Wesleyan University The Honors College

A Clustering Algorithm for Recombinant Jazz Improvisations

by

Jonathan Gillick
Class of 2009

A thesis submitted to the
faculty of Wesleyan University
in partial fulfillment of the requirements for the
Degree of Bachelor of Arts
with Departmental Honors in Computer Science

Middletown, Connecticut April, 2009

A Clustering Algorithm for Recombinant Jazz Improvisations

by

Jonathan Gillick
Class of 2009

Abstract

Music, one of the most structurally analyzed forms of human creativity, provides an opportune platform for computer simulation of human artistic choice. This thesis addresses the question of how well a computer model can capture and imitate the improvisational style of a jazz soloist. How closely can improvisational style be approximated by a set of rules? Can a computer program write music that, even to the trained ear, is indistinguishable from a piece improvised by a well-known player?

We discuss computer models for jazz improvisation and introduce a new system, *Recombinant Improvisations from Jazz Riffs* (Riff Jr.), based on Hidden Markov Models, the global structure of jazz solos, and a clustering algorithm. Our method represents improvements largely because of attention paid to the full structure of improvisations.

To verify the effectiveness of our program, we tested whether listeners could tell the difference between human solos and computer improvisations. In a survey asking subjects to identify which of four solos were by Charlie Parker and which were by Riff Jr., only 45 percent of answers among 120 people were correct, and less than 5 percent identified all four correctly.

Table of Contents

Acknowledgements

I would like to thank several people whose hard work, insight, and encouragement were essential to this thesis. First, Bob Keller, for getting me started on the project in the summer of 2008, for letting me work within his Impro-Visor software, and for his endless supply of good suggestions emailed between 2 and 3 in the morning. Second, my advisor Danny Krizanc, for steering me in the right direction and providing valuable feedback. Thanks to Kevin Tang for working with me all summer, and thanks to Ron Kuivila for sparking several months worth of thought in two conversations. Thanks also to my father for his encouragement, patience, and ability to answer all sorts of math questions, to the rest of my family for their support, and to my friends for providing much needed distractions.

Chapter 1 – Introduction

What do Bach, Chopin, Coltrane, and Parker have in common? They don't compose any more. Their works are endlessly analyzed and copied; some musicians spend much of their careers learning to imitate the masters, and not simply to play pieces - they learn to compose like Bach or to improvise like Parker. These musicians then create new music that, though different from any particular piece, sounds like it could have been composed by the artist they are emulating.

Somehow, through careful analysis of a set of pieces, musicians are able to extract and reproduce the elements that make up a composer's style. Lewis Rowell, a music professor at Indiana University who composed pieces in the style of J.S. Bach, remarks, "Bach developed a kind of grammar that I merely picked up, as could anyone who wished to. And then, armed with this grammar, I – just like anyone with a bit of musical talent – can easily compose any number of pieces in perfect Bach style." [Cope 2001, p. 57] Whether or not the process of acquiring a grammar and using it to create new pieces in true Bach style is as straightforward as Rowell claims, his comment suggests that the process of imitating Bach is reducible to a formula. Presumably, such imitation takes considerable time and effort, though. What if we could create a computer program to capture this process? Then we could simply feed Bach to the computer and receive a large output of new Bach-like compositions.

This observation raises several questions. First, could such a formula be applied to any type of music? Bach's music is highly structured and many of the rules he follows are well understood, but what about jazz music? Jazz, a more open, more spontaneous form of music, consists largely of improvisations. Still, the level at which some jazz players can mimic Charlie Parker is comparable to that at which classical composers imitate Bach. A second question, then, is how do people learn to emulate artists? Some surface elements might be easy to copy, but to truly capture the spirit of an artist's style, a deep understanding of the artist's approach and thinking seems necessary. Anyone who is familiar with an expert musician knows that he or she has a personal style. One might not easily be able to describe the way a musician plays, but many jazz fans can recognize their favorite players after listening for just a few seconds. In fact, there are those who claim to be able to identify saxophonist Stan "the Sound" Getz after hearing a single note. An excellent imitation of an artist should be similarly recognizable. Another set of questions arises, then: What is a good imitation of an artist? How good can an imitation be?

Answers to all of these questions appear likely to vary from musician to musician and are difficult to study empirically without a deep psychological investigation into the workings of the musical brain. Computer models provide an alternate platform to study how composition, improvisation, and style emulation work, particularly by enabling comparisons between the output of models with different structures or parameters.

1. 1: Music and Artificial Intelligence

To create a computer program to accomplish a human task, we must be able to describe the task in terms understandable by a machine. One school of thought, sometimes known as *mind beyond machine*, posits that some aspects of human existence cannot be understood or reproduced by machine [Kurzweil 1990]. One field that is sometimes presented as such, and thus plays a significant role in the Artificial

Intelligence (AI) debate, is the arts, and particularly artistic creativity. In *Gödel, Escher, Bach*, Douglas Hofstadter articulates a defense for music as a standard for human thought and creativity:

"Question: Will a computer program ever write beautiful music?

Speculation: Yes, but not soon. Music is a language of emotions, and until programs have emotions as complex as ours, there is no way a program will write anything beautiful. There can be "forgeries" – shallow imitations of the syntax of earlier music – but despite what one might think at first, there is much more to musical expression than can be captured in syntactical rules… To think – and I have heard this suggested – that we might soon be able to command a preprogrammed mass-produced mail-order twenty-dollar desk-model "music box" to bring forth from its sterile[sic!] circuitry pieces which Chopin or Bach might have written had they lived longer is a `grotesque and shameful misestimation of the depth of the human spirit. A "program" which could produce music as they did would have to wander around the world on its own, fighting its way through the maze of life and feeling every moment of it." [Hofstadter 1979]

Others argue that computers *could* produce music sounding every bit as inspired and human as Bach or Chopin, but that such music would not grant the computer intelligence. John Searle, in his "Chinese Room" thought experiment, suggests that a computer program, given the correct instructions, could behave in a manner indistinguishable from a human without having any true understanding of human nature [Searle 1985].

1.2: Thinking and Improvising

We chose jazz improvisation as our vehicle for study partly out of opportunity (I received a National Science Foundation grant in the summer of 2008 to work with Bob Keller, a professor doing research in automated jazz improvisation at Harvey Mudd College), but partly because improvisation is a quick, spontaneous, and natural process. Whereas classical composers might spend days or weeks assembling a piece in what might seem to be a rigorous and structured procedure, the thought process of an improviser may be more indicative of the nature of human musical creativity. Modeling a jazz player requires simulating the instinctive responses and connections that make music inventive and organic. While improvising, players must make decisions at many points, and their decisions can be thought of as rules applied at different levels. If I am improvising, and I have just played a fast passage of sixteenth notes, I might decide to pause and then start a new slower paced phrase. If I have just played an arpeggio[1] of several consonant sounding notes in a row, I might decide to play a note outside the chord to add tension to the melody. On a higher level, if I have just introduced an idea, I might repeat a variation of that idea before going on to something else. We attempt to capture such thought processes in our model.

1.3: Practical Uses for Automated Jazz Improvisation

Other than to gain insight into the way music and the musical brain work, what is the point of developing a computer program to improvise jazz? Why create solos that sound like Charlie Parker but rarely, if ever, end up as good? Imagine a jazz musician preparing for a show. He is supposed to take a solo during a song, but nothing he tries in practice sounds the way he wants. In desperation, he thinks, "What would Charlie Parker play here?" A computer program that can generate solos in the style of Charlie Parker over the particular set of chords in his song (most jazz songs are based on a sequence of chords, and soloists need to build their solos on top of the chords) might be very useful to him to come with his own ideas. Similar to the way master chess players often train with software ("What would Deep Blue do here?"), one could imagine musicians referring to programs like Riff Jr. for guidance and to explore possibilities.

[1] An arpeggio consists of several notes from a chord played one at a time.

Automated jazz improvisation can also be useful for education. Much of learning to solo involves learning what to play over certain kinds of chord progressions. Tools to generate solos in a given style have proven to be valuable, as evidenced by successful software such as PG Music's Band-In-A-Box[2] and Bob Keller's Impro-Visor[3] (Improvisation Advisor), open source jazz education software developed at Harvey Mudd College. The software developed for this thesis will be included in an upcoming release of Impro-Visor.

Finally, one could imagine performances by a virtual musician or band containing virtual musicians. Many artists compose songs for more instruments than they have musicians in a live band. Some use prerecorded parts or record and loop parts on stage, but virtual musicians could provide more flexible alternatives. Toby Gifford, a PhD student at Queensland University of Technology in Australia, has drawn interest for his software tool "Jambot" that listens to music and plays along, providing accompaniment, rhythm, or lead parts [Wilson 2008].

1.4: History of the Problem

Automated music composition has been around since long before the invention of computers. An early form of algorithmic composition, the *Musikalisches Würfelspiel*, or "musical dice game," dates back to the 18th century and was popularized by Mozart. For this game, Mozart composed eleven measures, any two of which sound coherent in succession. A roll of a pair of dice selects one measure, and sixteen consecutive rolls yield a sixteen-measure minuet, so this game can generate 11^{16} distinct stylistically acceptable minuets. The *Musikalisches Würfelspiel* was

[2] See pgmusic.com
[3] See http://www.cs.hmc.edu/~keller/jazz/improvisor

extremely popular in 18th century Europe and was commonly played in German parlors, but it eventually lost popularity and exhibited little influence for nearly two centuries [Cope 2001].

