Neural Network Fitness Functions for a Musical IGA

John A. Biles
Information Technology Department
Rochester Institute of Technology
102 Lomb Memorial Drive
Rochester, NY 14623-5608
jab@it.rit.edu
http://www.it.rit.edu/~jab/

Peter G. Anderson
Computer Science Department
Rochester Institute of Technology
102 Lomb Memorial Drive
Rochester, NY 14623-5608
http://www.cs.rit.edu/~pga/

Laura W. Loggi
Computer Science Department
Rochester Institute of Technology
102 Lomb Memorial Drive
Rochester, NY 14623-5608

Abstract

This paper describes recent enhancements to GenJam, a genetic algorithm-based model of a novice jazz musician learning to improvise. After presenting an overview and update of the current interactive version of GenJam, we focus on efforts to augment its human fitness function with a neural network, in an attempt to ease the fitness bottleneck inherent in musical IGAs. Specifically, a cascade correlation technique was used with data taken from populations of musical ideas trained by human mentors interactively. We conclude with a discussion of why this approach failed, and we speculate on approaches that might work.

1 Introduction

Applications of genetic algorithms in artistic domains are often hampered by the lack of an algorithm for determining fitness. In such domains fitness typically reflects an aesthetic judgment of which individuals in a population are better or worse, based on subjective and often ill-defined artistic or personal criteria. A genetic algorithm that uses human judgment to provide fitness is called an interactive genetic algorithm (IGA), a reference to its interactive training cycle. This cycle typically begins with the presentation of the individuals in the current population for the human mentor to experience. In visual domains, where each individual typically decodes to an image, all the individuals are usually presented at once, often in reduced size so that the entire population can be viewed at once. The mentor can compare and contrast the images

concurrently and determine the fitness of each individual in the context of all the others [Sims, 1993; Haggerty, 1991; Caldwell and Johnston, 1991].

The temporal nature of musical domains, on the other hand, prevents the compressed, parallel presentation of individuals. First, musical objects cannot be presented in a compressed form without distorting them. The musical analog to a reduced image would be a sped up musical sample, which would make the Three Tenors, for example, sound like the Three Chipmunks. Even if the correct pitch is preserved, the tempo would be altered, which certainly changes the perception of a piece of music. The second problem is that multiple musical samples cannot be presented concurrently without obscuring the identity of each individual. The eye can focus on one image at a time, but the ear cannot isolate one melodic line from a randomly contrapuntal piece of music.

The net result for music, then, is that each individual in a population must be presented individually and in real time. This leads to a severe *fitness bottleneck*, which often limits the population size and the number of generations that realistically can be bred in a musical IGA [Biles, 1994]. These limits are necessary not only to cut down on the length of time it takes to train a musical IGA, but also to help reduce the unreliability of human mentors as they attempt to sort through the individuals in a population, listening to only one sample at a time.

The musical tasks that have been performed by IGAs include generating a sound sample to be used as the source for a digital instrument [Takala, 1993], generating a single rhythm measure that might be looped to create a percussion track [Horowitz, 1994], evolving an "ear" module in a multi-level GA-based composition system [Jacob, 1995], and building a knowledge base of melodic ideas for use in improvising jazz solos [Biles, 1994]. In the Takala and Horowitz studies, the goal was to generate a single worthy individual, and the individuals in the populations decoded to very small musical objects. The relatively modest musical goals of these studies minimized the impact of the fitness bottleneck by allowing individuals to be experienced rapidly.

The goals of the Jacob and Biles studies, on the other hand, were to create entire populations that worked together, not just a single best individual. In the Jacob study, each individual decoded not to a musical object, but to a specialized mini-fitness function that evaluated harmonic combinations. Interactivity occurred as the human listener evaluated a given musical example and then assigned weights to the individuals according to how well they agreed with the listener's opinion of the sample. This is an easier task for the human in that only one musical sample is presented at a time, and the human simply reviews how each individual in the population rated that example.

In Biles's GenJam, the populations represent a cooperating knowledge base of melodic ideas that serve as the building blocks for improvised jazz solos. Each individual decodes to a measure or phrase of music and has to be heard both in an arbitrary harmonic context and in the context of other individuals. The following sections will summarize GenJam's operation, detail its genetic representations, and describe our attempts to use neural network techniques to automate or at least augment the acquisition of feedback.

