# *ARTIST*: a Real-Time Improvisation System

Jason Brooks
Yale University
51 Prospect Street
New Haven, CT 06520
jason.brooks@yale.edu

Kevin Jiang
Yale University
51 Prospect Street
New Haven, CT 06520
k.jiang@yale.edu

Charles Proctor
Yale University
51 Prospect Street
New Haven, CT 06520
charles.proctor@yale.edu

## ABSTRACT

This paper presents ARTIST, a real-time improvisation system for jazz music. The ARTIST system allows for both the generation of jazz improvisation over a given chord progression, as well as the performance of that generation on a xylophone using the Baxter robotics platform. In order to do so, the pipeline uses a novel hybrid algorithm that bridges an n-gram statistical model with a genetic algorithm to allow for fast generation without sacrificing the quality of the music. Results from an evaluation pipeline demonstrate that ARTIST is a promising start for computational jazz improvisation.

## 1. INTRODUCTION

As robots become more integrated in human life, their effectiveness will increasingly depend on their ability to seamlessly communicate with human counterparts. ARTIST aims to tackle this problem of human-robot communication in the domain of jazz improvisation. We present a system that can dynamically, appropriately, and creatively play with human jazz musicians.

Music in general, especially jazz, lives within a series of constraints. For example, when selecting notes during improvisation, a player typically chooses within the chord and mode. Furthermore, with the goal of performing on a robotics platform, there are limitations on the sequences of notes and durations that can be played.

DARPA has already shown interest in similar jazz improvisation systems with the expectation that examining this problem in a restricted domain will produce insights that can be applied to military robotics [1]. We hope that engineering ARTIST will not only result in the development of a novel jazz improvisation technology, but will also inform broader areas of human-robot communication for future research.

ARTIST works in a six stage pipeline. First, large repositories of Musical Instrument Digital Interface (MIDI) files are mined from Internet sources, each of which contains a computational representation of the composed jazz song. The chord progressions are extracted from the various songs and a trigram model is used to extract frequent patterns. Via a novel algorithmic pipeline, jazz improvisation is generated for a given chord using a combination of a statistical trigram model and a genetic algorithm. Finally, the improvisation is played via a robotic performer through a set of generated position transformations.

The rest of this paper details how each of these stages is accomplished.

## 2. PROCESSING MIDI DATA

To download the jazz MIDI files, a multi-threaded MIDI file scraper was implemented. Given a URL, the scraper discovers and downloads any linked MIDI files. Drawing from sites such as www.midkar.com, over 2,000 songs on which to train were downloaded:

| | |
|---|---|
| Songs | 2,082 |
| Tracks | 32,340 |
| Notes | 11,178,141 |

**Table 1: Jazz MIDI Sample Size**

Once the MIDI files were downloaded, a MIDI parsing pipeline was implemented to extract relevant features into a PostgreSQL database. The features of the MIDI track particularly important for future algorithmic music generation included information such as instrument track key signature, time signature, note on/off events with pitches and timing information, tempo markings, and instrument type. Each MIDI file was represented in the database by an entry in the song table; each song had many tracks, which represented a contiguous sequence of notes played by one instrument in a particular key and time signature; each track had many notes corresponding to relevant note on/off event information.

The harmonic analysis pipeline utilized the stored information in the PostgreSQL database of jazz music. Given the size of our data, the aforementioned MIDI parsing and harmonic analysis programs were too slow to run in a single process. Therefore, the following multi-processing distributed algorithm was developed, implemented in Python:

- For a computer with $n$ cores, let $p = \frac{3}{2}n$ be the size of the process pool used.
- Create $p$ separate PostgreSQL databases, one associated with each process. This limits the potential for database reading / writing being the bottleneck.
- For MIDI parsing, create a shared queue of songs to parse. Each separate worker process pulls from the queue and parses the song pulled into its database, until all songs have been processed.

- Create $p$ separate harmonic analyzer processes, one for each database. Each analyzes the songs in its database, until all processes have completed.
- Similarly, run the trigram trainer separately on each database and combine the results.

For perspective, MIDI parsing takes approximately 1 hour on an 8-core machine, whereas harmonic analysis as described below takes on the order of 24-36 hours to complete on the same machine.

## 3. HARMONIC ANALYSIS

Jazz improvisation relies on the knowledge of the current chord being played in the song. Unfortunately, none of the jazz MIDI tracks scraped from the Internet in the MIDI parser pipeline included chord progression information. Therefore, the following preference rule system for chord extraction using a dynamic programming approach was implemented as outlined in *The Cognition of Basic Musical Structures* by David Temperley [7].

