GenJam: Evolutionary Computation Gets a Gig

John A. Biles,
Information Technology Department, RIT

Abstract

GenJam (short for Genetic Jammer) is an evolutionary computation-based, real-time interactive jazz improvisation agent. GenJam improvises spontaneous autonomous solos and performs interactive and collective improvisation with a human performer by listening to what the human improvises, mapping what it heard to its internal chromosome representation, and using intelligent mutation and crossover operators to develop what the human plays into what it plays in response.

After an overview of GenJam's architecture in performance settings, this paper describes GenJam's chromosome structure for representing melodic material, and explains how it interacts in real time with a human performer. Where GenJam gets its musical ideas is discussed next, followed by HCI aspects from both the audience's and the performer's perspectives. Finally, a discussion of GenJam as an IT application and a brief prediction of its future conclude the paper.

Introduction

Evolutionary Computation (EC) has emerged as a powerful computational paradigm for a wide range of applications[1]. EC applies the principles of natural evolution and genetics to artificial search problems by evolving a population of potential solutions, where each individual solution in the population is represented by a chromosome-like structure. The fitter individuals in a population are selected to breed new solutions using crossover and mutation, and over succeeding generations of the population acceptable solutions emerge. The evolutionary paradigm has been applied to creative domains in areas as diverse as visual art, architecture, industrial design and music[2].

GenJam[3] applies evolutionary computation to jazz improvisation by evolving populations of melodic fragments, or licks in the jazz vernacular, from which it constructs improvisations in real time. GenJam's repertoire currently exceeds 200 tunes in a variety of jazz, Latin and new age styles, and performs regularly as a featured sideman in the author's Virtual Quintet. Attendees to this conference heard the Virtual Quintet perform at the conference reception Thursday evening. This paper describes how GenJam works and, in the context of an IT conference, discusses HCI issues and the IT philosophy of the GenJam project.

GenJam in Performance

A single execution of GenJam results in the performance of a single tune. Figure 1 shows GenJam's system architecture when performing a tune. GenJam reads several files that describe the tune and its arrangement (the rectangles with solid arrows), either reads or builds two

interrelated populations of melodic ideas (the rectangles connected with dashed arrows), and, during the performance, interacts with a human performer in a variety of ways (the Performer oval).