With the development of computers in the 1950s, interest in algorithmic composition reemerged. Lejaren Hiller and Leonard Isaacson wrote programs for ILLIAC, the first computer, which composed the *Illiac Suite for String Quartet* in 1956 [Cope 2001]. [Xenakis 1971], [Ames, et al. 1992], and [Cope 2001], among others, created programs to compose music in several styles, including jazz, popular, classical, and opera. Systems for jazz improvisation have been proposed, among others, by [Biles 1994], [Papadopolous and Wiggins 1998], and [Keller and Morrison 2007]. We will discuss some of these systems in chapter 3.

1.5: Goals and Organization of the Thesis

This thesis introduces a new program for automated jazz improvisation, *Recombinant Improvisations from Jazz Riffs* (Riff Jr.), with a few goals in mind. First, to most accurately simulate an artist's style, the only input we should use is data from the original artist. Some approaches require human training to optimize algorithm performance. User input, though, in addition to requiring time for a training phase, yields a model for a particular user's interpretation of an artist's style rather than deriving a function directly from an artist to a model. Second, we want to evaluate the effectiveness of our method in terms of both quality of output and style similarity. Finally, we want each generated solo to be unique and sufficiently different from any solo from the original artist, while still demonstrating the intended style.

In chapter 2, we describe the knowledge of jazz music necessary for our method, as well as the mathematics and computer science background we draw on, including Markov models, conditional probability, and context free grammars. Chapter 3 looks at and analyzes other approaches taken toward algorithmic composition and then outlines our new method based on a clustering algorithm. In chapter 4, we present the results of a listening test that we administered to 120 subjects, asking them to identify which among four solos were computer-generated and which were composed by Charlie Parker. Finally, in chapter 5 we discuss the success of our method and its implications for future progress in automated jazz improvisation.

Chapter 2 – Background

2.1 Musical Background

A deep understanding of music theory is not necessary for this thesis, but the basics of jazz structure will be used throughout.

2.1.1 What is jazz?

Jazz encompasses a wide range of categories in a constantly evolving landscape of music. Since people have varying notions of what defines jazz, when it originated, and what styles it grew out of, strict definitions tend to be controversial. Some characteristics, however, bridge the gap between eras and represent what the average person might picture when imagining jazz music. Two such distinguishing elements are improvisation and a swing feeling [Gridley 1985].

Swinging basically consists of playing a steady beat over syncopated rhythms, as opposed to classical music, where the tempo is less strict and the rhythms adhere more to the beat. The swing feeling in jazz is in large part defined by the swing eighth note pattern, in which rather than playing two notes of equal duration, the first note is held for twice as long as the second, in effect dividing beats into two thirds and one third instead of equally in half [Gridley 1985]. The swing feeling sets the stage for improvisation, a form of composition done concurrently with performance [Gillick et al. 2009]. While the rest of the band maintains a steady pulse, a soloist, or lead player, can play complex lines and freely diverge from the melody of the song. Although improvisation is largely spontaneous, players do practice ideas prior to performance - most players have a vocabulary of licks (short phrases) that they drawn on when constructing solos. Jazz bands traditionally start by playing the melody ("the head") of a song, and then lead players take solos in turns while the rhythm section (usually drums, bass, and piano or guitar) accompanies.

Since the late 1960s, modern jazz has merged with popular music and branched into a wide range of styles such as smooth jazz, nu jazz, and acid jazz, which stray in varying degrees from their roots. For this thesis, then, we restrict the music we use to bebop, cool jazz, and hard bop music from the 1940s and 1950s. Music played in these styles today is sometimes known as "straight ahead jazz."

Any jazz musician will tell you that the essence of jazz is in the subtleties with which it is performed. Timbre, inflection, and slight rhythmic nuances that make playing expressive and human are rarely notated and sometimes overlooked. Although in this thesis, we do not attempt to capture all of the expressiveness and spirit that make music "jazzy," it is important to consider the context and manner in which the melodies we examine are played.[4]

2.1.2 Structure

Jazz tunes are generally structured around a sequence of chords. A typical band might have a pianist playing the chords, a bassist outlining the chords with a walking bass line, a drummer keeping the beat, and a saxophonist playing the melody. When improvising, a lead player usually works off of the chords and the melody. In its simplest form, improvisation can consist only of slight variations to the melody, but at times, soloists disregard the melody and play just based on the chords. Players use scales or sets of notes that work with particular chords, and they build their melodic

[4] Deeper exploration of expressiveness in jazz performance has been done, in large part by precisely analyzing the rhythm and volume of individual notes. For further details, see [Ramirez, et al. 2008]

lines to fit with the chord changes. The chords can also be used to define a set of keys, or tonal centers, for the song. A given key is associated with a scale, and sometimes several successive chords in a song will fit into one key, allowing an improviser to play within one scale for an extended period of time. Some artists like to "play outside," or without connection to the harmony of a song, but we do not focus extensively on outside playing [Owens 1995].

2.1.3 Notation

The music on which all our data is based is written in a "lead sheet" format. Lead sheet notation displays the melody of a song in conjunction with a sequence of chords that outlines the harmony. Jazz groups often play songs from a lead sheet only, with each player determining his own part based on the chords. This is a convenient notation for abstracting songs into an easily understood form and allows for analysis of a simpler structure than complete written parts for each instrument.

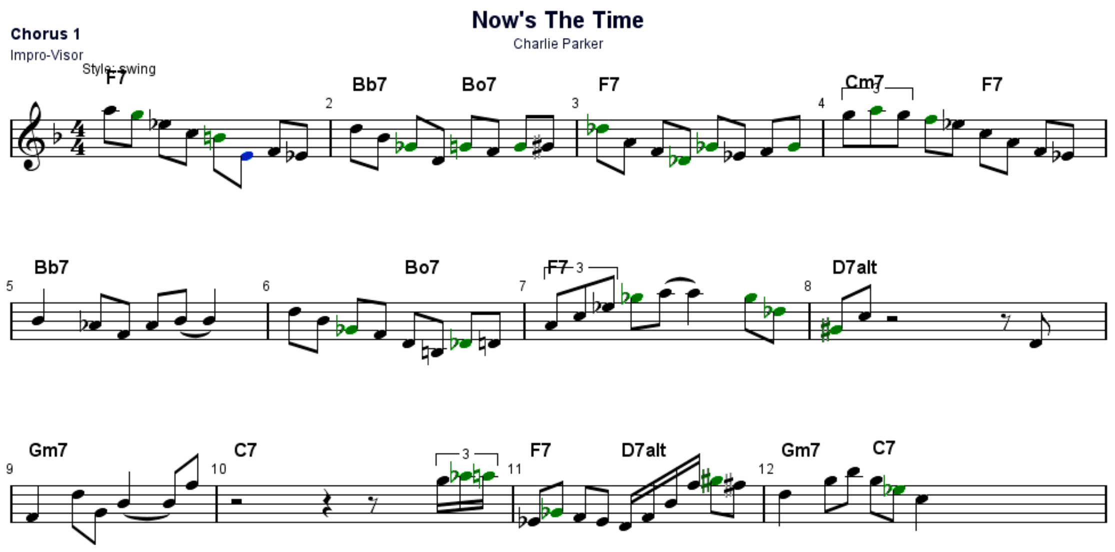


Figure 2.1: Impro-Visor displaying a lead sheet

2.1.4 Musical Representation

All of our musical data is in leadsheet files, developed by Bob Keller for Impro-Visor.[5] These files contain the data in a standard lead sheet: a sequence of chords and a sequence of notes. Notes are represented in the same form as in MIDI – as duples of pitch and duration, with the tone quality defined by the selected instrument for the song. Chords contain combinations of notes, and global tempo parameters set the absolute durations of notes and chords. In chapter 3 we will discuss higher-level musical structures that we apply to the data.

2.2 Relevant Mathematics

This section outlines the main mathematical concepts used in this thesis.

2.2.1 Training Data

To create a model of an artist's improvisational style, we need some amount of input, which is known as training data. For each artist, we have a number of transcriptions that we use to create a model (transcriptions are represented in lead sheet format with notes defined by pitch and duration). Depending on what we specifically want to model, we can use different sets of training data. If we want to examine how Charlie Parker's style changed from 1947 to 1953, we can construct two models, one trained on solos from 1947 and the other trained on solos from 1953. If we want to create a model for a hypothetical child genetically engineered from Bill Evans and Red Garland, we can use solos from the two of them as training data.

2.2.2 Markov Models

A Markov Model represents a finite system as a sequence of states and a set of

[5] For more on Impro-Visor, see http://www.cs.hmc.edu/~keller/jazz/improvisor/

transition probabilities between states. Making the “Markov Assumption” specifies that given knowledge of the current state of the system, the probability distribution of future states is uninfluenced by previous states. To model a melody, for example, we can designate each pitch as a particular state and let the probabilities of transitions between pitches be determined by which notes follow other notes in the melody. We can build a generative model for the melody by picking a “start state” from the notes in the song, and then recursively using the current note and the sequence of transition probabilities to select future notes until we have a “Markov Chain” of desired length.

2.2.3 Conditional Probability

A Markov Model answers the question: “Given the current state X, what is the probability that the next state is Y?” Or equivalently: “What is the conditional probability of Y, given X?” By the definition of conditional probability,

$P(Y \mid X) = P(Y \cap X) / P(X)$.

