Straight-Ahead Jazz with GenJam: A Quick Demonstration

John A. Biles
School of Interactive Games and Media
Rochester Institute of Technology
http://igm.rit.edu/~jabics/
jabics@rit.edu

Abstract

The author's GenJam project has spanned nearly 20 years of development and performance. This demonstration provides a brief overview of GenJam's current configuration illustrated with a visually annotated performance of a tune. After a brief explanation of GenJam's real-time interactive capabilities, we'll play more tunes, as time permits.

Introduction

GenJam, the Genetic Jammer, has been an ongoing project for nearly 20 years and has become my favorite jazz gig. GenJam is a real-time, MIDI-based, interactive improvisation system that uses evolutionary computation to evolve populations of melodic ideas (*licks* in the jazz vernacular), which it uses to generate its improvisations in live performance settings (Biles 2007, 1994). With a current repertoire of well over 300 tunes in a wide variety of straight-up jazz and Latin styles, we perform a couple of times a month on the average in mainstream venues, which makes GenJam a true, working musician.

The goal and focus of the GenJam project have evolved over the years from a proof of concept to demonstrate that a genetic algorithm could improvise, into a musical agent that plays a pivotal role in most of my live jazz gigs. This shift in focus from the technology to the music has shaped my conception of the use of technology in musical performance, which is the subject of my position paper at this conference (Biles 2013). This demonstration is intended to briefly describe GenJam's capabilities and architecture.

GenJam's Capabilities

GenJam can improvise full-chorus solos, trade fours, eights, 12s or 16s with its human sideman, and play an intelligent echo of what its human sideman plays, delayed by anywhere from one beat to four measures (usually a delay of one measure). In all cases, the improvisations it plays incorporate what it has heard its human sideman play earlier in the tune, so the level of interactivity is very high and fairly sophisticated.

GenJam can play in 3/4, 4/4, 5/4, 7/4, 12/8 or 16/8 time, and it can perform with either swing or even eighth-note quantization. It has a rather extensive harmonic knowledge and knows how to handle 18 different kinds of chords as it maps its lick chromosomes to actual pitches in real time.

GenJam's Architecture

Figure 1 shows GenJam's high-level architecture as it performs a tune with a human sideman.

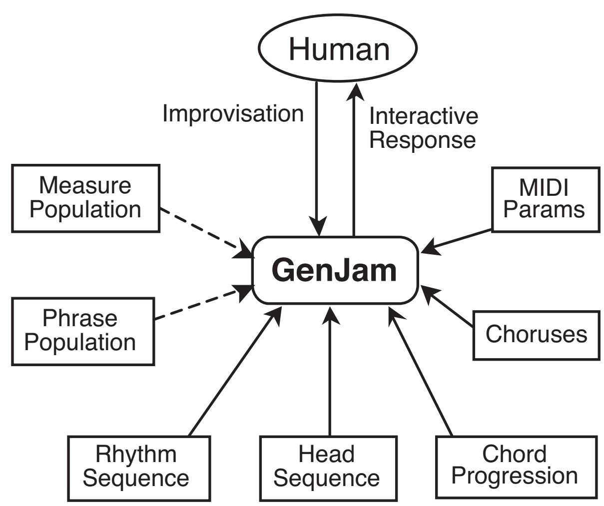

Figure 1. GenJam's Architecture in Performance

GenJam reads several files before performing the tune. The Chord Progression file sets the tempo for the tune, indicates whether to use swing or even eighth notes, and provides the chord progression for the tune with up to two chords per measure. The Choruses file indicates what GenJam should do for each succeeding chorus of the song form defined by the chord progression. In addition to the improvisation options listed in the capabilities section above, GenJam can rest while the human solos for an entire chorus or while the head for the tune is played. The MIDI Params file configures the tone generator to define what instruments will be used for the tune, where the different voices will land in the stereo field, how loud they will be, and some 30 other parameters that can be tweaked.

Two MIDI sequences are loaded as well. The Rhythm Sequence contains MIDI tracks for the rhythm section (typically piano, bass and drums, but possibly adding guitar and strings). I shamelessly use Band In A Box (Gannon 2013) to generate at least some of the rhythm section tracks so that I can be sure of a competent rhythm section. The optional Head Sequence is one or more tracks of harmony parts that are played on the head of the tune. This reflects the typical jazz arrangement of a written-out melody with possible harmony part(s) on the first and last choruses of the tune (called the head in jazz vernacular), with improvised choruses in between. As already noted, GenJam has a wide variety of improvisational modes to draw from.