In order to formalize the task of extracting chord progressions from unlabelled MIDI tracks, a *chord-span* was defined as a contiguous period of time during which the song (with all of it's various tracks) is labelled with one chord (with a unique root).

Given a song consisting of a series of unlabelled MIDI tracks, the goal is therefore to compute the optimal set of chord-spans, subject to the following two constraints:

- No two chord-spans can overlap.
- Every note must be in a chord-span.

### 3.1 Preference Rule System

The system starts with the following two definitions:

- The *line-of-fifths* is a linearized version of the circle-of-fifths. Namely,

$$\ldots B\sharp, E\sharp, A\sharp, D\sharp, G\sharp, C\sharp, F\sharp,$$
$$B, E, A, D, G, C, F,$$
$$B\flat, E\flat, A\flat, D\flat, G\flat, C\flat, F\flat \ldots$$

- The *strength of a beat* refers to the beat's alignment with the start of
  - the measure,
  - aligned half-notes (beats 1,3 in $\frac{4}{4}$),
  - aligned quarter-notes (beats 1,2,3,4 in $\frac{4}{4}$).

The three preference rules are as follows:

1. Compatibility: Prefer certain intervals between notes and the chord's root, in the following order:

$$\hat{1}, \hat{5}, \hat{3}, \flat\hat{3}, \flat\hat{7}, \flat\hat{5}, \flat\hat{9}$$

2. Strong Beats: Prefer chord-spans that start on strong beats of the meter. The stronger the beat (as defined previously), the greater the preference.
3. Harmonic Variance: Prefer chord roots close to nearby segment roots on the line-of-fifths.

The preference rules combine together to calculate a score for a particular chord-span.

### 3.2 Dynamic Programming

Following the standard dynamic programming approach, the chord-spans for an entire song can be extracted using the aforementioned preference rules:

**Algorithm 1** Extract chord-spans using preference rules

```
 1: Create a chord-span with the first note(s) played.
 2: for every time $t$ in the song do
 3:     Let $c$ be the chord of notes playing at time $t$.
 4:     Consider adding $c$ to the previous chord-span. Let
        the preference score obtained be $a$.
 5:     Consider adding $c$ to a new chord-span. Let the score
        obtained in the previous chord-span be $p$ and the score
        obtained in the new chord-span be $b$.
 6:     if $a > p + b$ then
 7:         Add $c$ to previous chord-span.
 8:     else
 9:         Create new chord-span, starting with $c$.
10:     end if
11: end for
```

Using this algorithm, the entire database of songs was successfully labelled with chord progressions.

## 4. TRIGRAM MODEL

Once the music in the database is labelled with the appropriate chord-spans, a trigram model is run on the note pitches.

As input, the trigram model takes a key $K$ in which to generate and a chord progression $C$, where $C(t)$ is the underlying chord at time $t$ (given by the roman numeral).

The trigam model seeks to produce a generated song, represented as a list of `(pitch, duration)` pairs,

$$(p[1], d[1]), (p[2], d[2]), \ldots, (p[n], d[n])$$

The trigram model runs on pitches only. The chosen duration $d[i]$ is always a quarter-note. Therefore, the ngram model seeks to produce a series of pitches, $p[1], p[2], \ldots, p[n]$. Using a generative process, the model seeks to maximize

$$P\Big(p[1], p[2], \ldots, p[n] \Big| C\Big)$$

Now the problem is that $C(t)$ does **not** necessarily equal $C(t-1)$, since chords change within a given piece. To accommodate, the trigram training and generation was run separately for each chord. For example, on a piece of chords

$$C = [c_1, c_2, \ldots, c_n]$$

the trigram model would run separately for $c_1, c_2, \ldots, c_n$. Now let $R_k \subseteq \{1, \ldots, n\}$ be a set of times that share the same chord. In other words,

$$\forall_{i,j \in R_k} C(i) = C(j)$$

Therefore, we seek to maximize

$$\prod_k P\Big(\bigwedge_{i \in R_k} p[i] \Big| C[R_k]\Big)$$

where $C[R_k]$ is the chord for times $t \in R_k$. Obviously, the product can be separated and thus the goal is to individually maximize each

$$P\Big(\bigwedge_{i \in R_k} p[i] \Big| C[R_k]\Big)$$

for the given chord $R_k$. For convenience, the times are labelled $t \in R_k$ as $r_{k,1}, r_{k,2}, \ldots, r_{k,m}$.