2.2.4 N-Grams

An N-gram represents a melody as sequences of n notes, or, more generally, a system as sequences of n states. As an example, consider the first six notes in the tune of “Happy Birthday:” C C D C F E

Unigram	Bigram	Trigram
C	C, C	C, C, D
C	C, D	C, D, C
D	D, C	D, C, F
C	C, F	C, F, E
F	F, E	
E		

Table 2.1: 1,2, and 3-gram models for “Happy Birthday”

Trigrams implicitly regard the current state as being represented by the last two notes played; unigrams and bigrams regard the current state as just the current note, but unigrams consider each note independent from previous notes. Depending on the size and nature of the data set, the value of the information captured by different order N-grams varies. A 100-gram representation is too specific to model a jazz solo, since any sequence of 100 notes would most likely appear only once in a set of songs, so using the 100-gram for generation would simply recreate a given melody. On the other hand, a bigram model might capture less information than a larger N-gram.

Note	C	D	E	F
C	1/3	1/3	0	1/3
D	1	0	0	0
E	0	0	0	0
F	0	0	1	0

Table 2.2: Bigram State transition matrix for “Happy Birthday”

Table 2.2 shows a bigram model for “Happy Birthday” with matrix entries indicating the transition probabilities from notes in the first column. Note that since nothing follows E in the training data, all transition probabilities from E are 0. To deal with such cases, we need to do some *smoothing* – approximating our model with a simpler model. The easiest form of smoothing here is to use the unigram model when the bigram model gives no estimation. The transition probability from E, then, would follow the probability distribution of notes in the song. Generating a 10-note song from this model might yield a melody such as C, D, C, C, F, E, C, F, E, D.

2.2.5 Hidden Markov Models

Hidden Markov Models (HMM’s) assume that a set of observed data follows a Markov process with unknown states. The states, then, are not surface level elements and must be inferred from the data. Rather than taking sequences of notes in a melody as states themselves, we can choose a set of states based on characteristics of the notes, build a transition matrix accordingly, and in generation produce a new sequence of notes based on the inferred states. For example, many jazz songs have A and B sections. A bigram HMM could contain two states A and B, where the current stated is inferred from some properties of the melody.

State	A	B
A	1/3	2/3
B	1	0

Table 2.3: Bigram HMM State Transition Matrix for Song Form

This model could generate forms for songs such as the common A,A,B,A.

2.2.6 Grammars

Grammars, first outlined by Noam Chomsky in 1956, describe a generative model for formal languages. Originally intended for the study of natural language, grammars have been applied in many areas, including music. Grammars consists of four components: E, N, S , and P, where E is a finite set of terminals – the alphabet of the language, N is a finite set of nonterminals, S is the initial nonterminal, and P is a finite set of production rules of the form $A \in N \rightarrow a \in (N \cup E)$. The process of applying production rules starting with S is known as expanding. The language of a grammar is equal to the set of terminal strings the grammar can generate [Chomsky 1956]. A context free grammar (CFG) is restricted to production rules whose left side contains a single nonterminal.

Grammars that associate probabilities with each production rule are known as *stochastic grammars*. Consider the language of the following stochastic context free grammar (SCFG) for “Happy Birthday.”

E = {c,d,e,f} N = {S, C, D, E, F}

Rule	Probability
S → C	1/2
S → D	1/6
S → E	1/6
S → F	1/6
C → cC | c	1/3
C → cD | c	1/3
C → cF | c	1/3
D → dC | d	1
F → cC | d	1

Table 2.4: Stochastic Context Free Grammar for "Happy Birthday"

A production of the form (C → cD | c) denotes that C can expand to either cD or c and is equivalent to two rules (C → cD) and (C → c). Generally, probabilities for both rules would be required, but we assume that the length L of a word is determined in advance, so once the string contains L-1 terminals, we choose (C → c) instead of (C → cD). The probability of a word in this language is equal to the product of the probabilities of the productions used in the expansion of the word, or in other words, the conditional probability of the last production taken. Among 3 note melodies, for example, P(cdc) = P(S → C)P(C → cD | c)P(D → dC | d) = (1/2)(1/3)(1) = 1/6. Any Markov model can be represented by a stochastic context free grammar (the SCFG in table 2.4 is a grammar for the Markov model in table 2.2). Note that the grammar requires smoothing for the same reason as the Markov model.

Chapter 3 – Algorithms

Many different avenues have been taken toward algorithmic composition, and this thesis does not attempt a comprehensive survey, but we will discuss the basic of some common approaches. Cope[2005] and Papadopolous and Wiggins[1999] examine a list of approaches, including rule-based algorithms, data-driven programming, evolutionary methods, neural networks, fuzzy logic, mathematical modeling, sonification, grammars, and hybrid systems. Sources disagree on the divisions between these methods, so we will discuss the basics of two more general categories of approaches – systems that learn and rule-based algorithms – into which many of the above algorithms can be argued to belong. We will also discuss several examples of implementations.

3.1 Systems that Learn

Systems that learn are dynamic models for jazz improvisation that change based on input. These systems can either build upon an initial set of rules or start with no a priori knowledge whatsoever [Papadopoulos and Wiggins 1999].

3.1.1 Genetic Algorithms

Genetic algorithms, inspired by the principles of natural selection, are generally used as heuristics in optimization or search problems. Given a large space of possibilities, genetic algorithms start with a set of initial solutions, and through various evolutionary operations such as mutation, selection, and inheritance, develop new generations of solutions iteratively until some convergence criteria are met. Genetic algorithms consist of several basic steps:

1. Initialize a population of individual solutions.

2. Determine the fitness of each individual in the population.
3. Choose a set of the fittest individuals to be parents.
4. Breed new individuals.
5. Initialize next generation with new individuals.
6. Repeat steps 2-5 until the population reaches an acceptable fitness level.

The method to determine fitness of individuals, called a fitness function, usually requires human input. In some cases, the optimization is done on the fitness function itself, so after sufficient evolution of the fitness function, the program can evolve its population without further input.

[Biles 1994] developed GenJam, a genetic algorithm that learns to improvise jazz. He points out that, as in many problem-solving scenarios, composers and improvisers devote much of their efforts to search – looking for a melody or chord that sounds right. Biles describes his creation:

"GenJams metaphor is an enthusiastic student musician who sits in at jam sessions. When this student plays well, the other musicians respond with 'Yeah!' and other classically cool jazz exhortations. When the student plays poorly, the other musicians might respond by 'gonging him off,' as Jo Jones did to a young Charlie Parker by sailing a cymbal at his feet during a Kansas City jam session. GenJam uses similar, though less dramatic, feedback to guide its search through a melodic space." [Biles 1994]

The population in GenJam consists of measure-length melodies, along with phrases, which are made up of four measures. Biles represents melodies with "chromosomes" of bits, and modifying the chromosomes yields variations of the melodies. During the program's training phase, a human listens to a piece and rates as she listens. Each rating applies to both a phrase and one of the measures in the phrase, and the set of ratings then determine which melodies will be selected as parents.

3.1.2 Neural Networks

Artificial neural networks (ANN's) are systems inspired by neuron connections in the brain. Components in a neural network have weighted connections to one another that can be strengthened or weakened based on training and experience. Given a set of possible solutions to a problem, neural networks can come up with similar solutions by triggering random initial output and then repeatedly adjusting connections until the output sufficiently resembles the input. Neural networks can learn unsupervised, using only the training data, or with supervision, where a user gives grades to output, which in turn affect the connections of the network. [Moser 1994] trained an ANN called CONCERT to generate melodies in the style of Bach. CONCERT looks to learn information about note-to-note transitions as well as higher-level structure of songs.

3.2 Rule-Based Algorithms

The idea of a rule-based algorithm is to model the structure of an artist's style with a set of rules based on music theory, the programmer's own preferences, or a corpus of data. These rules apply to particular conditions and can sometimes be represented by a set of conditional probabilities with a Markov Chain. Table 3.1 demonstrates a set of rules allowing intervals of up to four half steps, with the row number denoting the current interval and the column number the next interval.

Interval(Half Steps	-4	-3	-2	-1	0	1	2	3	4
-4	0.05	0.1	0.05	0.3	0	0.05	0.3	0.1	0.05
-3	0.15	0.05	0.3	0	0.05	0.25	0.05	0.1	0.05
-2	0.05	0.2	0.15	0.1	0.05	0.05	0.25	0.1	0.05
-1	0.15	0	0.3	0.05	0.05	0.15	0.05	0.15	0.1
0	0.05	0.05	0.2	0.2	0	0.2	0.2	0.05	0.05
1	0.1	0.15	0.05	0.15	0.05	0.05	0.3	0	0.15
2	0.05	0.1	0.25	0.05	0.05	0.1	0.15	0.2	0.05
3	0.05	0.1	0.05	0.25	0.05	0	0.3	0.05	0.15
4	0.05	0.1	0.3	0.05	0	0.3	0.05	0.1	0.05

Table 3.1: State transition matrix for bigram Markov Chain using intervals between 0 and 4 half steps

Rules such as those in the transition matrix in table 3.1 would produce melodies that tend to sound musical but lack direction. This matrix deals only with harmony, so it might be used in conjunction with a parallel matrix to specify rhythm or expanded to combine rhythm and interval into a single state, yielding matrix categories of the form: (duration, interval). Such simple sets of rules have been shown to generate interesting, if not musical, pieces [Cope 2005].