2 Overview of GenJam

GenJam models a student learning to improvise jazz solos under the guidance of a human mentor. As a featured member of the Al Biles Virtual Quintet, GenJam has performed in numerous concerts, demonstrations and other "gigs" over the last two years, and has even recorded a CD [Biles, 1995b]. GenJam's current repertoire includes over 60 jazz tunes in a variety of styles and tempi. In addition to the 4/4 version discussed in this paper, it can play in 3/4 and 5/4 time, and it can deal effectively with complex contemporary harmonic progressions. In a given performance six or more separately trained soloists might be used, each on tunes of different styles. A typical tune would feature several choruses, usually in a head-solos-head structure. Introductions and codas are supported, as are choruses in which GenJam trades fours or eights with a human soloist. Specialized versions of GenJam also have been used as a vehicle to explore audience-mediated performance, where the audience acts as a collective mentor for several training tunes [Biles, 1995a] before the resulting soloist performs with another human player.

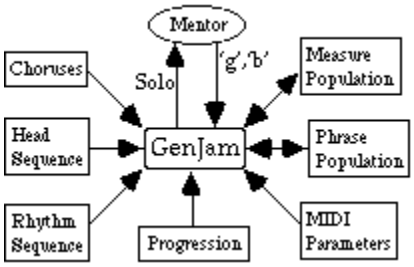

Figure 1. GenJam System Architecture

GenJam was developed in a Macintosh/Think C environment on top of the CMU MIDI Toolkit [Dannenberg, 1993]. Figure 1 shows GenJam's system architecture and provides a visual overview of its operation. To improvise on a tune, GenJam reads a progression file, which provides it with the tempo, rhythmic style (swing or even eighth notes) and the chord progression of the tune being played. It also reads a MIDI sequence for the rhythm section, which has been pre-generated using Band-in-a-Box [Gannon, 1991], and a MIDI sequence for the head (pre-generated melody and harmony parts). The choruses file tells GenJam when it should solo, trade fours or eights, rest for a human soloist, or play the head or a pre-written riff. The MIDI parameters file provides settings for instruments, loudness, reverb, stereo pan, and other synthesizer parameters.

GenJam improvises on the tune by building choruses of MIDI events decoded from members of the measure and phrase populations. Since, as we shall see, a phrase is implemented as a sequence of four measures, these two populations form a mutually dependent hierarchy of melodic structures. One can think of GenJam's measure and phrase populations as being its store of melodic ideas or licks. In breeding successive generations of these populations, then, GenJam builds a better and better collection of licks from which it can construct solos on arbitrary tunes.

While listening to a solo, the mentor can type one or more `g's if a portion is judged to be good, or one or more `b's if a portion is judged to be bad. The fitness for a given measure or phrase is accumulated by incrementing counters for the currently playing measure and phrase every time a `g' is typed, and decrementing them every time a `b' is typed. The modified fitness values are written back to the population files after the solo terminates.

When GenJam is in breeding mode, selection, crossover and mutation are applied, to replace half of each population by new offspring before a solo is presented for feedback. Selection and replacement are handled with a modified tournament selection scheme. Four individuals are selected at random from the population. The two fittest individuals in this "family" serve as parents for a single point crossover, which generates two children. One of these children then experiences a "musically meaningful mutation," which does considerably more than flip an occasional bit [Biles, 1994]. The two new children then replace the worst two individuals from the original family in the population. These operations are applied to both the measure and phrase populations, whose representations are described in the next section.

3 GenJam Representations

GenJam uses a cooperating, two-level, position-based, binary representation scheme. An individual in the measure population maps to a sequence of MIDI events, as will be detailed below. An individual in the phrase population maps to indices of measures in the measure population. In this way the two populations provide a crude approximation of the hierarchical nature of music, in that phrases are made up of measures, which are made up of notes. As was mentioned above, GenJam uses the entire populations of measures and phrases to build a solo, not just a single "best" measure or phrase. In this way GenJam more closely resembles a classifier system [Goldberg, 1989] or the "musical strata" of Horner [1993].