The Measure and Phrase Population files denote the two populations of melodic material with which GenJam begins a tune. The dashed lines indicate that GenJam builds these populations in different ways for different versions of the program. The original version of GenJam (Biles 1994) uses an interactive genetic algorithm (IGA), whose fitness is derived from feedback supplied by a human mentor and is used to drive a generational learning regimen. A subsequent autonomous version of GenJam (Biles 2001) evolves these populations from databases of licks without using fitness. The chromosome representation and how it maps to actual notes is critical to understanding how GenJam works, so a brief illustration follows.

GenJam's Chromosome Representation

GenJam co-evolves two hierarchically interrelated populations of licks, which it uses to generate its improvisational content. In the Measure Population, individual chromosomes map to one measure of notes in performance. Individuals in the Phrase Population represent four-measure phrases made up of individuals from the Measure Population. Figure 2 shows an example phrase individual and the three measure individuals it references in the Measure Population (one measure is used twice in the phrase).

Our example is phrase 23 (out of 48 in the entire Phrase Population), and it is made up of measure 57, measure 57 again, measure 11, and measure 38. The -12 for phrase 23 is its *fitness*, which is used only by the IGA version of GenJam and indicates how well-regarded this measure is, based on the results of a training regimen, with 0 being neutral.

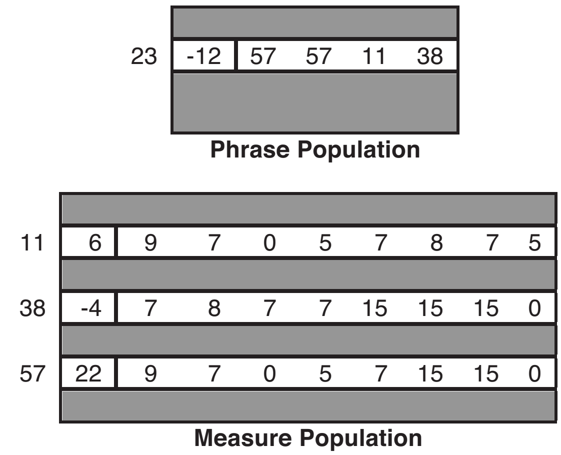

Figure 2. Chromosome Representation (Biles 1994).

Measures 11, 38 and 57 have chromosomes with eight genes, each of which represents an eighth note in a 4/4 measure. In 3/4 time there would be only six eighth-note genes in a measure chromosome, and in 7/4 time there would be 14. Each of these individuals also has a fitness value (22 for measure 57). There are a total of 64 individuals in the measure population, which requires six bits for the measure reference genes in the phrase chromosome.

The range of values for the eight genes in a measure is 0-15 (four bits). Two of the possible values provide rhythmic information, and the remaining 14 provide tonal information. A 0 value for a note gene represents a *rest event* and is performed by generating a MIDI note-off event. A 15 represents a *hold event* and is performed by doing nothing, i.e., holding the event from the previous gene through this eighth-note window. The remaining values (1-14) represent *new-note events* and are performed by generating a MIDI note-off event immediately followed by a MIDI note-on event. The MIDI pitch of the note-on is determined by using the gene value as an index into a table of actual MIDI pitches in a 14-note scale (roughly two octaves) derived from the chord being played by the rhythm section at that point in the piece. As indicated earlier, GenJam knows 18 different kinds of chords, so the result is that GenJam cannot play a theoretically wrong note – unlike the author!
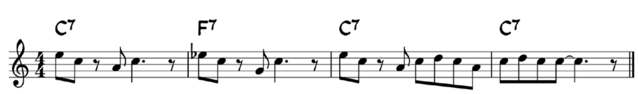
Figure 3. Phrase played from Figure 2.

Figure 3 shows the example from Figure 2 mapped to actual notes using the first four measures of a typical C blues progression. Note that even though measure 57 was repeated in the first two measures of the phrase, the actual notes played were different in the two cases because the scale for a C7 chord is different from the scale for an F7 chord.

Figure 4 shows GenJam's chord-scale mappings with examples using a chord root of C.