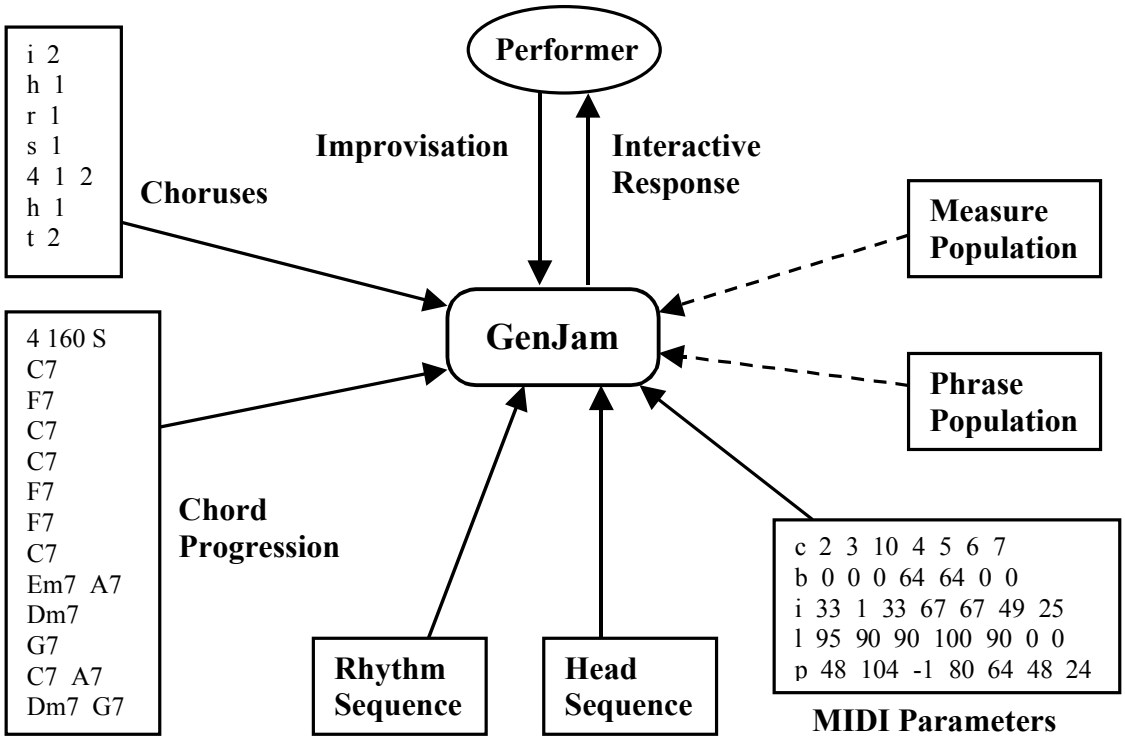

Figure 1. System architecture of GenJam in performance

The Chord Progression text file describes the tune to be performed, including the tempo (160 beats per minute in Figure 1), whether to use swing or even eighth notes (the 'S' in the first line), the octave range in which GenJam should play (the 4 in the first line indicates a tenor sax range), and the chord progression of the tune (one measure per line, up to two chords per measure, beginning with the second line).

The Choruses file describes what GenJam should do for each cycle of the form described by the chord progression. The musical structure for most jazz performances is theme and variation, where the soloists perform the original melody in the first and last choruses and improvise in the middle choruses. The improvisations can be full-chorus solos by one soloist, "trading fours" or eights, where two soloists take turns every four or eight bars, or collective improvisation, where soloists improvise simultaneously. Spontaneous interactivity between soloists is a key ingredient to a successful jazz performance, and GenJam accomplishes this, as indicated by the interaction arrows with the Performer oval in Figure 1.

The Rhythm and Head Sequence files are standard MIDI files that provide the rhythm section accompaniment and arranged harmony parts for the first and last choruses (the "head" in jazz

parlance). These are canned sequences that will be identical for each performance of the same tune. The MIDI Parameters file configures the tone generator by indicating which instruments to use for each part, how loud each should play, where in the stereo field they should be, and up to 30 other parameters.

The Measure and Phrase Populations are data structures that represent the melodic ideas from which GenJam can construct improvisations. These populations are either read directly from text files or constructed internally from a licks database, as will be described in the Woodshed section. However, I will describe the representation of a four-measure phrase (which I refer to as GenJam Normal Form or GJNF) now because it is central to understanding how GenJam improvises.

GenJam Normal Form

Figure 2 illustrates a cooked-up example phrase embedded in the measure and phrase populations. Our example phrase happens to be phrase number 23 (out of 48). It has a fitness of –12, which means that it is regarded as a "bad" phrase (more on this in the Woodshed section). The four numbers in measure 23's chromosome are pointers into the measure population, which means that phrase 23 consists of measure 57, followed by measure 57 again, followed by measure 11, followed by measure 38.

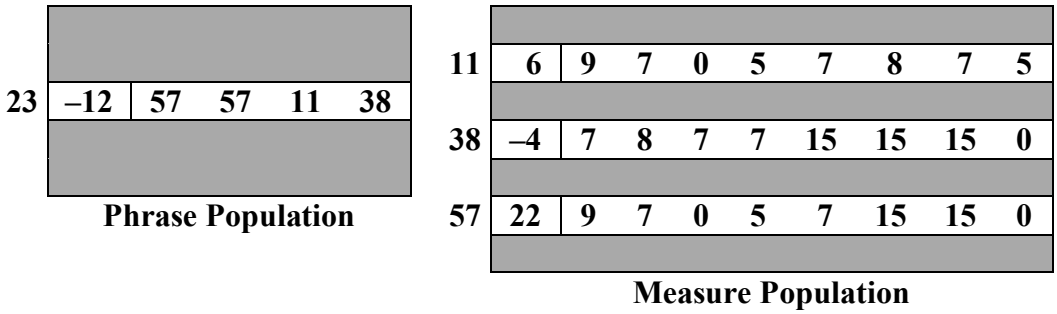

Figure 2. Example phrase[3]