The complete model can now be derived. Using the chain rule, the probability of a series of pitches in a given chord $R_k$ is

$$\begin{aligned} &P\Big(p[r_{k,1}], p[r_{k,2}], \ldots, p[r_{k,m}] \Big| C[R_k]\Big) \\ &= P\Big(p[r_{k,1}] \Big| C[R_k]\Big) P\Big(p[r_{k,2}] \Big| p[r_{k,1}], C[R_k]\Big) \\ &\ldots P\Big(p[r_{k,t}] \Big| p[r_{k,t-1}], p[r_{k,t-2}], \ldots, p[r_{k,2}], p[r_{k,1}], C[R_k]\Big) \end{aligned}$$

Using the standard trigram model, assume that each note in a given chord only depends on the prior two notes in the chord:

$$P\Big(p[r_{k,t}] \Big| C[R_k]\Big) \approx P\Big(p[r_{k,t}] \Big| p[r_{k,t-1}], p[r_{k,t-2}], C[R_k]\Big)$$

Substituting above, it is evident that:

$$\begin{aligned} &P\Big(p[r_{1,k}], p[r_{2,k}], \ldots, p[r_{m,k}] \Big| C[R_k]\Big) \\ &\approx \prod_{t=1}^{n} P\Big(p[r_t] \Big| p[r_{t-1}], p[r_{t-2}], C[R_k]\Big) \end{aligned}$$

Now, to compute a particular trigram probability, use a maximum likelihood estimation:

$$P\Big(p[r_{k,t}] \Big| p[r_{k,t-1}], p[r_{k,t-2}], C[R_k]\Big) \quad (1)$$

$$= \frac{Q\Big(p[r_{t-2}]p[r_{t-1}]p[r_t] \Big| C[R_k]\Big)}{Q\Big(p[r_{t-2}]p[r_{t-1}] \Big| C[R_k]\Big)} \quad (2)$$

where $Q\Big(abc \Big| C[R_k]\Big)$ is the count of the sequence $abc$ in chord progression $C[R_k] (= C(a) = C(b) = C(c))$.

## 4.1 Implementation

To implement the aforementioned trigram model, the algorithm is broken into two stages: training and generation.

During the training phase, all music in the database is iterated through to compute

$$Q\Big(p[r_{t-2}]p[r_{t-1}]p[r_t] \Big| C[R_k]\Big)$$

for each I-VII major chord. The algorithm transposes into the key of C-major and stores the counts in a three-dimensional `numpy` matrix.

During the generation phase,a chord progression $C$ and $K$ are taken as input. For each distinct chord in the progression, the algorithm initializes by outputting the $\hat{1}$ and $\hat{3}$ of the chord. The algorithm then proceeds to choose uniformly according to the trigram probability

$$P\Big(p[r_{k,t}] \Big| p[r_{k,t-1}], p[r_{k,t-2}], C[R_k]\Big)$$

as derived in Equation 2.

Finally, the algorithm transposes from C-major to the desired key and outputs the series of pitches generated.

## 5. GENETIC ALGORITHM

In general, genetic algorithms are efficient search methods when dealing with problems that have (1) very large search spaces that and (2) multiple correct and complex solutions.[4] Since jazz composition is an inherently creative domain, the benefits of using a genetic algorithm are magnified, as the algorithm can be seeded with a broad range of information that subsequently has the potential to generate novel results.

In general, the use of genetic algorithms in the AI community to generate music improvisations can be categorized into three different areas based on the type of fitness function used: a knowledge-based fitness function, a human user-based fitness function, and a neural network as a fitness function approach.[5][2] Often, these genetic algorithms utilize a limited search space, which compromises both the harmonic quality and overall creativity of the output melody.

We propose a genetic algorithm using a knowledge-based fitness function that takes into account both musicality-specific and performance-specific considerations. Musicality constraints were constructed based on a literature review of cognitive psychology research in music and expert interviews with professional musicians at the Yale School of Music. Performance-specific constraints involved taking into account the physical limitations of our robot performer, Baxter. The complete genetic algorithm can iterate through 800 generations in less than 30 seconds on a 1.3 GHz Intel Core i5 processor; it ultimately generates an improvised 12-bar blues melody over a given chord progression in both Baxter track and MIDI track formats.

## 5.1 Algorithm Overview

A genetic algorithm was implemented with a general structure inspired by Papadopoulos and Wiggins' genetic algorithm used to generate jazz melodies over an input chord progression.[3] The high-level steps of the algorithm are detailed below.

**Algorithm 2** Improvise jazz melody for given chord progression

```
1: Input chord progression and initialize n-gram seeded
   generation.
2: while $max - generation$ not yet reached do
3:     Elitism: preserve highest fitness chromosomes from
       previous generation.
4:     Crossover: children generated from previous genera-
       tion's parents
5:     Mutation with hill-climbing: randomly mutate chil-
       dren and keep if higher fitness
6: end while
7: Generate MIDI and Baxter tracks represented by highest
   fitness chromosome
```

A detailed breakdown of this algorithm will be explored in subsequent sections.