Chord	Scale	Notes
Cmaj7	Major (avoid 4th)	C D E G A B
C7	Mixolydian (avoid 4th)	C D E G A Bb
Cm7	Minor (avoid 6th)	C D Eb F G Bb
Cm7b5	Locrian (avoid 2nd)	C Eb F Gb Ab Bb
Cdim	W/H Diminished	C D Eb F Gb G# A B
C+	Lydian Augmented	C D E F# G# A B
C7#5	Whole Tone	C D E F# G# Bb
C7#11	Lydian Dominant	C D E F# G A Bb
C7alt	Altered Scale	C Db D# E Gb G# Bb
C7#9	Mix. #2 (avoid 4th)	C Eb E G A Bb
C7b9	Harm Minor V (no 6th)	C Db E F G Bb
CmMaj7	Harmonic Minor	C D Eb F G Ab B
Cm6	Dorian (avoid 7th)	C D Eb F G A
Cm7b9	Melodic Minor II	C Db Eb F G A Bb
Cmaj7#11	Lydian	C D E F# G A B
C7sus	Mixolydian	C D E F G A Bb
Cmaj7sus	Major	C D E F G A B
C7Bl	Blues	C Eb F Gb G Bb

Figure 4. Chord/Scale Mappings.

GenJam does a very simple vertical harmonic analysis (Russell 1959). For example, a minor seventh chord will always map to a hexatonic minor scale that avoids the sixth, because the chord might function as a tonic minor, which would indicate a flatted sixth (Ab with a chord root of C), or it could be a dorian minor in a II-V-I sequence, which would indicate a natural sixth (A with a chord root of C), or it could be a minor seventh chord that happens to sound good at that point in the harmonic sequence. The point is that without doing a functional harmonic analysis of the chord progression, which would often lead to a misinterpretation or at best an ambiguity, especially in jazz, the choice of which sixth to use is problematic, so GenJam solves the ambiguity by avoiding any sixth in the scale for a minor seventh chord. This concept of avoid notes is well established in jazz pedagogy (Haerle 1980), and while the conventional wisdom currently seems to be in the direction of avoiding avoid notes and instead using them as passing tones when harmonically appropriate, GenJam simply avoids them in order to be safe.

This reflects my underlying design philosophy that starts with simple, robust choices and tries to avoid complex solutions to specific situations. I want GenJam to always sound competent and never sound "wrong". I'll sacrifice the possibility of an occasionally brilliant moment that demonstrates sophisticated handling of a specific scenario in order to avoid the more frequent embarrassing moments when that scenario is misidentified and GenJam sounds incompetent in a straight-ahead context. One way to look at it is that at the meta-level, GenJam is very simple because when I have experimented with making it more sophisticated, the resulting music it played was worse, not better. For me, it's about the music, not the underlying algorithms.

Interactivity

When GenJam performs with its human sideman, it listens to the human using a pitch-to-MIDI converter and builds chromosomes using the same representation described earlier, which it in turn uses to evolve its improvisations. There are three modes of interactivity, and on a typical tune all three are used.

When GenJam trades fours with the human, it builds four fresh measure chromosomes and a phrase chromosome as the human plays his/her four measures. In the last instant of the human's four, GenJam stops listening and applies mutation operators to the phrase and four measure chromosomes, and it then immediately plays back the resulting phrase as its response in the next four bars of the tune. The mutation operators are musically meaningful and include compositional devices like retrograde, inversion, transposition, sequencing, hemiola, sorting, and a host of other melodic manipulations. In this way, GenJam uses its intelligent mutations to develop the human's four into its response.

The Roland GI-10 pitch-to-MIDI converter that I use (Roland 2013) makes frequent mistakes, usually triggering more notes than the human played and messing up the pitches and note onset times, but because it is mapping to the chromosome structure, which will be mutated anyway and played back using scales over the next four bars of the tune, the errors are irrelevant and can even be interpreted as "development". It's always nice to turn a bug into a feature! Actually, I have tried several other off-the-shelf pitch trackers, including the recent Sonuus G2M-V2 (Sonuus 2013), but they generate less MIDI note-on/note-off traffic. I'd rather have the pitch tracker generate false notes than miss notes that I play.

When GenJam does an intelligent echo of the human's improvisation, it maintains a buffer into which it places its guesses of the notes the human is playing and plays that buffer back after a delay as it continues to listen. The delay can be set to as low as a single eighth note, or as long as four measures. I typically use a delay of one measure and sometimes a delay of half a measure (the latter is especially effective in 7/4 time, where a delay of 7 eighth notes falls within a beat).

Because the delayed chromosomes are played back over the current chords in the tune, the result is a loop delay effect that follows the chord changes, something that currently available effects boxes cannot do. The result from a performance standpoint is a unique experience, which I discuss more fully in my position paper for this workshop (Biles 2013).