Those three measures are shown in the Measure Population in Figure 2. The measure population contains 64 individuals, each representing a sequence of eighth-note-length events. In 4/4 time that works out to eight events per measure. Different versions of GenJam play in 3/4, 5/4, 12/8, 7/4, and 8/8 time, which lead to measure chromosomes of 6, 10, 12, 14 and 16 events, respectively. In our example, measure 57 has a fitness of 22, which means that it has been regarded as a "good" measure. That a good measure can occur in a bad phrase is due to the fact that a given measure can occur in more than one phrase. Since there are 48 phrases, each consisting of four measures, a given measure would be expected to occur three times, on average, in the phrase population.

The events in a measure chromosome map to actual notes in real time during a performance. There are three types of events. A *rest* event (represented by a 0) maps to a MIDI note-off event when it is performed, which terminates any note begun earlier. A *hold* event (represented by a 15) maps to nothing, which results in the note or rest in the previous event being held through

this event. A *new-note* event (represented by 1-14) maps to a MIDI note-off followed immediately by a MIDI note-on, where the note's pitch is determined by using the event number as an offset into roughly two octaves of the scale suggested by the current chord from the chord progression. GenJam understands 18 different chord types, and the result is that GenJam cannot play a theoretically wrong note.

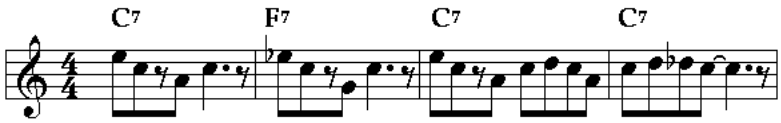

Figure 3. Phrase from Figure 2 played against first four bars of tune from Figure 1.[3]

Figure 3 shows the phrase in Figure 2 played against the first four chords of the progression in Figure 1. Notice that even though measure 57 is repeated in the first two measures, the resulting notes are different because the chords (and therefore the resulting scales) are different. Notice also that in the fourth measure, the repeated 7's in chromosome positions 3 and 4 of measure 38 resulted in a Db for position 3, which is a chromatic passing tone. This is the result of a heuristic to provide some chromatic color. When GenJam performs a full-chorus solo, then, it randomly selects enough phrase individuals to fill up a chorus of the tune and simply plays them as just described.

Interactivity

As mentioned above, spontaneous interaction between performers is a key ingredient of jazz. Since the rhythm section is canned, there is no two-way interaction between it and the soloists, but there is interaction between GenJam and a human soloist because GenJam can hear what the human plays and use what it hears in its improvisations. This interactivity occurs in all of GenJam's improvisational modes, even full-chorus solos.

Trading fours is the easiest interactive mode to describe. When GenJam trades fours with a human soloist, it listens to the human's last four measures using a Roland GI-10 pitch-to-MIDI converter, maps what it thought it heard to GJNF, mutates some or all of the chromosomes in the last instant of the human's four, and plays back the mutated phrase as its response in the next four measures. Since the human's four is mapped to GJNF, it doesn't matter if the pitch tracker (or the soloist) make errors because GenJam will map whatever it ends up with in the chromosomes to reasonable notes in the context of the chord progression. Indeed, the human can play random notes that may sound terrible, and GenJam will respond with a competent four.