## 5.2 Knowledge Representation

A knowledge representation system was constructed that emphasized maximum flexibility in musical expression while still allowing for fast processing speeds and easy conversions between different key signatures and time signatures.

Specifically, the chromosomes used consist of a series of $< pitch, duration >$ pairs given in the order that they are meant to be played. Pitches are represented as an integer from 1 to 21 and correspond to scale notes; thus, for a 7-note scale, three octaves of pitch possibilities are covered. Rests are encoded as the value -1. In this way, a single genetic algorithm generation can be easily hashed into any of the fifteen conventional key signatures. Durations are encoded using durks–or 32nd notes–where 32 durks be a whole note in a 4/4 time signature. The chord progression input is encoded as a $< chord, duration >$ pair, where chord is the chord number between 1 and 7 and duration is encoded in durks as described above.

## 5.3 Initialization

The genetic algorithm can take two different categories of initial chromosomes. The first, simpler initialization involved a random generation of $< pitch, duration >$ pairs, where each possible pitch (1-21 and -1 for rests) and duration value up to a half note (1-16) is randomly chosen. A second initialization used the n-gram model described earlier as an initial seeding for the genetic algorithm. The n-gram model seeding tended to produce more musically pleasing results, while the random seeding sometimes had the advantage of producing slightly more varied and creative results. Ultimately, the final genetic algorithm was seeded using the n-gram model described.

## 5.4 Genetic Operators

### *5.4.1 Mutation*

In general, we attempted to implement musically significant mutations that modify chromosomes in a manner than maximizes the possibility of generating creative melodies. These mutations were programmed to occur for random fragments of each chromosome of a random length. The implemented mutations included reversing a fragment in time, concatenating contiguous rests and pitches, transposing a fragment up or down a certain number of pitches, altering a few pitches while maintaining the rhythm, sorting into ascending or descending pitch, permuting in time, and transposing each note by a random number of degrees. A mutation was also implemented that takes a fragment and inserts it back in at a different point of the chromosome to encourage more pattern matching and thematic development.

### *5.4.2 Crossover*

One point crossover was implemented using two parents from the previous generation selected through a weighted tournament of four chromosomes.

## 5.5 Fitness Function

In general, the overall fitness of a given chromosome was calculated using a weighted sum of each of the characteristics.

$$fitness = li + pm + su + db + hb + ln + er + ps + pop$$

The parameters in the fitness function can be categorized into two general types: jazz musicality and Baxter-specific fitness considerations. In developing the fitness function, subjective human input from listeners with different levels of musical background–skill levels ranged from professional jazz musicians to casual music listeners–was used to alter specific weights. The following section describes these characteristics, the corresponding weights associated with each characteristic, and the reasoning behind each characteristic.

### *5.5.1 Musicality Characteristics*

- **Pattern Matching** $(pm)$

  In order to create the feeling of thematic development, chromosomes with consecutive notes of similar pitch patterns are rewarded. This intuition was supported by expert interviews with professional musicians and cognitive psychology literature–specifically, the Gestalt Law of Similarity purports that repeated patterns in music is desirable for human listeners.

  A pattern matching function was implemented that allows users to specify the minimum number of notes that constitutes a full pitch pattern as well as an option to punish patterns that are simply repeated pitches. In the final implementation, each non-overlapping pattern of notes is rewarded (+10) except for patterns that repeat the same pitch, which are each punished (-5).

- **Suspension** $(su)$

  A musical suspension is a means to create tension by prolonging a consonant note while the underlying harmony (i.e. chord progression) changes. Thus, for a chord sequence of length n, the melody can have at most n-1 suspensions.

  Each potential suspension is examined and assigned a weight based on the suspension and the underlying chords. Each suspended note that is consonant over two chords is rewarded (+10) while each suspended, dissonant note is punished (-20). If there is no suspension or a suspended note that is a rest, a small reward is given (+5).

- **Prioritizing Downbeat and Half-Beat** $(db)/(hb)$

  The most significant and memorable notes in a song measure occur at the first beat of the measure and the half-beat of the measure–for a measure with four beats, the half-beat would be the third beat. While in general it can be good to have dissonance in a jazz solo, excessive dissonance at critical points of the song can result in listener confusion. It is especially important for these notes to be consonant, as they psychologically enable the listeners to orient themselves in a free-form jazz melody.

  Thus, consonant downbeats and half-beats are rewarded (+10/+5) as are rests (+10/ +5) while dissonant notes are punished heavily (-20/ -20).

- **Long Note** $(ln)$

  Generally, long notes are points of stasis in musical compositions, and thus it is preferable for them to be harmonically stable. The duration that constitutes a long note is user definedâĂŤwe define a long note as any note with a duration exceeding eight durks.