The third mode of interactivity involves evolving the Measure Population in the direction of the human's playing. When the human plays the head and/or takes a full-chorus solo, GenJam listens and builds a measure chromosome for each measure of the human's chorus. After a measure chromosome has been built, GenJam searches through the entire measure population of 64 individuals and selects one whose first and last new-note events are as close as possible to the first and last new-note events of the human's measure. It then performs an intelligent crossover between the two chromosomes, choosing a crossover point such that the resulting horizontal interval is minimal to insure a smooth-sounding measure. It performs the crossover at that point and selects the child that contains more of the human's content. Finally, it replaces the original measure from the population with this child, which now has a fragment of the human's measure. The result after the head and a human solo chorus is a subtle migration of GenJam's licks in the direction of the human's performance.

Choosing a parent measure individual whose first and last new-note events are close to the first and last new-note events of the human's measure insures that the resulting child will integrate smoothly in the phrase individuals that use the now-evolved measure. Just as the crossover operator chooses its crossover point within a measure to insure a smooth-sounding measure individual, this matching of measure endpoints assures that the evolved measure will sound smooth in the phrases that use it. Again, the design philosophy tilts toward simple and robust.

Implementation

GenJam is implemented using Roger Dannenberg's venerable Carnegie Mellon MIDI Toolkit (CMT). The good news is that I've had no technical issues over the years implementing the many enhancements I've made. The bad news is that the CMT is considered obsolete (Dannenberg, 2013), which makes GenJam, alas, a legacy system. Currently, I am working to launch an open source community to bring GenJam into the current millennium. This work is being initiated and coordinated by RIT's Center for Media, Arts, Games, Interaction and Creativity (MAGIC 2013).

While GenJam's underlying code base is not overwhelming (about 10,000 lines of C code on top of the CMT), successfully productizing GenJam is not trivial. We want to go beyond a simple port of the existing code base and address idiosyncrasies that arise from a custom console application used by a user community of cardinality one. These include general UI issues for various user communities, staying with MIDI versus moving to real-time audio, integrating background and harmony tracks for standard tunes while avoid intellectual property issues, integrating modern sample libraries to replace standalone tone generators, and deciding on a forward-looking pitch-tracking strategy. For example, we are currently mulling over a port to the latest X-Box or PS platform and using the formidable GPU capabilities on these platforms to execute pitch-tracking that is currently handled by the Roland GI-10. We also intend to integrate a version of an existing stand-alone visualizer that displays GenJam's and the human's notes as different-colored balls that spawn on the right edge of a window (vertical axis mapping to pitch) and scroll across the screen from right to left.

The goal of all this is to make GenJam available to a wide user base, particularly in the jazz education community. However, I will continue to enjoy performing with GenJam and putting together more tunes!

References

Biles, J. A. 2013. Performing with Technology: Lessons Learned from the GenJam Project. MUME 2013 Workshop.

Biles, A. 2012. GenJam: the Genetic Jammer. Presented at TEDx Binghamton University, Binghamton, NY, March 11, 2012. http://www.youtube.com/watch?v=rFBhwQUZGxg&list=PL89268920295951CC&index=5&feature=plpp_v

Biles, J. A. 2007. Evolutionary Improvisation: GenJam. In *Evolutionary Computer Music*, ed. E. R. Miranda and J. A. Biles, 137-169. London: Springer-Verlag.

Biles, J. A. 2001. Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness. In Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program, San Francisco.

Biles, J. A. 1994. GenJam: A Genetic Algorithm for Generating Jazz Solos. In Proceedings of the 1994 International Computer Music Conference, 131-137. San Francisco: International Computer Music Association.

Dannenberg, R. 2013. CMU MIDI Toolkit Website. http://www.cs.cmu.edu/~music/cmt/

Haerle, D. 1989. *The Jazz Language*. Miami, Studio 224.

Gannon, P. 2013. Band-in-a-Box. P G Music Website. http://www.pgmusic.com/

MAGIC. 2013. RIT Center for Media, Arts, Games, Interaction and Creativity Website. http://magic.rit.edu/

Roland USA. 2013. Roland GI-10 User Manual. Roland USA Website. http://media.rolandus.com/manuals/GI-10_OM.pdf

Russell, G. 1959. *The Lydian Chromatic Concept of Tonal Organization*. New York: Concept Publishing Co.

Sonuus Limited. 2013. Sonuus G2M. Sonuus Website. http://www.sonuus.com/products_g2m.html