3.2.1 Experiments in Musical Intelligence

More complex models, such as David Cope's Experiments in Musical Intelligence (EMI), have produced accurate imitations of composers. Programs like EMI work by taking as input a corpus of works from a composer, analyzing the music, and obtaining from it a set of rules. We take this type of approach in our work, although the sorts of rules we extract from jazz solos differ from what Cope gathers from classical pieces.[6]

The key component of Cope's work is *recombination*. In one version, he uses a corpus of 370 Bach chorales, pieces usually consisting of four voices, as a basis for new Bach-influenced chorales. Cope divides the training data into beat-length or measure-length sections and then recombines them in a Markovian process by looking for extracted parts that, if placed sequentially, follow the rules of Bach's voice leading.[7] Every extracted section is marked by its first set of notes that sound

[6] Cope does apply EMI to a variety of genres, including ragtime and opera, but his work usually composes the entire harmony of a piece, whereas we create monophonic solo lines to be played over chords.

[7] In music for multiple parts, the voices combine to form chords. When the chord changes, the way that each part moves to reach the next chord is called voice leading. For more on voice leading see chapter 1 of [Cope 2005].

together as well as the first set of notes that follow the part. Extracted sections are grouped into what Cope calls lexicons, each of whose members has exactly the same set of starting pitches. So, in recombination, a current state's next set of notes points to a particular set of lexicons from which EMI can choose a next state. In this process, which can be described by a bigram Markov chain, EMI only chooses harmonic transitions that have appeared at least once in the training data.

Clearly, a sufficiently large set of training data is necessary for successful composition of new pieces. If given more than one choice for the next state, EMI does not choose the actual transition that occurred in the corpus. With too small a data set, often the only possibilities that follow the correct voice leading rules would be the particular sections that appeared in order in one of the chorales, so the program would simply reproduce a piece from the training data.

Cope points out that although these rules ensure logical connectivity between beats or measures, the approach yields pieces lacking in larger-scale structure. He suggests higher-order Markov chains as a vehicle for additional structure but argues that larger N-grams are insufficient, since "In music, what happens in measure 5 may directly influence what happens in measure 55, without necessarily affecting any of the intervening measures [Cope 2005]." As a result, Cope adds rules for global structure and form.

EMI uses a multi-level top down structure. Based on an analysis of the musical tension of a section of a song, Cope classifies the section as one of five identifiers: Antecedent, Preparation, Extension, Statement, or Consequent, in ascending order of stability.[8] Each of these types can break down into two or more types in what Cope calls a SPEAC hierarchy. These hierarchies expand like a grammar from one identifier, ending with a sequence of identifiers that can be translated into music.

3.2.2 Grammars

[Keller and Morrison 2007] developed an SCFG for automated jazz improvisation in which nonterminals can have a counter associated with them to indicate when to stop expanding. The counter denotes a number of beats, so to generate a solo over 32 beats, the start symbol receives parameter 32. Given a start symbol P, expansions are of the form (P 32) → (Seg2)(P 30), where (Seg2) eventually expands to a sequence of terminals that will produce two beats of music. Each production rule has a probability, allowing different terminal strings to be produced each time.

The key idea in this system is manifested in the terminals. Keller and Morrison choose as their terminals duples containing duration and one of several categories of notes that are relevant to jazz playing. They define the categories as follows:

1. Chord tones: tones of the current chord.
2. Color tones: complementary tones for the current chord. These are tones that are not in the chord, but which are individually sonorous with it. They are also often called “tensions.”
3. Approach tones: non-chord tones that make good transitions to chord-tones or color tones.
4. Other: tones that do not fall into one of the categories above.

[8] For a more detailed description of the way EMI calculates tension, see chapter 7 of [Cope 2005].

The note categories are represented in the grammar with specific symbols, namely:

1. C a chord tone.
2. L a color tone.
3. A an approach tone.
4. X an arbitrary tone.
5. R a rest.

To these categories, Keller and Morrison add:

6. H a "helpful" tone, defined to be any one of the above.
7. S a scale tone, a member of a scale designated as being compatible with the chord.

The grammar might generate the sequence (A8 C8 L4), representing an 8th note approach tone, followed by an 8th note chord tone, and finally a quarter note color tone. Keller and Morrison apply the terminal strings generated by the grammar to lead sheets, and based on the chords and a few additional constraints such as a bound for maximum interval between notes, they determine the final melody.

3.3 Analysis of Previous Approaches

Cope's approach with EMI was quite successful – good enough to fool listeners, including rooms full of music theorists, into confusing an EMI composition with Bach or Chopin. Cope tests EMI on human audiences with what he calls "The Game," in which he plays four pieces with the instruction that at least one is a human composition and at least one is an EMI product, asking listeners to identify which is which. Large groups of listeners generally average between 40 and 60 percent correct responses, including a test of 5000 subjects in 1992 [Cope 2001]. Douglas Hofstadter, author of the impassioned defense of music as a human phenomenon mentioned in chapter 1, describes his reaction upon hearing EMI for the first time:

"I noticed… an Emmy[9] mazurka supposedly in the Chopin style, and this really drew my attention because, having revered Chopin my whole life long, I felt certain that no one could pull the wool over my eyes in this department. Moreover, I knew all fifty or sixty of the Chopin mazurkas very well, having played them dozens of times on the piano and heard them even more often on recordings. So I went straight to my own piano and sight-read through the Emmy mazurka – once, twice, three times, and more – each time with mounting confusion and surprise. Though I felt there were a few little glitches here and there, I was impressed, for the piece seemed to *express* something. If I had been told it had been written by a human, I would have had no doubts about its expressiveness. I don't know that I would have accepted the claim that it was a newly uncovered mazurka by Chopin himself, but I would easily have believed it was by a graduate student in music who loved Chopin. It was slightly nostalgic, had a bit of Polish feeling in it, and it did not seem in any way plagiarized. It was *new*, it was unmistakably *Chopin-like* in spirit, and it was not *emotionally empty*. I was truly shaken. How could emotional music be coming out of a program that had never heard a note, never lived a moment of life, never had any emotions whatsoever?" [Cope 2001]

Biles' GenJam has also proven to produce aesthetically pleasing music – he often plays gigs with it, headlining as "The AI Biles Virtual Quintet." Biles sums up GenJam's playing after sufficient training as "competent with some nice moments [Biles 1994]." Mozer states that CONCERT generates melodies that are "preferred over compositions generated by a third-order transition table, but still "suffer from a lack of global coherence" [Papadopoulos and Wiggins 1999]. Keller, who teaches jazz improvisation, remarks that his grammar generates solos that compare favorably with those of his intermediate level college students [Keller and Morrison 2007].

3.4 New Contributions

Our software is built on top of the work of [Keller and Morrison 2007] in the Impro-Visor software tool. Whereas Keller and Morrison used a handcrafted grammar to generate jazz melodies, we add functionality in the tool to generate solos in the style of a given artist based on recombination of a corpus of performances. The way in which the recombination is done depends largely on two factors: the way that the data is broken into fragments, and the way that these fragments are combined.

[9] Cope and Hofstadter often refer to EMI as Emmy.

3.4.1 Representation:

In our implementation, we break training data into pieces of equal size. After some trial and error, we found that given the limited size of our data sets for each artist (the largest set, Charlie Parker solos, being about 400 measures), measure length pieces (all training data is in 4/4 time) tend to give the best results. Short fragments yield more possible combinations but sound less coherent, while longer fragments often sound too much like the original piece. Recombination is by nature a data hungry method, and given larger data sets, we can expect better results. David Cope used 370 chorales for his Bach imitations, a data set much larger than our 9 Charlie Parker tunes. Cope breaks his training data for the most part into beat length pieces but also mixes in measure length sections.

By using measure-size fragments, we are essentially capturing a collection of licks. Jazz players are known to have "toolkits" of licks from which they draw in their improvisations, and we model these toolkits with measure length extractions. Though licks can be longer than a measure, measures are long enough to contain ideas or expressions in of themselves. To vary the possibilities in generation, we abstract the extracted licks so that each can be filled in several different ways over any particular chord progression. Of the possible instantiations of each lick, only ones that fit well with adjacent measures will be chosen.

To abstract a lick, we start with Keller and Morrison's note categories – chord tones, color tones, approach tones, and other tones – and add an additional constraint that we call *slope*. We associate each note in an abstract lick with a slope, which defines minimum and maximum allowed intervals from the previous note. When instantiating from an abstraction, we look for notes within the allowed intervals that fall into the correct category given the current chord in the progression. Since chord tones are the most consonant and the most significant in shaping the melody, we weight the importance of slopes more highly than all other types of notes. Consequently, when generating, if no note of the desired category falls with the slope bounds, we look outside the bounds if the desired note is a chord tone, but not for the other types of tones.