Consonant long notes are rewarded (+10) while dissonant long notes are punished heavily (-20). Additionally, long rests excessively disrupt the melody, especially in the context of a jazz solo, and thus they are also punished (-20).

- **End Song on Root** (*er*)

  Dissonant final notes generally do not offer adequate closure and diminish greatly a listener's impression of a given solo, especially if the listener is not well acquainted with jazz. Thus, jazz solos that end on the root note of the given scale are rewarded (+20) and solos that end otherwise are punished (-20).

- **Off-Pulse and Syncopation** (*pop*)

  Notes that do not fall on the eight-note pulse (i.e. every 4 durks) tend to muddy thematic development in the melody as they result in disjointed rhythms. Additionally, based on subjective preference interviews, human listeners preferred syncopated rhythms, which involve notes that fall on the weak beat of the measure as opposed to the strong beat. Thus, each off-pulse note is punished (-5) while each syncopated note is rewarded (+5).

#### 5.5.2 *Baxter-Specific Characteristics*

- **Large Interval** (*li*)

  Due to physical constraints, Baxter cannot quickly and accurately move its arms between notes that exceed 9 pitches in range based on our pitch encoding. Thus, any note jumps that exceed 9 pitches is penalized (-10).

- **Short Note Duration** (*ps*)

  Baxter can accurately play generated solos at thirty beats per minute provided that only a few notes are shorter than 4 durks. However, it is useful to keep some short notes in a melody for rhythmic variation. Thus, short notes are punished (-80) but are allowed to exist in the final melody.

### 5.6 Summary of Other Preferences and Parameters

While experimenting with the genetic algorithm, a variety of decisions were made that were not explicitly motivated by the musicality and Baxter considerations detailed in above sections. Specifically, we chose to implement hill-climbing (children cannot replace parents if it was not at least as fit in subsequent generations); a selection process for mating that involves tournament selection with a size of 4 and $p = .9$;[6] elitism, which keeps the best 25 chromosomes in each generation; a generation termination value of 800; and a mutation probability of each different mutation of $p = .2$.

## 6. HARDWARE TRANSLATION PIPELINE

The final step in the music performance process is to translate data from the genetic algorithm into actual produced sounds. Hardware is controlled via a set of python scripts running on the Robotic Operating System (ROS).

### 6.1 Selection of Hardware

ARTIST utilizes three pieces of hardware in order for the generated music to be played: a robot to perform the generated music, a musical instrument to be played, and the hardware to connect the robot with the instrument.

#### 6.1.1 *Robotic Platform*

Much thought went into selecting the proper platform to use as the robotic music performer. An initial examination of the robotic platform space led to four potential options:

- **KUKA Arms**: Industrial robotic arms designed for factory automation.
- **Baxter**: A human sized, collaborative robot designed for monotonous tasks in order to free up skilled human labor.
- **Nao**: A small, humanoid robot with the ability to walk.
- **Custom Rig**: A platform built entirely from scratch to best suit the needs of the system.

Ultimately, the decision was made to use Baxter, a system developed by Rethink Robotics. Standing at 6 feet tall with a weight of 306 pounds, the Baxter platform most resembled an actual human who would play jazz improvisation on an instrument. Furthermore, Baxter contains seven degrees of freedom in each of its two arms, allowing it to easily manipulate a large variety of musical instruments. The platform also has the ability to switch between several arm manipulators allowing for increased versatility, and it has a large screen to display information during the performance.

One of the weaknesses of Baxter lies in its lack of absolute precision when manipulating its arms. This can prove problematic when trying to perform on certain types of musical instruments. Other robotic platforms that were considered do not have this limitation, specifically the KUKA Arms and a custom rig. The KUKA platform was ultimately not chosen due to its sheer strength and power and its associated danger around humans. A custom rig was not constructed due to both time constraints on the research and lack of realism to an actual performer. In addition, the Nao platform demonstrated an inability to perform musical instruments due to its size.

A final weakness of the Baxter platform is the speed in which its arms can move between positions. Fast song tempos are a common occurrence in jazz music. This is a known limitation in the initial version of ARTIST, and a cap is imposed on the maximum tempo that songs that can be played.

Nevertheless, initial testing of the Baxter platform for ARTIST's use case yielded positive results that allowed it to be used for this initial version.

#### 6.1.2 *Musical Instrument*

The selection process for choosing a musical instrument started with looking at the landscape of instruments typically used in jazz music. A typical jazz big band consists of horns (saxophone, trumpet, and trombone), a guitar, and a rhythm section (piano, bass, drums, xylophone/vibraphone). Given that horns require constant streams of air to be played and advanced embouchure techniques, these were dismissed from the selection process. The guitar, drums, and bass were eventually dismissed as well due to the complex techniques needed to perform at a reasonable level.