A key to trading fours is for each soloist to develop what the other soloist played in the preceding four, not just parrot what the other soloist played. GenJam does this by using a collection of musically meaningful mutations. These mutations are really melodic development techniques and not the usual flipping of the occasional bit. GenJam's measure mutations include playing the measure backward, playing it upside down, playing it backward and upside down, transposing it up or down a random amount, and sorting the new-note events to create an ascending or descending melodic line. Phrase mutations include playing the measures in reverse order, rotating the measures, and repeating a measure. The key to these mutations is that they "do no

harm." In other words, given a good human four, the mutations will result in a good four from GenJam. In this respect, they can be considered intelligent mutations.

GenJam also has three collective improvisation modes in which it and the human improvise simultaneously. In one such mode GenJam listens to the human's current four as it plays the human's last four. This resembles trading fours except that GenJam plays every four, not every other four, and GenJam does not mutate what it hears the human play. Eliminating mutation is necessary because in collective improvisation, the players must complement each other in real time, which means that the human soloist must remember his last four and play off of it in his next four. If the human's last four were mutated, the result would be a "moving target" for the human soloist.

The other two collective improvisation modes are an "echo" mode, where GenJam plays the human's last measure as its next measure, and a "delay" mode, where GenJam sets up a delay line of a selectable number of events and plays back the human's improvisation, delayed by that number of events. Again, no mutations are used, and in this case only one measure chromosome is needed to store the human's improvisation. The collective improvisation modes have become a favorite challenge for the performer, as I'll describe in the HCI section.

The final piece of interactivity occurs in full-chorus solos. When the human takes a full-chorus solo, GenJam listens to each measure of the human's solo, maps it to a measure chromosome, and then does an intelligent selection and crossover with an individual in the measure population. The mate for the human's measure is selected by choosing an individual from the measure population whose first and last new-note events most closely match those in the human's measure. The intelligent crossover picks a crossover point such that the horizontal intervals between adjacent notes in the children will end up as small as possible. After crossover, the child whose first and last new-note events most closely match those in the measure selected from the measure population replaces that measure in the measure population. This results in the measure population being subtly influenced by the human's solo without disrupting the melodic flow of the phrases.

GenJam in the Woodshed

So where do those individuals in the measure and phrase in the populations come from, or in musical terms, where does GenJam get its ideas? Jazz players often refer to long hours "in the woodshed" when describing the hard work of acquiring improvisational ideas and skills. Two classic woodshed activities are transcribing and then memorizing jazz solos of the masters, and trying out one's own ideas, often with the aid of music-minus-one records. GenJam uses adaptations of both activities to build its measure and phrase populations, but each activity leads to a different version of GenJam.

The original version of GenJam[3] is an interactive genetic algorithm (IGA), which requires a human mentor to provide fitness for each individual. IGAs are common in creative domains where the worth of an individual is an aesthetic judgment that can't be encoded in an algorithm. Figure 4 shows how a human mentor trains a GenJam soloist with an IGA. In this version the measure and phrase populations are stored as explicit text files. When GenJam is executed, these

files are read into the internal data structures for the populations. To train GenJam, its mentor creates an arrangement for a tune in which every chorus is a GenJam solo. As GenJam improvises, the mentor listens and provides feedback in real time by typing 'g' for good and 'b' for bad. Each time the mentor types a 'g', the fitness for the currently playing measure and phrase individuals is incremented by 1. When the mentor types a 'b', those fitness values are decremented.