In actual operation, all individuals in both populations are initialized to random chromosomes and fitnesses of 0. To illustrate the representations, however, Figure 2 shows a "composed" example phrase, which maps to a rather unhip rendition of the first four bars of Sonny Rollins's Tenor Madness. In both populations, the single number to the left of the heavy line in each individual is the fitness value, and the remaining numbers represent the chromosome.

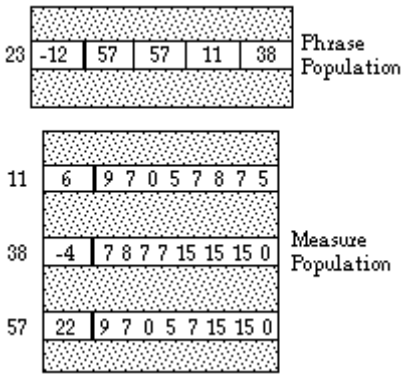

Figure 2. Example Phrase and its Measures

The example focuses on phrase number 23 and its constituent measures. Phrase 23 has a fitness of -12, which means that it has not been particularly well received by the mentor. Its chromosome is the concatenation of four numbers, each of which is a pointer (array index) into the measure population. The current population sizes for GenJam are 48 phrases and 64 measures. The number 64 is not arbitrary because in order to get maximum efficiency from the phrase representation, the size of the measure population must be a power of two; 32 measures is too small a population to support sufficient melodic diversity, and 128 is too large to sample adequately for acquiring fitness.

Individuals in the measure population are made up of a fitness value and a chromosome that is interpreted as a series of eight events, one for each eighth note duration of a 4/4 measure. There are three types of events: a new note, a rest, and a hold. A *new-note* event causes a MIDI note-off followed by a note-on. A *rest* causes a note-off only. A *hold* causes nothing to happen, which has the effect of holding a note already turned on or lengthening a rest.

There are 14 different new note events (encoded as 1-14 in Figure 2), one rest (encoded as 0), and one hold (encoded as 15), which adds up to 16 possible events that can occur at each eighth-note position in a measure. An event, then, can be represented in 4 bits and a 4/4 measure in 32 bits, yielding a melodic space of something less than 2^{32} different measures (a rest following a rest and a hold following a rest will sound the same).

The major advantage to thinking in terms of note-off followed by note-on, rather than the reverse, is that note durations can be represented in half a bit per event (two bit permutations out of 16 for each four-bit event). This efficiently unifies pitch and rhythmic structures in a single representation, as opposed to the more typical approach of treating pitch and rhythmic sequences separately [Ames and Domino, 1992; Giomi and Ligabue, 1991; Fry, 1984].

The 14 new-note events are mapped to actual MIDI pitches through scales suggested by the chord progression being played. As was shown in Figure 1, a progression file is read and processed before the solo is generated. This results in a note map for each half measure of a chorus of the tune. Each note map is an array of 14 MIDI pitches, roughly in the two octaves ascending from middle C. New note events are simply used as indices into the appropriate note map to produce actual notes. This means that GenJam can develop its ideas to fit different harmonic contexts and will not play a theoretically wrong note.

It also means that the same individual appearing in different harmonic contexts will likely map to different notes. In Figure 2, for example, measure 57 is repeated in phrase 23. If this phrase is applied to the chord progression of the first four bars of Tenor Madness, which is C7 F7 C7 C7, the first instance of measure 57 would map to the notes E C A C, based on the scale suggested by the first C7 chord, while the second instance would map to Eb C G C based on the scale suggested by the F7 chord. This illustrates that the

measure individuals represent somewhat abstract melodic templates, not specific notes. It also illustrates that a good template tends to sound good in most, if not all, harmonic contexts which helps explain why measure individuals tend to acquire consistent and meaningful fitness values fairly rapidly during training. This leads us to our attempts to build a neural network that can at least provide a first pass fitness value for measure individuals.

4 Training GenJam Interactively

As discussed above, the human mentor becomes a fitness bottleneck in any musical IGA, and that bottleneck is especially narrow in GenJam, because the mentor's task is especially challenging. The ideal mentor would be able to reliably rank the individual members of each population according to their musical merit; however, this is clearly an unrealizable goal, given the size of the populations and the inability of mentors to compare individuals easily. Another issue is that individuals can be experienced realistically only in a harmonic context, since, as we have seen, the melodic templates only become instantiated to actual notes when played over the chords of a specific tune.