Slope, a construct intended to capture information about the contours of melodies, allows for representation of many common jazz idioms. As an example, figure 3.1 demonstrates *enclosure*, where a player leads up to a chord tone with notes above and below. Contours have been used in other representations, but never in conjunction with note categories as we use it.[10]

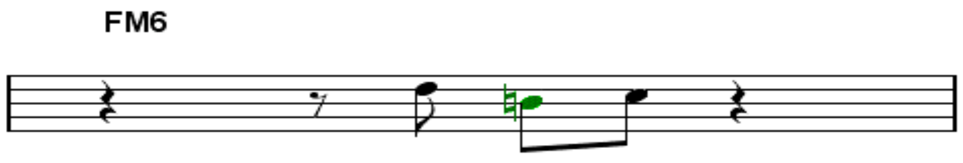
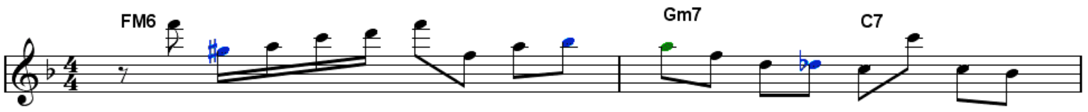


Figure 3.1: Enclosure

We represent the lick in figure 3.1[11] with the S-expression (R4 R8 S8 (slope -3 -4 S8)(slope 1 2 C8) R4). "R4" and "R8" represent a quarter rest and eighth rest respectively. We then have an eighth note scale tone followed by a scale tone three to four half steps down, a chord tone one to two steps up, and finally a quarter note rest. This abstraction can be instantiated over any chord sequence. Note that we only abstract pitches; rhythms are captured exactly.

[10] For more on use of contours for musical representation, see [Kim et al. 2000].
[11] The colors in figures 3.1 – 3.3 represent the different note categories – black for chord tones, green for color tones, blue for approach tones, and red for others.

In addition to short idioms, we can capture larger selections like the line below from a Red Garland solo.



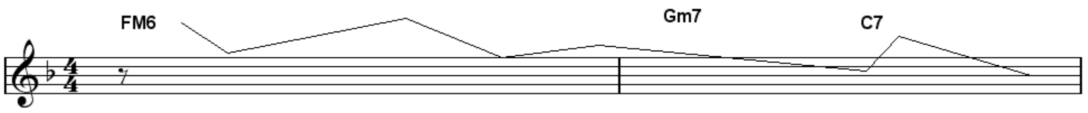


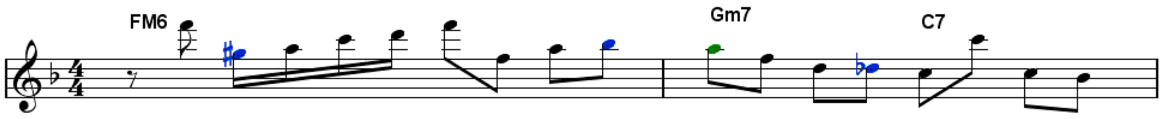
Figure 3.2: Melody line and slope representation

We represent the melody in figure 3.2 with another S-expression:

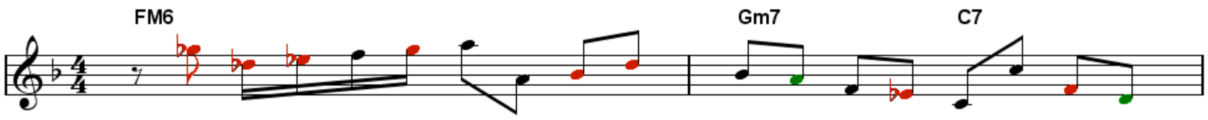
(R8 C8 (slope -9 -9 A6) (slope 1 3 C16 C16 C16 C8) (slope -12 -12 C8) (slope 1 4 C8 A8) (slope -4 -1 L8 C8 C8 A8 C8) (slope 12 12 C8) (slope -12 -2 C8 C8)).

Notes such as the second note in the first measure have only one interval from which to choose minimum and maximum slopes. In these cases, we found that relaxing the bounds by a half step in each direction yielded better results, so we translate "(slope -9 -9 A16)" to "(slope -8 -10 A16)" before instantiating. Figure 3.3 demonstrates several new licks generated from our abstraction of Red Garland's melody.

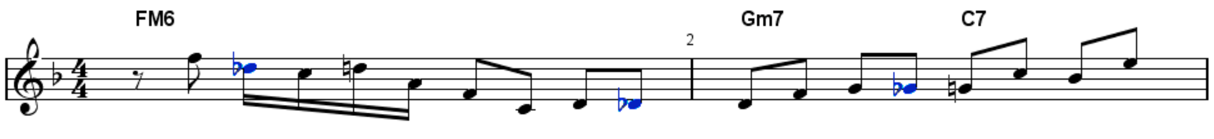
Original Melody



Generation using contour



Generation using note categories



Two generations using contour and note categories

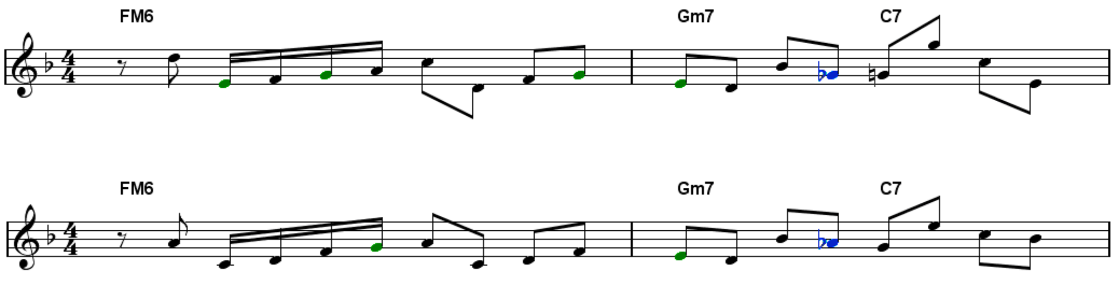


Figure 3.3: Original melody and variations

3.4.2 Recombination

Once we have our units for recombination – abstract measures – we need to decide how to combine them to create new solos. We implement the low level connection between units with a bigram Markov chain. To create a Markov chain, though, we need some way of grouping measures to avoid recreating sequences from training data.

3.4.2.1 Clustering

Grouping similar points in a data set based on some characteristics is known as clustering. We use an unsupervised learning algorithm known as k-means clustering [MacQueen 1967]. This algorithm uses an iterative process to collect data points together in an n-dimensional space. In our implementation, we use seven characteristics of abstract measures, assign values for each parameter, and determine the distance between points with a Euclidean distance metric. Several other methods for determining melodic or rhythmic similarity have been explored, such as the *Swap* *distance* and the *Hamming distance* [Ebert 2008], but these metrics are designed for traditional pitch and duration musical representation, as opposed to our representation based on contour and note categories. Table 3.2 shows the clustering parameters and their weights.

Parameter	**Weight**
Location of the first note struck within the measure	1.0
Consonance[12]	0.5
Number of notes	1.2
Total duration of rests	1.0
Average maximum slope of ascending or descending groups of notes[13]	1.0
Whether the measure starts on or off the beat	1.3
Order of the contour (how many times it changes direction	1.1

Table 3.2: Parameters for clustering

We found, after some trial and error, that these parameters grouped measures effectively, in particular because they weight rhythm higher than contour, and contour higher than harmony, which we found to be important for genuine sounding recombinations. Given a number of clusters k, the k-means algorithm randomly selects k points in space as cluster centers, and then iterates the following two steps for a number of iterations proportional to the size of the data set or until few enough data points switch clusters between iterations:

[12] We assign a "consonance" value to a measure based on the note categories. For each note, we add to the consonance value a coefficient for the note category multiplied by the length of the note. The coefficients are 0.8 for a chord note, 0.6 for an approach note, 0.4 for a color note, and 0.1 for other notes.

[13] Each ascending or descending segment has a maximum and minimum (absolute value) slope. We take the average of the maximum slopes of all segments in a measure to get this parameter.

1. Assign each data point to the nearest cluster center.
2. Recalculate cluster centers.

Figure 3.4 shows instantiations of three abstract measures that the algorithm clustered together from a corpus of Charlie Parker solos. The top line of figure 3.5 shows two measures of a melody, and the bottom line shows two measures instantiated from the same clusters as in the top line.

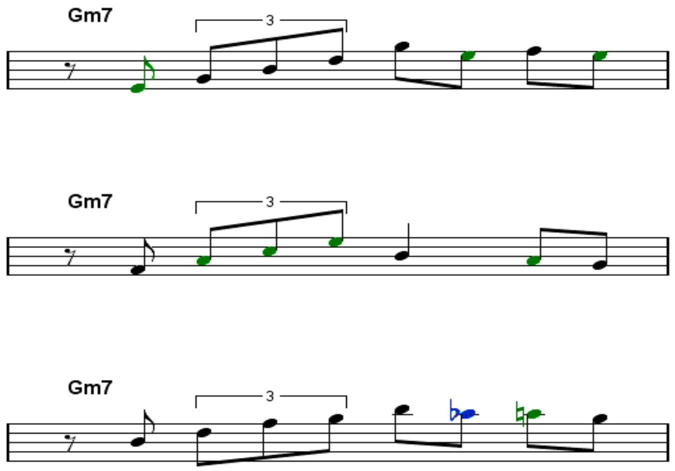


Figure 3.4: Three representatives from the same cluster

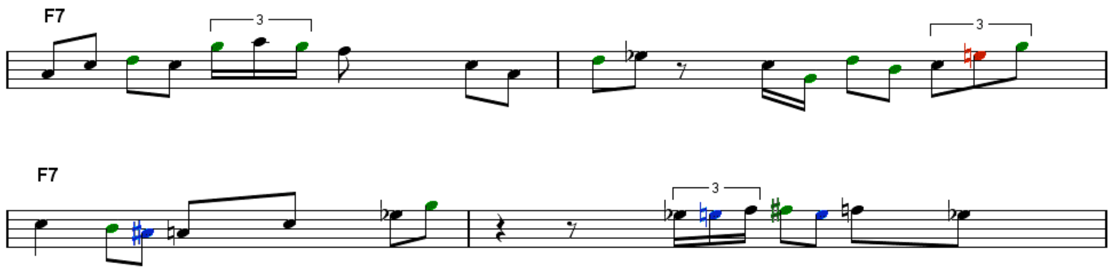


Figure 3.5: Original two-measure melody and two measures from the same clusters

Once we have a set of clusters, we go back through the training data, map each measure to its associated cluster, and build up a table of transition probabilities between clusters. Given a set of clusters and transition probabilities, we can generate new solos based on a Markov chain. We implemented bigram, trigram, and 4-gram models, but given the limited size of our data sets, higher-order N-grams added no additional information.