Ultimately, the decision was made to use a xylophone as the musical instrument. It's size, cost, and quality made it a more favorable choice over the larger piano and vibraphone. Finally, there was a low barrier of entry in getting the instrument to produce sounds via a robotic platform.

#### 6.1.3 *Connecting the Robot and Instrument*

The final part of the hardware selection phase was finding an appropriate method for allowing Baxter to play the xylophone. Typically, a human performer uses mallets to strike keys on the instrument. Typical mallets were unusable for several reasons. First, the angle at which a performer typically strikes the xylophone was not achievable by Baxter. Furthermore, Baxter does not have the ability to move his arms with the same force and speed as its human counterpart. Finally, Baxter's gripper attachments are not friendly towards holding a typical mallet – while one arm has a standard gripper, the other arm is a suction pump with no grasping ability.

A custom set of mallets were designed for Baxter to perform. A spring was placed between the head of the mallet (the part that actually strikes a key) and the rest of the stick. The spring allowed for the keys to be hit at a slower speed and force than a human performing the same instrument while maintaining the quality of sound produced. Custom attachments were built to allow Baxter to hold the mallets without them coming out of place.

### 6.2 Generalizable Model

One of the main advantages of ARTIST is that it's not limited to a specific model xylophone. Rather, the system was constructed to easily and quickly adapt to a wide variety of instruments. This mimics the versatility of a real-world jazz setting, where a player needs to be able to easily switch between different xylophones while performing. The hardware pipeline allows for the quick modification of the performance algorithm based on the instrument it is currently using.

As ARTIST finds notes by seeking a given arm position using the Baxter inverse kinematics module, it's easy to switch between instruments.

#### 6.2.1 *Baxter Training Software*

At the start of a performance, a user can begin the training pipeline, which teaches the robot where to move its arms in order to hit each playable key on the instrument. One of the many features of the Baxter robotics platform is that each arm can easily be repositioned by an operator by squeezing the arm cuff and freely moving each of the seven joints to the desired position. The training program allows for the operator to select a given note using a dial on Baxter based on scientific pitch notation, manipulate the arm until it is in the correct position, and then use a button to set the joint positions for later playback.

#### 6.2.2 *Range of Keys*

Using the training program, ARTIST is only limited in the number of keys and octaves it can play by the maximum wingspan of Baxter. Given the width of a standard xylophone key, ARTIST has the ability to play the entire seven and a half octaves seen on a traditional 88-key piano within its wingspan.

#### 6.2.3 *Variable Height Instrument*

A typical xylophone has elevated keys for sharps and flats (the black keys on a piano). The training software accounts for the potential of variable height keys on different xylophones by storing the positions of each individual key and using inverse kinematics to seek them when needed.

### 6.3 Mallet Transformation Generation

After receiving the Baxter track from the genetic algorithm, a pipeline generates a set of $(pitch, start_time)$ tuples. $start_time$ dictates a relative time from the start of the performance that the note should be played, based on the notes duration and position in the generated music.

These tuples are then used to generate mallet transformations, which dictate which arm should be used to play a given note and where that arm must move to in order to make the sound play. Finally, these transformations are sent via a multithreaded, asynchronous process to queues for each arm and await playback. Separating the queues between arms allows for multiple notes to be played at once.

### 6.4 Performance Behavior

As mentioned, one of the limitations of the ARTIST system is in the speed at which notes can be played. As a result, ARTIST constantly monitors its performance to make sure that a single misplayed note (either due to an incorrect duration or pitch) does not act as a single point of failure. The system can discard notes that have missed their play time so that the performance can recover gracefully, much like a real jazz musician.

## 7. EVALUATION

### 7.1 Setup

A survey was created to test the quality and "human-ness" of the robotic compositions in comparison with a professionally composed jazz classic ("Now's the Time" by Charlie Parker). To control for playback quality, short clips of Baxter playing each of the composed pieces were recorded at a tempo of 30 beats per minute. In the survey, respondents were asked to listen to two computer generated pieces and Parker's classic in a random order and then asked to (1) group the songs based on whether they thought it was human generated or computer generated, (2) rank the songs by preference, and (3) provide qualitative feedback on each of the songs.

To prevent unwanted selection bias, the order in which the songs were presented was randomized, and each song was named with a primary color:

- **BLUE**: "Now's the Time" by Charlie Parker
- **RED**: A computer-generated piece
- **GREEN**: A computer-generated piece

### 7.2 Demographics

The survey was taken by $n = 41$ participants and asked for the demographic information presented in Tables 2 and 3.