These constraints led to the simplification of the mentor's interface -- GenJam plays solos accompanied by a synthesized rhythm section, and the mentor simply reacts with simple indications of good, bad or indifferent while GenJam plays. This is actually a good approximation of how human jazz improvisers learn -- they play solos at jam sessions and get real time feedback from the other musicians and the audience. However, while this kind of feedback is realistic, it tends to be inconsistent and incomplete, which introduces considerable noise in GenJam's fitness values. The GA machinery is adept at accommodating this noise, but it presents a severe problem when the fitness data is used to train a neural network.

In training GenJam interactively, mentors tend to lose concentration, particularly in early generations when most of the melodic ideas are literally random. At this early stage, mentors tend to reward anything remotely musical and often have a difficult time recognizing melodic fragments that might have promise. Sooner or later, though, a few pleasant licks begin to emerge, and one or two solid phrases tend to appear, at least by the fourth or fifth generation. Typically, at around the eighth or tenth generation, a "golden" generation occurs where almost all the newly hatched phrases sound musical. At this point, the mentor's standards can shift from rewarding anything that sounds vaguely musical to rewarding only what really sounds nice.

Another training issue arises from the tendency of the GA machinery to converge on highly fit individuals. This can lead to "the lick that ate my solo," when one highly fit individual emerges early and dominates a population. The set of musically meaningful mutation operators includes mutations that thin out overused measures and reintroduce under-used measures in the phrase population in an effort to promote diversity, but mentors often get tired of an overused lick and start punishing individuals in later generations that had been rewarded heavily in earlier generations. This phenomenon also

contributed to our failure to find a neural network solution to the fitness problem, which will be summarized in the next section.

5 Neural Network Fitness Function

Our initial goal was to build a neural network that could at least identify measure individuals that were clearly unmusical so that early generations of the measure population could be bred without inflicting these mostly bad melodic ideas on the mentor. Our approach was to construct a neural network of structure N-M-K. Such a network would have N input nodes, M hidden nodes, and K output nodes. For a single output, whose meaning runs from, say, VERYGOOD to VERYBAD, we would use K=1. A typical input would be a vector of N parameters derived from measure individuals. We start, only knowing N and K. If we choose M too small, the system will never learn the input-output mapping; if we choose M too large, the system will over-fit the training data, be very slow to learn , and will not generalize to a testing set. Cascade-correlation [Fahlman and Lebiere, 1990] addresses this dilemma.

Cascade-correlation (cascor) starts by building a simple network with no hidden nodes and trains it as well as it can. It then adds a single hidden node to try to learn the error and again trains the new network as well as it can. This process continues. Each new hidden node receives input from all the nodes in the previously constructed network, and it is trained to model the error of the previous network. When that training is completed as well as it can be, the output node is trained to give the desired output, using as inputs all the nodes in the network.

Cascor training is fast. As each node joins the network, the weights of its input synapses are trained and frozen; they never participate in later training. Therefore, the time-consuming step of backprop does not occur. Furthermore, Fahlman's quickprop algorithm [Fahlman, 1988] is used to speed up the training even more. Cascor, then, allows us to rapidly and easily investigate whether a neural network approach is reasonable, and, if so, what size network (how many hidden nodes) we should use.

In our experiments, we found that it invariably took a very large number of hidden nodes for cascor to achieve good performance on the training set. Even in the few cases when the network was able to learn the training set, it failed on the testing set, indicating that it overfit the training data and did not generalize. We will not present specific results of the numerous experiments, but we will describe the various input vectors and approaches we tried in our efforts to draw something from the fitness data.

The data we used were derived from the measure and phrase populations of human-trained soloists. The initial experiments were conducted on measure populations alone in an attempt to build a simple and reliable module that could be added to GenJam to generate initial fitness values for newly hatched measure individuals. This focus on the measure population was intended to simplify the neural network training problem by supplying a well defined data set. From a neural network training standpoint, the measure population can be thought of as 64 data points in the melodic space being searched by the

GA. For several of the soloists that have been trained by various mentors, a copy of both populations was saved for each generation. These soloists served as a ready-made data source.