For short solos of 4 to 6 bars, all three models regularly produce good results that could easily be mistaken for the original artist, but longer solos tend to lack a sense of direction. The 4-gram model tends to produce longer coherent solos than the bigram or trigram, with about 25% of generations over 12 or 16 measures sounding well structured (as measured by my listening).

3.4.2.2 Global Structure – the Outline Method

To give high-level structure to solos, we base each generation on one solo from the training data. For each transcription in the training data, we store a vector of the clusters for each measure and refer to this list as an outline. A 32-measure solo, then, is represented by a vector of 32 clusters. If we want to generate over a chord progression of a certain number of measures, we pick one of the available outlines. This means that we can only use the outline method over tunes of lengths that we have in the training data, but since jazz tunes often come in one of several standard lengths, we can use this method to generate over many songs even with limited data.

In our representation, we set the clusters to have, on average, ten data points. The data is not likely to be evenly spread, so some clusters are larger than others, but within a cluster, there is, on average, a ten percent chance of choosing a particular representative. For a 32 bar song, then, given an outline of 32 clusters, there are approximately 10^{32} possible combinations of abstract measures to fill in, with each measure able to be instantiated in several ways as well. If we generate two solos over the same outline, however, the chance of a measure being selected at the same point in each is 1/10. The expected value of common abstract measures between two solos is 3.2, and the probability of one or more in common, given by a binomial distribution with parameters n=32 and p = 1/10, is 0.97. Pairs of measures could be repeated as well – there are 31 pairs of measures, and the probability the same pair of measures appears at a given spot in two generations is 1/100, so the probability of seeing one or more repeated pairs of abstract measures, also given by a binomial distribution, is about 0.27. Seeing the same three consecutive measures in two solos is less likely, with probability about 0.03.

Seeing measures repeated in the same location is not necessarily bad, since artists do tend to play similar licks at the same point during their solos. Also, measures are instantiated differently each time, so the probability of the exact same melody being played for an entire measure in two generations is small. Suppose, however, that in a solo in the training data, measures from clusters A and B appeared next to each other, but those clusters never showed up together throughout the rest of the corpus. Then it might not be best to use measures from clusters A and B every time we generate from that outline, since the original artist would be unlikely to play them together regularly.

To avoid this problem and to further increase the space of possibilities in order to prevent solos from sounding like the basis for their outline, we use what is known as hierarchical clustering. The idea of hierarchical clustering is to maintain a tree of clusters with data points partitioned in the leaves of the tree, and membership in a non-leaf node defined by membership in any descendant. The k-means algorithm gives the original partition of clusters, and then, to build the tree, we determine each cluster's three nearest neighbors, with distance between clusters A and B given by the average pairwise distance of 100 pairs of points randomly selected from A and B. So if A and B are leaves, a point in cluster A also belongs to a larger cluster C, which we say is above A in the hierarchy, and a point in B belongs to a larger cluster D. Now, given the tree, if we see cluster A in our outline, we can choose a measure from any of the clusters in C. Then, when we see B next in the outline, we use the bigram probabilities transitioning from cluster C to choose which cluster in D we will take our next measure from.

3.4.3 Additional Constraints

When making the transition from a vector of clusters given by an outline to an actual solo, we apply a couple of additional constraints. First, although we do choose representatives randomly from clusters, we choose with higher probability abstract measures that were initially played over the same chords as the chords in the measure for generation (with transpositions considered equivalent). For example, the first measure of Charlie Parker's tune "Ornithology," is played over an A major 7 chord. If we are generating a solo over Ornithology, given an outline, we will look within the first cluster of the outline for measures that were originally played over some major 7 chord and choose those measures with twice the probability of the rest. Measure 10 of Ornithology consists of a B minor 7 chord followed by an E dominant 7 chord, so if the 10th cluster of the outline contains an abstract measure that was originally played over a C minor 7 and an F dominant 7 chord, we are likely to pick that measure.

We apply the other main constraint when instantiating the notes themselves. After choosing representatives from each cluster, we have one long abstract melody to translate into a real melody using slopes and note category constraints. To these, we add a third constraint by looking at the head (main melody) of the song we are generating on (skipping this step if we don't have a transcription of the head). When choosing pitches within an interval given by a slope, if one of the notes in the interval is the pitch sounding at the corresponding time in the head, we will with some probability choose that pitch even if it conflicts with the desired note category. Back when we break up the solo transcriptions for clustering, we collect statistics on every song in the training data for which we have the head, determining a value for the percentage of the time that the artist plays the same note in a solo as in the corresponding place in the head. The top line in Figure 3.6 shows the first measure of the head of Charlie Parker's "Laird Baird," and the bottom line shows the first measure of a Parker solo over the tune.

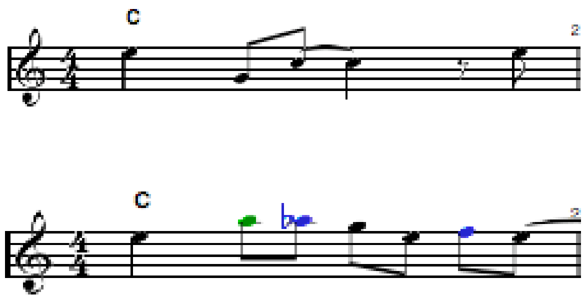


Figure 3.6: Charlie Parker lines played at the same point in the head and in a solo

Of the 4 beats in this measure, Parker plays the same pitch for a beat and a half. From our data, most players played the same pitch class[14] in solos as in the head

[14] Notes of the same pitch or differing by some number of octaves are said to be in the same pitch class.

about 20 percent of the time, whereas generating solos without adjusting to the head usually yields the same pitch class 10 to 15 percent of the time. In order to raise the percentage to around 20, we probabilistically choose some notes specifically to be the same pitch class as in the head.

Chapter 4 – Results

Music is a personal and subjective phenomenon, so quantifying the success of our work is a difficult task. Without a clear gauge for the quality of our program, we attempted to measure our results through listening tests on human subjects. One way to test our music is to play it for a sample of the population and ask them to give it a grade. This test has a couple of shortcomings. First, many factors influence the way people rate a song besides the song itself – how familiar listeners are with the genre or song, what mood they are in, what expectations they have, etc. People also tend to be influenced by preconceived notions that they have about the song or artist, and these conceptions are inclined to be particularly strong with computer-composed music. Secondly, our program outputs music in MIDI form. Even state of the art synthesizers do not come close to the sound of human performances on real instruments, particularly horns, the instruments of most of the artists in our training data. Synthesizer performance is a research area of its own that we do not cover in this thesis. We focus on the composition of music, not the performance, so we want to make sure to isolate the composition aspect of the music for assessment.

Consequently, we propose as a test a version of David Cope's "Game," which is based on a traditional method for evaluating artificial intelligence known as the Turing Test [Turing 1950]. The setup of the Turing Test is as follows: Three players A, B, and C sit in separate rooms. A is a computer and B is human, and the task of C, a human judge, is to determine which is which. C can ask any questions of the two but is limited to written communications to make a decision. A machine passes the

Turing Test if it answers in such a way as to make C guess incorrectly just as often as he guesses correctly.

