colombo, F., and Gerstner, W. BachProp: Learning to Compose Music in Multiple Styles. *CoRR abs/1802.05162* (2018).

[6] Fernandez, J., and Vico, F. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research 48* (Nov 2013), 513–582.

[7] Guo, R., Herremans, D., and Magnusson, T. Midi Miner – A Python library for tonal tension and track classification.

[8] Hadjeres, G., and Pachet, F. DeepBach: a Steerable Model for Bach chorales generation. *CoRR abs/1612.01010* (2016).

[9] Herremans, D., Chuan, C., and Chew, E. A Functional Taxonomy of Music Generation Systems. *CoRR abs/1812.04186* (2018).

[10] Hiller, L., and Isaacson, L. *Machine Models of Music.* MIT Press, Cambridge, MA, USA, 1992.

[11] Li, S., Jang, S., and Sung, Y. Automatic Melody Composition Using Enhanced GAN. *Mathematics 7*, 10 (Sep 2019), 883.

[12] Liu, D., Lu, L., and Zhang, H.-J. Automatic Mood Detection from Acoustic Music Data.

[13] Pachet, F. The Continuator: Musical Interaction With Style. *Journal of New Music Research 32*, 3 (2003), 333–341.

[14] Papadopoulos, A., Roy, P., and Pachet, F. Assisted Lead Sheet Composition Using FlowComposer. *Principles and Practice of Constraint Programming* (2016), 769–785.

[15] Teymuri, A. PythonInMusic – Python Wiki. `https://wiki.python.org/moin/PythonInMusic`, 2019. Accessed: 2019-10-01.