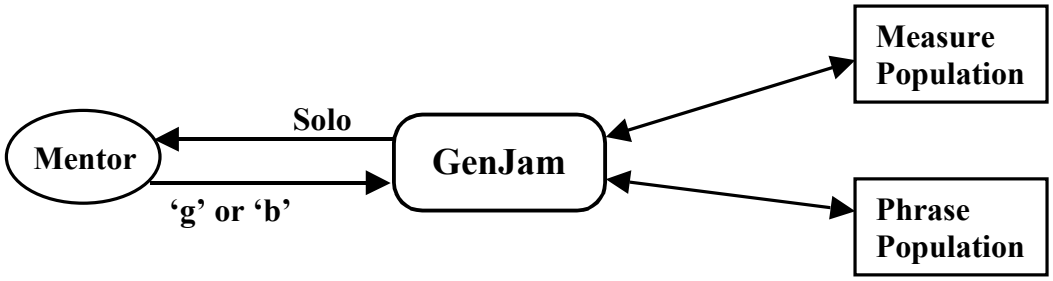

Figure 4. Training GenJam with an IGA

A typical training session begins with randomly generated initial populations (generation 0) and proceeds with a sequence of tunes, each tune resulting in a new generation. Both the measure and phrase populations evolve with the children of the better individuals replacing the worse individuals of the previous generation. The algorithm for creating new individuals is:

> Select 4 individuals at random to form a family (tournament selection)
> Select the 2 family members with the greatest fitness to be parents
> Perform a single-point crossover between the chromosomes for the two parents
> Mutate one of the resulting 2 children
> Replace the two non-parent family members with the new children in the population

In a single generation half of the individuals in each population are replaced (a 50% generation gap). This has the effect of preserving favored licks and replacing undesirable licks with variations of favored licks. It also can result in convergence on minor variations of a few super licks[4]. Convergence on a single best individual is highly desirable in standard GAs, but in GenJam such convergence results in a small set of overused ideas, which actually does model the behavior of some jazz players the author has encountered at jam sessions but is certainly not preferable in a sideman. To combat convergence GenJam uses diversity mutation operators and heuristics to insure that new children are sufficiently different from existing population members. Another problem with this version of GenJam is the fitness bottleneck formed by the necessity for the mentor to experience each individual in both populations in order to provide fitness. This fitness bottleneck is notorious in IGAs[5].

The other version of GenJam[6] eliminates this fitness bottleneck by eliminating fitness entirely. This rather radical departure from the standard EC paradigm makes GenJam an autonomous agent but calls into question its status as an EC system. The EC paradigm can be viewed as a generate-and-test paradigm, where the generators (initialization, crossover, mutation, selection,

replacement) are typically dumb (usually random, which is about as dumb as you can get), and the testing (fitness) is very smart. In fact, fitness is the only intelligent component in standard EC systems.

The meta-evolution of GenJam from an IGA to an autonomous agent is an interesting story. When developing GenJam's interactive capabilities, specifically trading fours, it became quickly evident that there could be no fitness in that process because there was no way for a mentor to provide fitness in time for GenJam to perform the four it had just built. Indeed there is only a 1/32 note interval between the time when GenJam has built the chromosomes (GJNF) of the human's four and the time when it must play the mutated four as its response. This puts pressure on the mutation operators to always generate good results, given a reasonable (or even an unreasonable) four from the human. Consequently, the intelligence is in the mutations. When trading fours, GenJam always plays a competent four and often comes up with a truly stimulating four without the need for fitness.

Extending this idea into GenJam's measure and phrase populations, if GenJam initialized its populations with individuals that sounded good and performed intelligent crossovers and mutations that guaranteed pleasing variations, there would be no need for a mentor. In other words, if GenJam's generators were smart enough, there would be no need to test at all, and fitness could be eliminated altogether. Autonomous GenJam does exactly that, as illustrated in Figure 5.