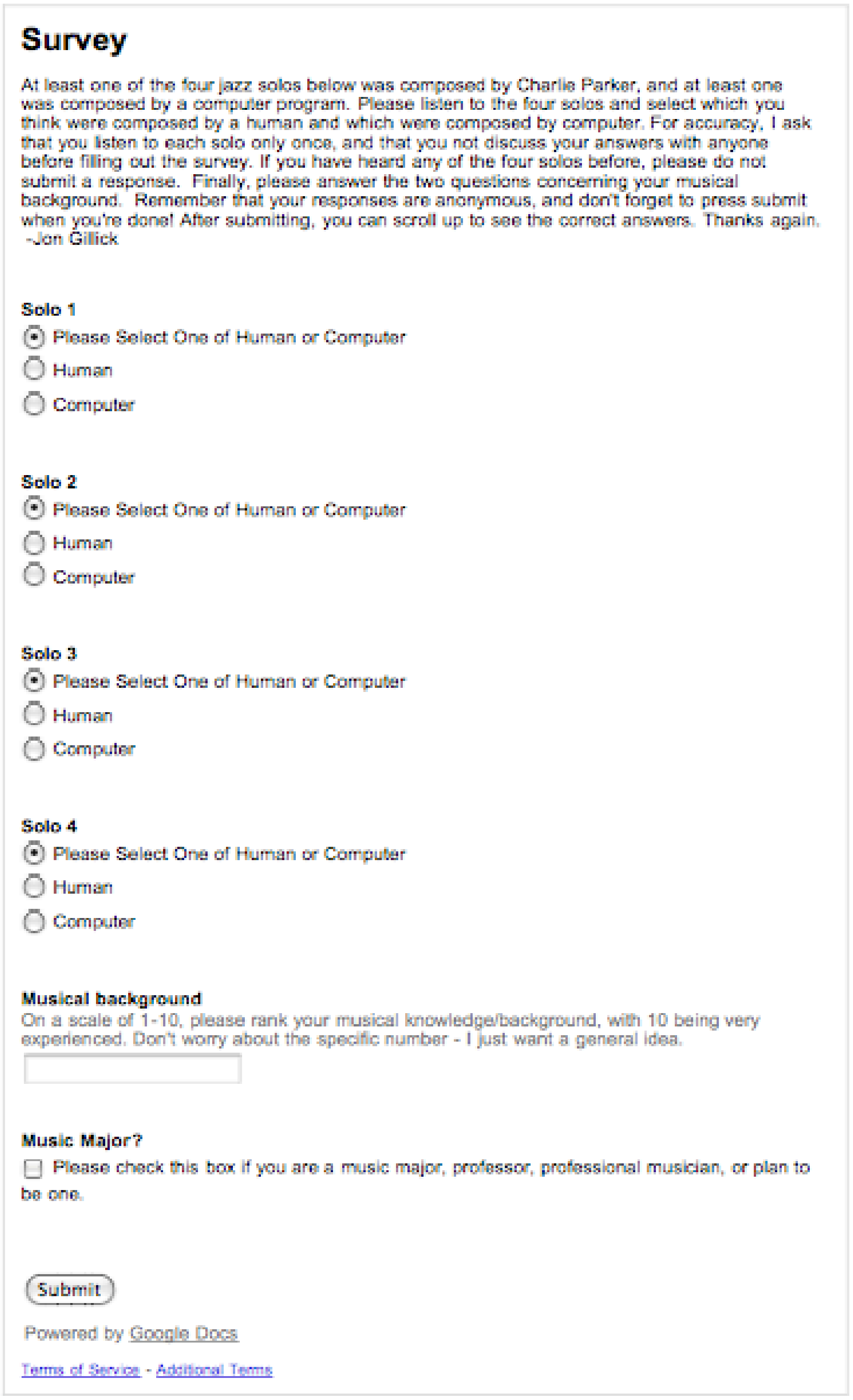
4.1: Human Versus Computer

In our test, we try to determine whether listeners can tell the difference between jazz solos composed by Charlie Parker and solos composed by Riff Jr. "in his style." We don't attempt to show that our program demonstrates intelligence, but we do claim that passing the test validates the quality of its compositions.

We posted four solos to a website, two by Charlie Parker and two by Riff Jr., asking test subjects to identify the composer of each. For the computer-generated solos, we selected examples that, based on listening to a large number of solos, sound better than the average generation but were not outliers. We used Parker in part because he is the artist for whom we have the most data (396 measures), but also because he is one of the fathers of jazz improvisation and one of the most commonly imitated players.

To conduct the survey, we solicited volunteers from Wesleyan's math and computer science departments, as well as the Yahoo group for registered Impro-Visor users.[15] Figure 4.1 shows a screenshot of the test website, and Table 4.1 shows a summary of the results.[16]

[15] groups.yahoo.com/group/impro-visor/
[16] The survey and audio files can be accessed at jrgillick.web.wesleyan.edu



Survey

At least one of the four jazz solos below was composed by Charlie Parker, and at least one was composed by a computer program. Please listen to the four solos and select which you think were composed by a human and which were composed by computer. For accuracy, I ask that you listen to each solo only once, and that you not discuss your answers with anyone before filling out the survey. If you have heard any of the four solos before, please do not submit a response. Finally, please answer the two questions concerning your musical background. Remember that your responses are anonymous, and don't forget to press submit when you're done! After submitting, you can scroll up to see the correct answers. Thanks again. -Jon Gillick

Solo 1

- [✓ Please Select One of Human or Computer]
- Human
- Computer

Solo 2

- [✓ Please Select One of Human or Computer]
- Human
- Computer

Solo 3

- [✓ Please Select One of Human or Computer]
- Human
- Computer

Solo 4

- [✓ Please Select One of Human or Computer]
- Human
- Computer

Musical background

On a scale of 1-10, please rank your musical knowledge/background, with 10 being very experienced. Don't worry about the specific number - I just want a general idea.

Music Major?

☐ Please check this box if you are a music major, professor, professional musician, or plan to be one.

Figure 4.1: Test Website

	Solo 1	Solo 2	Solo 3	Solo 4
Correct Answer	Human	Human	Computer	Computer
# Human	36	56	58	59
# Computer	84	64	62	61
# Correct	36	56	62	61
% Correct	30	47	52	51

Table 4.1: Test results

Results clearly showed that listeners could not tell the difference between the Parker solos and the computer-generated solos. 120 test subjects submitted guesses about the identity of each solo's composer, and 209 out of the 480 total responses, 45 percent, were correct. Only five participants, or 4 percent, correctly identified all four, a fraction less than the expected value of 7.5 participants, or 6 percent, if every subject guessed.

We also asked participants to rank their musical background from 1 to 10 and to disclose whether they were music students or professionals. The group of 26 subjects who identified themselves as such fared similarly to the rest, averaging 48 percent correct answers as opposed to 44 percent for non-musicians. The group of 57 who ranked their musical background 6 or higher averaged 43 percent, while the 63 who rated their background 5 or lower averaged 47 percent.

One statistic stands out about the data. In the test, we labeled the examples "Solo 1," "Solo 2," "Solo 3," and "Solo 4," with solos 1 and 2 by Charlie Parker and 3 and 4 computer-generated. While subjects identified solos 2, 3, and 4 with 47, 52, and 51 percent accuracy respectively, 70 percent incorrectly marked solo 1 as computer generated. The probability of 70 percent incorrect or greater over 120 responses, given by a binomial distribution, is about 6.9 in a million, so this result was clearly influenced by a non-random variable. Since disproportionately many people answered incorrectly rather than correctly, though, the variable was not that subjects could tell the difference. We hypothesize that the order in which the solos were presented caused this result – many listeners, unused to the sound of MIDI, marked the first one they heard as computer. We expect that this problem would be fixed by repeating the experiment with the order of songs randomized for each subject. This result also raises the concern that perhaps MIDI sounds so bad that nobody can tell any MIDI solos apart at all. With the version of the program using HMM's and not the outline method, however, 10 out of 10 subjects in a small scale test given to friends correctly identified which of two solos was really Charlie Parker.

4.2: Artist Matching

Another test, although performed with an earlier version of the program, merits inclusion here. In this test, to see how well generated solos capture the style of artists, we set up an experiment to determine whether or not test subjects could match the styles of three prominent jazz trumpet players with solos composed in the style of each player. We created models for Clifford Brown, Miles Davis, and Freddie Hubbard from 72 bars of solos from each and then played the subjects one clip from each artist and one clip composed by each model, with each computer solo generated over the same tune (Ray Henderson's "Bye Bye Blackbird"). Without revealing the names of the artists, we asked them to match the artists from the computer-composed solos with the human players. We also asked the participants to qualitatively indicate how close the resemblance was by "Not close," "Somewhat close," "Quite close," or "Remarkably close." [Gillick et al. 2009]

Out of 20 test subjects, 95 percent correctly identified Clifford Brown, 90 percent identified Miles Davis, and 85 percent identified Freddie Hubbard. 85 percent correctly matched all 3 solos. All subjects characterized the resemblance to the original artists as either "Somewhat close" or "Quite close", with 9 votes to "Somewhat close," 10 to "Quite close" and 1 unable to decide. 50 percent ranked their own musical knowledge between 2 and 5, and 50 percent between 6 and 9.

Chapter 5 – Conclusions

That test subjects were unable to tell the difference between Charlie Parker and Riff Jr. shows that our method for style emulation is effective. Clustering has proven to be a viable means of grouping licks, and abstraction of licks using contours and note categories has shown to yield a workable balance between novelty and correctness of style.

Our method also demonstrates improvements over other systems in terms of flexibility, ease of use, and accuracy of imitation. While GenJam and most other genetic algorithms based systems require user training and feedback for optimal performance, Riff Jr. depends only on an initial corpus of training data, which lessens both overhead and user-induced variables.

In addition, Riff Jr. represents improvements to the original Impro-Visor system it was built upon [Keller and Morrison 2007]. Bob Keller remarks: "Jon Gillick's improvements to Impro-Visor go a long way toward humanizing generated solos. For example, space is left in opportune places, while connected melodic lines are extended in length, thanks to more attention being paid to global construction of the solo."

As Keller says, the most important difference between Riff Jr. and other jazz improvisation systems is the inclusion of a model for structure from start to finish by basing each generated solo at a high level on an outline of one original solo. For best results, some form of global structure seems necessary, as n-grams are usually insufficient even given a reasonable amount of training data. David Cope effectively incorporates global structure into EMI's classical compositions and successfully runs listening test, but we appear to be the first to use the outline method for jazz improvisation, as well as the first to run a version of the Turing Test on a jazz improvisation system.