Various combinations of several statistical parameters were tried in forming the input layer. These parameters were derived from the individual measures and included the number of new note events in a measure, the number of unique new note events, the size of the maximum interval (numeric difference between adjacent notes), the number of changes in direction of intervals between adjacent notes, and the following "energy" statistic, which attempts to combine the magnitude of vertical activity with the rate of that activity. Measures with lots of large intervals will be very energetic while measures with small intervals and or long notes will have low energy.

$$\text{Measure Energy} = \sum_{\text{Intervals in Measure}} Interval^2 / Duration^2$$

All of these parameters showed promise in statistical analyses of human-trained soloists [Biles, 1994]. Their population-wide means and distribution shapes changed in expected directions over the course of several generations, and it seemed as though a combination of them would be able to predict measure fitness, at least at a gross level.

Unfortunately, this was not the case. All combinations of the above parameters were tried on numerous combinations of generations taken from multiple soloists, using both scaled and unscaled fitness values. In an effort to correct for the appearance of a given individual in more than one generation, the generation number was added as an input node in some trials, and a policy of using only the most mature instance of an individual (the fitness from the latest generation in which a given individual appeared) was tried in other trials. In an effort to fuzzify the output, the output layer was implemented as two or four nodes (interpreted as two or four gradations of good to bad). In all of these experiments, each of which consisted of numerous trials, the net failed to learn the training set, even when the number of hidden nodes grew to exceed the population size!

Undaunted, we then tried a different set of inputs based on "histograms" of interval and note onset data, both alone and in combination with the parameters described above. We tried this to test the hypothesis that the statistical parameters lost too much information about these distributions. When this failed on the measure data, we extended the approach to phrase data by accumulating frequency data for intervals in the four measures of each phrase and using the phrase fitness as the output. This too failed, so we encoded the actual sequences of intervals in both measures and phrases in an effort to somehow represent the actual melodic contours. With the phrase-level interval sequences, we finally were able to train a network to learn the training set, but the resulting net had more hidden nodes than data points, and it failed miserably on the testing set. In a final set of experiments, we attempted to anchor both interval and note onset structures to the rhythmic structure of the tune by using vectors of events, eight note onset events per measure and/or seven interval events per measure. We also extended this notion to the phrase level. Neither approach worked.

To summarize, then, the populations that so clearly evolve for the better under the guidance of a human mentor are so much statistical mush to a neural network. The last section offers some possible reasons for this disparity.

6 Discussion and Conclusions

The clear and unsurprising conclusion from this study is that humans listen to music in complex and subtle ways that are not captured well by simple statistical models. Upon examining the data, we found numerous situations where two measures were nearly identical in their chromosomes, but had maximally opposite fitnesses. While some of this disparity is clearly due to the noise inherent in the accumulated fitness data, the extreme magnitude of fitness differences in some nearly identical measures must have a more fundamental origin.

Part of the answer lies in the context sensitive nature of musical perception. Our approach defined away the harmonic context by focusing on the measure individuals, not on their instantiations as actual notes. We also ignored the structure of the tunes and where in a tune or phrase a particular measure appeared. The fact that we finally built a network that could learn a training set only after we broadened the focus to the phrase level indicates that we did not include a large enough "window" on the data and, therefore, lost too much contextual information.

As a specific example, the occurrence of large intervals illustrates this context sensitivity. Early generations of measures can be characterized as having a large proportion of large intervals. In literally every soloist trained by human mentors, the average interval size decreases steadily from around seven scale degrees, which would be expected by chance in the totally random zeroth generation, to around 2 scale degrees, which is a typical average for a mature soloist used in performance. The inference we made was that too many large intervals sound too chaotic, and that mentors tend to punish them. However, in any mature population, there are always a few highly fit measure individuals with a large interval. There seems to be two reasons these individuals thrive -- the large interval is either ignored by the ear in favor of an otherwise interesting melodic fragment, or the large interval is itself "interesting" in the context of the phrases that contain the measure. It seems, then, that there are so few absolutes that there is nothing objective to tap for even the crudest determinations of musical merit.