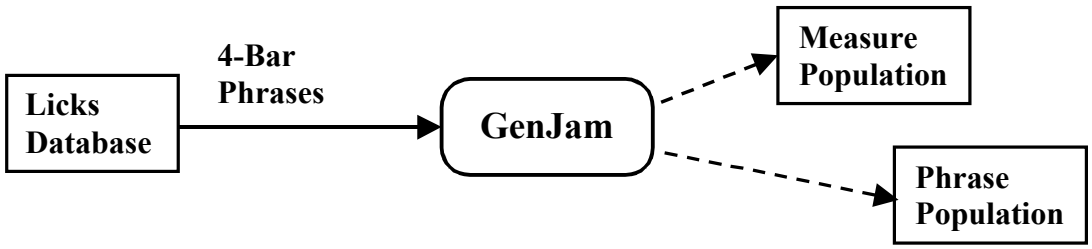

Figure 5. Autonomous GenJam creating populations from Licks

The licks database consists of at least 16 four-measure phrases in GJNF. While these licks could come from anywhere, the author currently uses databases adapted from a recent publication, *1001 Jazz Licks*[7], which conveniently provides 1001 four-bar licks in a variety of jazz styles. The author has hand-encoded roughly a third of these 1001 licks in GJNF to build nine different databases. When a tune is played, GenJam reads the licks from the licks database selected for that tune and builds the measure and phrase populations as follows:

Select 16 four-measure licks at random from the licks database
Build the measure population from the 64 measures in those licks
Build the first 16 phrase individuals to represent the original 16 licks
Build 32 more phrase individuals by applying intelligent crossover to pairs of the first 16

The intelligent crossover operator operates on a pairs of phrases and selects crossover points such that the resulting measures will retain the horizontal intervals of the parents at the crossover points as closely as possible. The child phrase that best preserves the parent's horizontal intervals is then selected for inclusion in the phrase population. This intelligent crossover generates new children that blend the original phrases and sound good without the need for fitness. When combined with the intelligent measure crossover with human measures during performance, as described in the Interactivity section, the resulting full-chorus solos tend to be stimulating and fresh. One can imagine that autonomous GenJam has moved to Garrison Keilor's Lake Wobegon, where the stored licks are strong, the heard licks sound good, "and all the children are above average."

GenJam on a Gig: Interacting with Performer and Audience

So, what does all this technology bring to a jazz gig? In an IT context, it is useful to consider the HCI aspects of GenJam, with respect to both the audience and the performer[5]. Don Norman's elegant framework of user interaction[8] is an excellent jumping off point for describing how users form mental models from which they can deal with technology, and we'll begin by applying Norman's framework to the audience interface to GenJam.

In Norman's framework the designer's mental model of the system informs the creation of a *system image*, which the user accesses and manipulates to form a user mental model of the system. The user interprets the system image in the context of his or her knowledge, experience and expectations to create a mental model that hopefully facilitates the user's achieving some goal by using the system. In musical performance, the goal is hopefully to enjoy the music, and the system image includes all visual and aural aspects, including the name of the performing group, information in a printed program, the appearance of the group on stage, the "gear" used by the group, the stage presence of the performers, and, of course, how the group sounds.

Each audience member brings a set of expectations to a performance, including his or her expectation of what music in the perceived genre should sound like. With GenJam that expectation is hopefully straight-up jazz. As described in the Interactivity section, jazz is supposed to feature spontaneous interaction between the performers in the group. Most concertgoers expect to see the interaction as well as hear it, but with GenJam, the interaction, while certainly present, is aural only. In fact GenJam's visual stage presence is non-existent—its performance is manifested only by sound coming out of speakers. This coupled with the author's subdued stage presence (in keeping with "cool" jazz) leads to a performance with little visual impact, particularly when GenJam takes a full-chorus solo.

This makes GenJam less successful in foreground performance settings like a jazz club than it is in background settings like a reception or mid-ground settings like a coffeehouse. The audience's expectation in a foreground setting is to pay primary attention to the entertainment, but the expectation in a background setting is to chat with others while ear candy fills the air. The opening reception at this conference is a classic background gig, and many attendees will likely not realize that the music is live unless they happen to glimpse the trumpet. At a typical reception a few people will watch the Virtual Quintet for a while, venture up to look at the computer screen, and ask questions between numbers or while GenJam is soloing. This brings

the performance into the foreground for those audience members at that time, but there is no expectation for the entire audience to "pay attention." Incidentally, GenJam's visual display is minimal and essentially tells the human what GenJam is doing now and will do in the next chorus so that the human soloist doesn't get lost in the arrangement, which brings up the other HCI user class to discuss—the performer.