5. 1: Continuations

Our work merits further exploration in all main areas – musical representation, algorithms, and data analysis. First, jazz musicians tend to comment that the main problem with Riff Jr.'s solos is that they often modulate into incorrect keys. We currently determine appropriate scales (and note categories) only by one chord. Some chords fit into several keys though, so we could make better scale choices by looking at multiple chords at a time. In terms of abstracting melodies, we currently apply only three constraints to pitches – slope, note category, and relationship to the corresponding pitch in the head of the tune. Examining more specific information about notes, such as interval from the root of a chord, could prove to be beneficial.[17] Second, while our clustering algorithm works well, there certainly exist better parameters than the 7 we describe in chapter 3. Finally, we currently only use standard transcriptions for our musical data. Examining microtiming for pitch and rhythm as in [Ramirez at al. 2008] could better capture inflection, timbre, and expressiveness. We also have only used data sets of at most 400 measures per artist. 400 proved to be significantly better than 50 or 100 measures, so a data set of 10,000 measures might give much better results.

The limits of algorithmic composition and improvisation will not soon be reached. One could imagine a program able to imitate all aspects of Charlie Parker so

[17] For more about note categories and scale choices, see http://www.cs.hmc.edu/~keller/jazz/improvisor/Scales.html

well as to be, for all practical purposes, a musical clone. When and whether such a program will ever exist is just one of the vast number of questions raised by the rapid progress of artificial intelligence, but striving to create it can help us better understand the fundamental workings of human creativity.

References

[Ames, et. Al 1992] Ames, Charles and Domino, Michael. “Cybernetic Composer: An Overview.” *Understanding Music with AI*, edited by Mira Balaban, Kemal, Ebcioglu, and Otto Laske. The MIT Press: Cambridge, MA, 1992.

[Biles 1994] Biles, J. A. GenJam: A Genetic Algorithm for Generating Jazz Solos, *Proceedings of the International Computer Music Conference*, p.131-137, IGMA, San Francisco, 1994.

[Chomsky 1956] Chomsky, Noam. “Three models for the description of language," *Information Theory, IEEE Transactions* 2 (3), p. 113–124.

[Cope 2001] Cope, David. *Virtual Music: Computer Synthesis of Musical Style*. The MIT Press: Cambridge, MA, 2001.

[Cope 2005] Cope, David. *Computer Models of Musical Creativity*. The MIT Press: Cambridge, MA, 2005.

[Ebert 2008] Ebert, Manuel. “A Phenomenological Approach to Artificial Jazz Improvisation” (BS thesis, University of Osnabruck, 2008)

[Gillick 2003] Gillick, Daniel. “Comparing Methods for Authorship Attribution” (BA thesis, Wesleyan University, 2003)

[Gillick et al. 2009] Gillick, Jonathan, Kevin Tang and Robert Keller. “Inferring Probabilistic Grammars for Jazz Solo Generation,” to be submitted, 2009.

[Gridley 1985] Gridley, Mark C. *Jazz Styles: History & Analysis*, Prentice-Hall, inc. Englewood Cliffs, NJ, 1985.

[Hofstadter 1979] Hofstadter, Douglas. *Godel, Escher, Bach: An Eternal Golden Braid*. Basic Books, Inc: New York, NY, 1979.

[Keller and Morrison 2007] Keller, Robert M. and David R. Morrison. “A Grammatical Approach to Automatic Improvisation,” *Fourth Sound and Music Conference*, Lefkada, Greece, 2007.

[Kim et. Al 2000] Kim, Youngmoo, Wei Chai, Ricardo Garcia, and Barry Vercoe. “Analysis of a Contour-based representation for Melody,” *Proceedings of International Symposium on Music Information Retrieval*, 2000.

[Kurzweil 1990] Kurzweil, Ray. *The Age of Intelligent Machines*. The MIT Press: Cambridge, MA, 1990.

[MacQueen 1967] MacQueen, J. “Some methods for classification and analysis of multivariate observations,” *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume I, Statistics*. Edited by Lucien M. Le Cam and Jerzy Neyman. University of California Press, 1967.

[Mozer 1994] Mozer, M. C. “Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing.” *Connection Science*, 6 (2-3), p. 247–280, 1994.

[Owens 1995] Owens, Thomas. *Bebop: The Music and Its Players*, Oxford University Press: Oxford, 1995.

[Papadopoulos and Wiggins 1999] Papadopoulos, George and Geraint Wiggins. “AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects,” *Proceedings of AISB Symposium on Musical Creativity*, 1999.

[Papadopoulos and Wiggins 1998] Papadopoulos, George and Geraint Wiggins. “A Genetic Algorithm for the Generation of Jazz Melodies,” *Proceedings of STeP*, 1998.

[Ramirez et al. 2008] Ramirez, Rafael, Amaury Hazan, Esteban Maestre and Xavier Serra. “A genetic rule-based model of expressive performance for jazz saxophone.” *Computer Music Journal*, Volume 32, Issue 1, p. 38-50, 2008.

[Searle 1985] Searle, John. “Minds, Brains, and Programs.” *Mind Design* (1985), p. 282-307.

[Sivaraman 2008] Sivaraman, Shyam. “K-means Cluster Analysis Algorithm.” http://www.sourcecodesworld.com/source/show.asp?ScriptId=1227.

[Turing 1950] Turing, Alan. “Computing Machinery and Intelligence.” *Mind*, Volume 59, No. 236, p. 433-460.

[Xenakis 1971] Xenakis, Iannis. *Formalized Music: Thought and Mathematics in Music*. Indiana University Press: Bloomington, IN, 1971.

[Wilson 2008] Wilson, Rachael, “Jamming Robot Puts the Rhythm into Algorithm.” *Science Alert*, 23 October 2008. Magazine on-line. Available from http://www.sciencealert.com.au/news/20082310-18331.html. Accessed 4 April 2009.

Appendix A – Artists and Songs Used

All songs used in this thesis are included with Impro-visor[18] and were taken originally from various jazz fake books, except for “Dizzy Atmosphere” and “Ornithology,” which were transcribed by Charles McNeal.[19]

Artist	Song
Clifford Brown	Now’s the Time
John Coltrane	Giant Steps
John Coltrane	Moment’s Notice
Miles Davis	On Green Street Dolphin
Bill Evans	What Is This Thing Called Love
Red Garland	Bye Bye Blackbird
Dizzy Gillespie	Groovin High
Tom Harrell	Have You Met Miss Jones?
Tom Harrell	Little Dancer
Tom Harrell	Solar
Coleman Hawkins	Body and Soul
Freddie Hubbard	Byrd Like
James Moody	Con Alma
Lee Morgan	Ceora
Lee Morgan	Moment’s Notice
Charlie Parker	Anthropology
Charlie Parker	Cheryl
Charlie Parker	Dewey Square
Charlie Parker	Dizzy Atmosphere
Charlie Parker	Laird Baird
Charlie Parker	Moose the Mooche
Charlie Parker	Now’s the Time
Charlie Parker	Ornithology
Charlie Parker	Scrapple from the Apple
Charlie Parker	Yardbird Suite

Table A-1: Artists and songs used

[18] Available for download at groups.yahoo.com/group/impro-visor/
[19] Available at charlesmcneal.com

Appendix B – Code and Implementation

All code was written in Java version 1.5 using NetBeans IDE[20] and the Java Polya library, which provides Lisp-like lists and I/O in Java.[21] The software for this thesis was implemented within the source code for Impro-Visor[22], which handles the interface, Main class, scale and chord vocabulary, and note category identification, among other functions. The next release of Impro-Visor will contain this new code. Skeletons for the classes used for k-means clustering (JCA, Cluster, Centroid, and DataPoint) were provided by [Sivaraman 2008]. Table B-1 shows the major classes and functions implemented or modified for this thesis.

Class	**Purpose**
DataPoint	This class represents an abstract melody. It contains fields for each of the seven parameters used in cluster analysis.
Cluster	This class represents a Cluster object. It is associated with a vector of DataPoints, a Centroid, and a JCA instance. It contains methods for retrieving random DataPoints and checking if DataPoints match the current chord sequence during solo generation.
Centroid	This class represents the Centroid, or center point in space, of a Cluster. The initial Centroids for each Cluster are spaced evenly in each of the seven dimensions, depending on the value of k.
JCA (Java Cluster Analysis)	This class runs the k-means clustering algorithm. It contains an array of clusters and a vector of DataPoints. JCA iteratively assigns DataPoints to the nearest cluster and then recalculates each Centroid.

[20] http://www.netbeans.org.
[21] http://www.cs.hmc.edu/~keller/polya/
[22] http://launch.groups.yahoo.com/group/impro-visor/files/Source%20files/

ClusterSet	This class represents a step up from the bottom in the cluster hierarchy. It contains a vector of the closest relatives of a Cluster, as well as methods for computing pairwise distances between Clusters.
NGram	This class contains information about the first n-1 states of an N-gram and vectors of the possible next states along with probabilities. It contains methods to probabilistically choose the next state in a Markov Chain.
CreateGrammar	This class contains the method to create a model for a soloist and is invoked by a button in the Impro-Visor interface. It contains methods for loading data from files into DataPoints, calling the clustering algorithm, and writing the outlines (vectors of ClusterSets) to a file.
Notate	This class sets up Impro-Visor's GUI, including the button to generate solos. It contains methods for loading outlines from a file and initializing solo generation.
LickGen	This class generates a solo from an outline. It picks a random outline from the available set, choosing clusters from the outline and then abstract measures from the clusters. It also instantiates abstract melodies.
NoteChooser	This class chooses a pitch for a note given a previous pitch, slope, and note category. It contains a method to return a particular note based on a probability table.

Table B-1: Classes and Purposes