Of the $n = 41$ participants, 10 of them had previously heard one of ARTIST's presentations in CPSC 473 and were therefore familiar with the algorithms used.

| | Average Years | Standard Deviation |
|---|---|---|
| Age | 25.05 | 12.22 |
| Played musical instrument | 9.44 | 8.05 |
| Studied music theory | 3.15 | 3.34 |

**Table 2: For how many years have you either played a musical instrument, or studied music theory?**

| | Rating (1-5) | Standard Deviation |
|---|---|---|
| Experience | 2.12 | 1.00 |

**Table 3: How experienced are you with the jazz genre of music? (1 = no experience, 5 = a lot)**

## 7.3 Quantitative Survey Results

When asked to rank the songs by preference, participants ranked Charlie Parker's Blue song the highest, with an average ranking of 1.6829. That being said, the computer generated songs were close behind, with average rankings of 2.0732 and 2.2440.

In Table 4, the average rank and standard deviation for each of the songs is presented. The same results appear graphically in Figure 1.

| | Average Rank | Standard Deviation |
|---|---|---|
| BLUE | 1.6829 | 0.8197 |
| RED | 2.0732 | 0.8482 |
| GREEN | 2.2440 | 0.6993 |

**Table 4: Rank by Song**

The results of the song categorization show similar results. The majority of participants (22 vs. 19) determined that Charlie Parker's Blue song was professionally composed, whereas the majority of participants determined the Red and Green songs to be computer generated.

In Table 5, the categorization for each of the songs is presented. The same results appear graphically in Figure 2.

As expected, the majority of participants ranked Charlie Parker's "Now's the Time" as their top choice and 22/41 thought that it was professionally composed. That being said, the results are closer than would be expected for a jazz improvisation system trained on freely available online MIDI data. For example, 16/41 participants mistook the computer-generated Red song for a professionally composed piece of music and 13/41 similarly mistook the Green song.

## 7.4 Qualitative Survey Responses

Respondents were asked to qualitatively describe and evaluate each song that they heard. In general, the diversity of opinion found in the responses to each song suggested that (1) many people could not accurately decipher which songs were professionally composed vs. computer generated, (2) participants described musical features–often in great detail– in computer generated compositions as if there were composed by a human, and (3) people often preferred, at a non-negligible rate, computer generated compositions to the professionally generated track.

Below, some representative sample comments from each of the songs are presented.

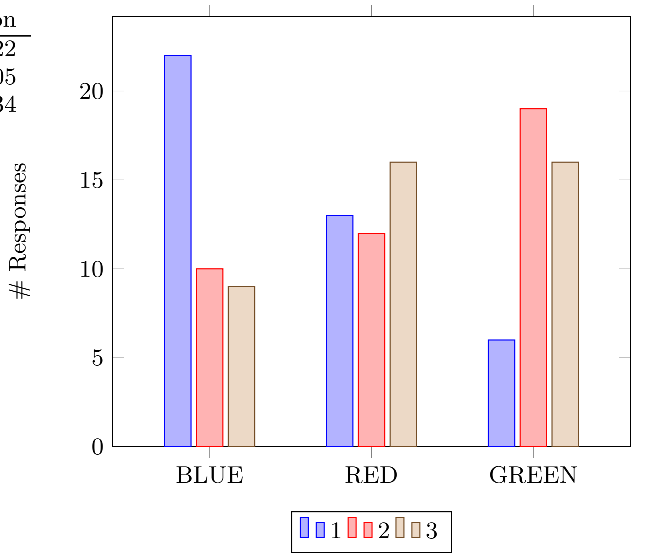

**Figure 1: Rank by Song**

| | "Computer Generated" | "Professional Composer" |
|---|---|---|
| BLUE | 19 | 22 |
| RED | 25 | 16 |
| GREEN | 28 | 13 |

**Table 5: Categorization by Song**

### 7.4.1 *Red Song (Computer Generated)*

Participants generally found the Red song more musically appealing and considered it more likely to be human generated than the Green song. Many participants rated the Red song as their favorite tune, and some even thought that they had "heard" the song before, despite the fact that it was a generated improvisation. In particular, people tended to like the finale of the Red song.

- "Sounded more disjointed, but jazzy."
- "Fun and jumpy. It seemed the most improvised."
- "It sounded like something I've heard."
- "This could be the ending of a song. The last note is harmonic, ending on a major chord note."
- "Red song is the most lyrical and harmonically interesting of the three pieces, containing larger arpeggiated motions outlining an appealing tonal/functional harmony."
- "Red song was probably my favorite."
- "Simple and uninteresting at first, but had a surprising twist in the middle."

### 7.4.2 *Green Song (Computer Generated)*

The Green song was generally the least liked out of all the songs and the most likely to be rated as computer generated. Nonetheless, there were still respondents that praised the musicality of the composition.