From a performer's perspective (at least the author's perspective), GenJam is a stimulating and even formidable sideman, particularly when trading fours and improvising collectively. Part of the jazz tradition when trading fours in chase choruses is a competitive one-up-man-ship, where each soloist takes the other player's last four and "plays rings around it" in response. This involves hearing the opponent's four clearly and developing or extending it in real time. GenJam hears me better than any human I've ever encountered at a jam session, and its developments (mutations) are often more involved than a human could manage in real time (How many improvisers can play a retrograde inversion normalized to the range of the original four while modifying the pitches to fit the chords in the next four?). Consequently, GenJam can be a monster in chase choruses, and it provides the human improviser with quite a challenge.

The collective improvisation modes, where GenJam plays back what the human plays delayed by some time interval, present a different challenge for the improviser. The goal here is to play duets with oneself in real time. I find that a delay of four measures can be too long because it is hard to remember what I played four bars ago. A delay of one measure usually works well because I can rest every other measure to "trade ones," play different figures to generate counterpoint, attempt to play a transposed version of the last measure to generate harmony, or (usually) a combination of all three as the mood dictates. In this context, mutation is not desirable because if I am playing a harmony or counterpoint with what I played in the recent past, I'd like the past to stay put. It's hard enough to play a complementary response to the past while planning for the future at the same time, without having the past shift under my feet. In short, this recent addition to GenJam's improvisational modes is a terrific and stimulating challenge.

GenJam as an IT Application

A defining aspect of Information Technology as an emerging discipline is that it emphasizes the effective use of existing technology over the creation of new technology. In this sense, the GenJam project is very much an IT project. From the beginning, the philosophy has been to see how far one can get by adapting and integrating off-the-shelf components, both hardware and software, and to create from scratch as little as possible. For example, GenJam's rhythm section is Band in a Box, a commercial product[9]. Other computer jazz researchers have wrestled with creating an interactive rhythm section that attempts to respond to the soloist the way a sensitive human rhythm section could. Unfortunately, these projects, while interesting, have not led to performable results. The goal of the GenJam project is to play gigs, not to study how rhythm sections interact with soloists, so the use of a canned but competent rhythm section, which Band in a Box provides, was clearly prefereable.

Pitch tracking is another rock on which interactive music systems can run aground. The Roland GI-10 used by GenJam makes plenty of errors and is definitely inferior to the pitch tracking

systems used by other researchers, but since the target is GJNF, the tracker only needs to be close, and the GI-10 is perfectly adequate. Again, the goal is pragmatic, and the robustness of GJNF significantly lowered the bar for the pitch tracker.

One slight downside to the IT philosophy in the GenJam project comes with the use of the Carnegie Mellon MIDI toolkit (CMT), which was built by CMU's Roger Dannenberg in the early 1990's[10]. The CMT is a C-based development environment that takes care of the MIDI interface and real-time event scheduling issues that are critical to a real-time interactive MIDI system like GenJam. The author had no desire to implement the low-level environment, and the CMT provided a perfect software platform on which to build GenJam. The programmer's interface is clean and elegant, and the choice of C as a language was not inappropriate when the project began in 1993. However, like most university software, the CMT has not been maintained as well as a commercial product, and the author has had to come to terms with the fact that GenJam is a… well… a legacy system. The CMT has kept the author using version 5 of Think C running on a Macintosh Powerbook 180 under MacOS version 7.1. This has not limited GenJam's development over the years, but it has made GenJam impossible to distribute, which isn't really a bad thing, since the original intention for GenJam was to be the author's sideman.

The legacy issue does bring up the question of what lies in GenJam's future. Clearly it would be nice to port GenJam to a more modern development environment like Java, but to date there are no classes available that handle what the CMT does, mainly because the real-time requirements are too close to the operating system for a clean Java class.