Automating fitness may well require the use of knowledge intensive artificial intelligence techniques such as those used by the music cognition community in their research into artificial intelligence models of musical listening and analysis [Balaban, 1992]. Such approaches may work, but they will be difficult to implement and could require extensive computational resources. One of the appealing aspects of GenJam has been how far it has come using very little hardwired knowledge of music and relying on its mentor to provide the musical knowledge it lacks. From that perspective, it shouldn't be surprising that automating the mentor proved to be the formidable barrier that it apparently is.

References

[Ames and Domino, 1992] Cybernetic Composer: An Overview. In M Balaban, K. Ebcioglu and O. Laske (Ed.), *Understanding Music with AI,* AAAI Press, Cambridge, MA, 186-205, 1992.

[Balaban, 1992] Mira Balaban, Kemal Ebcioglu and Otto Laske. *Understanding Music with AI: Perspectives on Music Cognition*. AAAI Press/MIT Press, Cambridge, 1992.

[Biles, 1995a] John A. Biles and William Eign. GenJam *Populi*: Training an IGA *via* Audience-Mediated Performance. In *Proceedings of the 1995 International Computer Music Conference*, ICMA, San Francisco, 1995.

[Biles, 1995b] Al Biles Virtual Quintet. *GenJam*. Compact disk DRK-144. Dynamic Recording Studio, http://www.dynrec.com, Rochester, NY,1995.

[Biles, 1994] John A. Biles. GenJam: A Genetic Algorithm for Generating Jazz Solos. In *Proceedings of the 1994 International Computer Music Conference*, ICMA, San Francisco, 1994.

[Caldwell and Johnston, 1991] Craig Caldwell and Victor S. Johnston. Tracking Criminal Suspect through "Face-Space" with a Genetic Algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 416-421, Morgan Kaufmann, San Mateo, CA, 1991.

[Dannenberg, 1993] Roger Dannenberg. *The CMU MIDI Toolkit, Version 3.* Carnegie Mellon University, Pittsburgh, PA, 1993.

[Fahlman and Lebiere, 1990] S Fahlman and C. Lebiere. The Cascade-Correlation Learning Architecture. In D. S. Touretsky, ed., *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, San Mateo, CA, 1990.

[Fahlman, 1988] Scott E. Fahlman. Faster-Learning Variations on Back-Propagation: an Empirical Study. *Proceedings of the 1988 Connectionist Models Summer School*., Morgan Kaufmann, San Mateo, CA, pp. 38-51, 1988.

[Fausett, 1993] Laurene Fausett. *Neural Network Architectures*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Fry, 1984] C. Fry. Flavors Band: A Language for Specifying Musical Style. *Computer Music Journal* **8** (4) pp. 20-34, 1984.

[Gannon, 1991] Peter Gannon. Band-in-a-Box. PG Music, Inc., Hamilton, Ontario, 1991.

[Giomi and Lagabue, 1991] Francesco Giomi and Marco Ligabue. Computational Generation and Study of Jazz. *Interface* **20**, pp. 47-63, 1992.

[Goldberg, 1989] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[Haggerty, 1991] Michael Haggerty. Evolution by Esthetics, an interview with William Latham and Stephen Todd. *IEEE Computer Graphics and Applications* **11**, pp. 5-9, 1991.

[Horner, et al., 1993] Andrew Horner, Andrew Assad and Norman Packard. Artificial Music: The Evolution of Musical Strata. *Leonardo* **3**, pp. 81, 1993.

[Horowitz, 1994] Damon Horowitz. Generating Rhythms with Genetic Algorithms. In *Proceedings of the 1994 International Computer Music Conference*, ICMA, San Francisco, 1994.

[Jacob, 1995] Bruce Jacob. Composing with Genetic Algorithms. In *Proceedings of the 1995 International Computer Music Conference*, ICMA, San Francisco, 1995.

[Sims, 1993] Karl Sims. Interactive Evolution of Equations for Procedural Models. *The Visual Computer* **9** (8), pp. 466-476, 1993.

[Takala, 1993] T. Takala, J. Hahn, L. Gritz, J. Geigel, and J. W. Lee. Using Physically-Based Models and Genetic Algorithms for Functional Composition of Sound Signals, Synchronized to Animated Motion. In *Proceedings of the 1993 International Computer Music Conference*, ICMA, San Francisco, 1993.