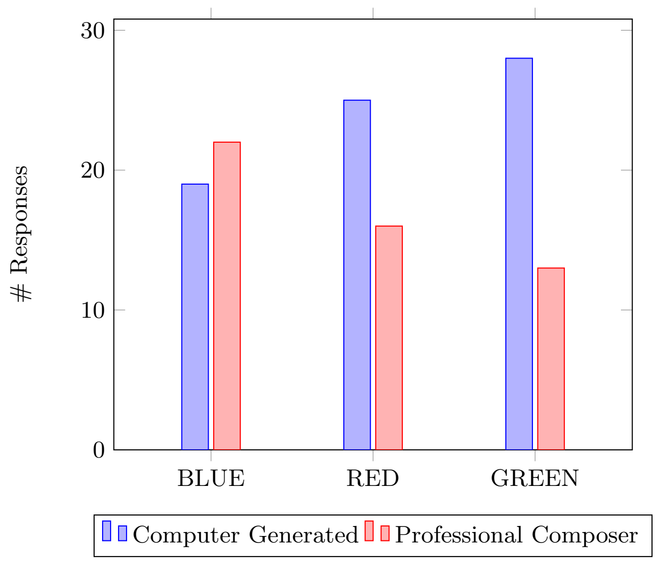

**Figure 2: Categorization by Song**

- "This song seems to go somewhere. Professionally written."
- "Has a pseudo-melody, feels pretty real."
- "Two separate musical phrases - the first was harmonic. The second phrase was more dissonant to listen to."
- "It's clear that the robot was playing something that had some complete form, although the starting F's in the C major key were a bit unclear in terms of their harmonic function and didn't quite fit into the rest of the tune."

#### 7.4.3 *Blue Song ("Now's the Time" by Charlie Parker)*

Charlie Parker's 1945 classic generally garnered higher praise from listeners and was the most likely to be rated as professionally composed. However, this sentiment was far from unanimous. Highlighted below are some of the dissenting comments.

- "Blue song is not as harmonically interesting as Red, but its use of repetition and the phrase rule of "short-short-long" creates an aurally attractive motif, despite no real harmonic motion."
- "Didn't sound great. I feel like the notes may have been randomly chosen on this one."
- "Remarkably monotonous. It was pretty bad."
- "Very repetitive and not as much of a sense of greater form and structure."

## 8. CONCLUSION AND FUTURE WORK

This paper describes a novel approach for the generation of jazz improvisation by combining a statistical trigram model with a genetic algorithm. In the first stage of the pipeline, unlabelled MIDI data is scraped and parsed into a PostgreSQL database. Given the unlabelled music data, the harmonic analysis program uses a dynamic programming implementation of a preference rule system to label the chord progressions. Once the chords have been labelled, the trigram algorithm trains a separate model for each chord. Given the trained models, the genetic algorithm takes the generated trigram output and optimizes based on the presented set of features.

Using a generalizable model for training and performing the xylophone with the Baxter robotics platform, the paper concludes with a discussion of promising results for the future of real-time jazz improvisation.

Currently, the generation pipeline takes the chord progression and desired output key as input to the algorithm. Rather than taking these as input, future implementations of an ARTIST-inspired jazz improvisation system could (1) generate their own unique chord progressions or (2) extract the chord progression from the underlying music. In particular, it would be interesting to build a system that can extract chord progressions in real-time from a live musical performance and thus allow ARTIST to really act like a live musician.

Analyzing the way people see robotic performance versus a live performance would be an interesting next step to investigate. We speculate that future research could produce a robot with the ability to play live music and eventually serve as an appealing form of entertainment.

## 9. ACKNOWLEDGEMENTS

A special thanks to Aditi Ramachandran and Professor Brian Scassellati of the Yale University Computer Science department for advising us on this project.

## 10. REFERENCES

[1] ANDREWS, R. Darpa is building jazz-playing robots, October 2015. [Online; posted 24-October-2015].

[2] BILES, J., A. P., AND LOGGI, L. Neural network fitness functions for a musical iga. *Technical report, Rochester Institute of Technology* (1996).

[3] G., P., AND G., W. A genetic algorithm for the generation of jazz melodies.

[4] GOLDBERG, D. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[5] HORNER, A., AND GOLDBERG, D. Genetic algorithms and computer-assisted composition. *In Proceedings of the Fourth International Conference on Genetic Algorithms* (1991).

[6] MILLER, AND GOLDBERG. Genetic algorithms, tournament selection, and the effects of noise.

[7] TEMPERLEY, D. *The Cognition of Basic Musical Structures.* MIT Press, 2004.