An ongoing issue is making GenJam sound more human (less mechanical). GenJam already uses a host of heuristics to subtly randomize its playing in human-like ways, and the recent addition of a physical modeling synthesis card has made most of GenJam's voices more realistic. However, the parameters available in the physical modeling card are extremely powerful and offer the opportunity to greatly increase GenJam's expressivity. Tweaking those parameters will require a much more sophisticated (knowledge intensive) performance module, which the author is loath to get sucked into.

Finally, the MIDI issue must be addressed. When the first version of GenJam was built in 1993, MIDI was the only rational choice for the underlying technology. The computers of that time were incapable of dealing with real-time audio on the scale required by GenJam, and the then-emerging general MIDI standard provided a good IT-oriented way to use off-the-shelf tone generators and controllers that integrated well with one another and with a computer. These days digital samples can be computed in real time, and the potential exists for going beyond MIDI to technologies that provide greater realism and responsiveness. In the mean time, however, GenJam continues to get gigs, and I continue to grow with it as a jazz player.

References

1. Bentley, Peter, An Introduction to Evolutionary Design by Computers. In Peter J. Bentley (ed.), *Evolutionary Design by Computers*, pp. 1-73. San Francisco, Morgan Kaufmann, 1999.

2. Bentley, Peter J, and Corne, David W., Creativity in Evolution: Individuals, Interactions, and Environments. In P. J. Bentley and D. W. Corne (ed.), *Creative Evolutionary Systems*. San Francisco: Morgan Kaufmann, 2001.
3. Biles, John A., GenJam: Evolution of a Jazz Improviser. In P. J. Bentley and D. W. Cor (ed.), *Creative Evolutionary Systems*. San Francisco: Morgan Kaufmann, 2001.
4. Biles, J. A., GenJam: A Genetic Algorithm for Generating Jazz Solos. In *Proceedings of the 1994 International Computer Music Conference*, ICMA, San Francisco, 1994.
5. Biles, John A., Life with GenJam: Interacting with a Musical IGA. In *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, pp. 652-656, Tokyo: IEEE, 1999.
6. Biles, John A., Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program*, San Francisco, July, 2001.
7. Shneidman, Jack, *1001 Jazz Licks*. New York: Cherry Lane Music Company, 2001.
8. Norman, D. A., *The Design of Everyday Things*. Doubleday, New York, 1988.
9. Gannon, Peter, *Band-in-a-Box*. PG Music, Inc., Hamilton, Ontario, 1991-2002, http://pgmusic.com/.
10. Dannenberg. Roger B., *The CMU MIDI Toolkit, Version 3*. Carnegie Mellon University, Pittsburgh, PA, 1993.

John A. Biles
Al Biles (it's a middle name thing) has been the Undergraduate Program Coordinator in RIT's Information Technology department since 1996. Al joined the RIT faculty in 1980 and has served in various capacities, including Coordinator of Student Services and Advising in the Undergraduate Computer Science department, Artificial Intelligence Area Coordinator in the Graduate Computer Science department, and Department Chair of the Computer Science department, from 1990-93.

Al's recent teaching in the IT department has focused on computer music and HCI, but in previous lives he taught courses in genetic algorithms, expert systems, logic programming, speech and natural language understanding, and knowledge acquisition, along with the usual programming and other CS core courses. He was a co-principal investigator on RIT's speech understanding project in the Northeast Artificial Intelligence Consortium, sponsored by the Rome Air Development Center in the 1980's.

Al began work on GenJam during a well-earned professional development leave in the 1993-94 academic year and has milked it for at least one paper a year since then. More importantly, he performs regularly with GenJam in his Virtual Quintet, which allows him to indulge his passion for jazz while calling it IT research. GenJam combines everything Al likes to do: play trumpet, program, arrange tunes, buy music gear, and perform at receptions like the one on Thursday.

Email: jab@it.rit.edu
Phone: 585-475-7453
Web: http://www.it.rit.edu/~jab/