Exploiting Functional Relationships in Musical Composition

Amy K. Hoover and Kenneth O. Stanley
Evolutionary Complexity Research Group
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2362 USA
{ahoover, kstanley}@eecs.ucf.edu
http://eplex.cs.ucf.edu/neatmusic

To appear in:
Connection Science Special Issue on Music, Brain, & Cognition,
Abington, UK: Taylor & Francis, 21:2, 227-251, June 2009.

Abstract

The ability of gifted composers such as Mozart to create complex multipart musical compositions with relative ease suggests a highly efficient mechanism for generating multiple parts simultaneously. Computational models of human music composition can potentially shed light on how such rapid creativity is possible. This paper proposes such a model based on the idea that the multiple threads of a song are temporal patterns that are *functionally* related, which means that one instrument's sequence is a function of another's. This idea is implemented in a program called NEAT Drummer that interactively evolves a type of artificial neural network (ANN) called a Compositional Pattern Producing Network (CPPN), which represents the functional relationship between the instruments and drums. The main result is that richly textured drum tracks that tightly follow the structure of the original song are easily generated because of their functional relationship to it.

Keywords: compositional pattern producing networks; CPPNs; computer-generated music; interactive evolutionary computation; IEC; NeuroEvolution of Augmenting Topologies

1 Introduction

A most intriguing capability of human composers is that they can often quickly conceive multiple instrumental parts *simultaneously* during the creative process.

For example, Mozart could hear complex multipart pieces form "in his head," suggesting a powerful creative mechanism for generating accompaniment (Ruppel, 1998; Deutsch, 1965; Hymer, 1990). Relatedly, rock guitarists in jam sessions and jazz musicians can improvise together while simultaneously perfectly respecting the interdependencies of their separate parts (Barrett, 1998; Berliner, 1994; Katz and Longden, 1983; Oliver, 2006; Schuller, 1968; Weick, 1998).

Thus although intuition may suggest that complex interdependent constructions should require care and labor to devise, in fact such constructions in music appear almost effortless. Thus it is likely that no explicit serial reasoning is involved in the creative construction of accompanying instrumental tracks. What kind of mechanism then is responsible for such a capability?

This paper suggests a possible high-level answer to this question. The key idea is that different instrumental parts are *functionally* related, which means that one can be expressed as a function of another. Furthermore, although we may perceive the interplay between two or more simultaneous instruments as rich and complex, in fact the function that relates one to the other can be quite simple. Thus, in this view, once a single track such as a melody is created, it can serve as a *scaffold*, i.e. an existing support structure, upon which other tracks are generated. In this way, while composers may *seem* to improvise entire harmonies and drum tracks one note at a time, fundamentally they need only construct a simple function for each part that transforms the scaffold.

In fact, because the scaffold *already* in effect embodies the intrinsic contours and complexities of the song, any transformation of the scaffold *inherits* these features and thereby embodies the same thematic elements automatically. Thus the space of possible transforming functions is highly forgiving, in part explaining why improvising accompaniment can appear effortless. As long as the accompaniment is expressed as a function of the scaffold, it is difficult to go significantly wrong.

While this idea of functions relating one pattern to another is difficult (though not necessarily impossible) to confirm at a neurological level, it does suggest a promising model for computer-generated music. This paper describes the implementation of such a model and presents its results. The goal is to generate drum tracks to accompany existing songs. Because rhythm is simpler than melody or harmony, rhythm generation is an appealing stepping stone to full blown harmonization. It effectively highlights the advantages of the functional perspective in clear and simple terms that do not require musical expertise to appreciate.

Formally, an appealing drum pattern for a particular piece over time can be described as a function of time, $f(t)$. However, a good pattern for a particular drum may be highly complex with respect to t, making its discovery prohibitive. Yet given an existing part $p(t)$ (i.e. the scaffold) that varies over time, it is likely significantly easier to discover the pattern $g(p(t))$ rather than $f(t)$ even though they produce the *same pattern*. In effect, p makes discovering the accompanying pattern easy because it provides the scaffold, thereby allowing the composer to focus only on devising the much simpler $g(p(t))$.

This idea is implemented in this paper in a program called NEAT Drummer,

which automatically generates drum tracks for existing songs. It accepts existing human compositions as input to a type of artificial neural network (ANN) called a Compositional Pattern Producing Network (CPPN; Stanley 2007) and outputs drum patterns to accompany the instruments. The inputs to NEAT Drummer are specific parts of a Musical Instrument Digital Interface (MIDI) file (e.g. the lead guitar, bass guitar, and vocals) and the outputs are drum tracks that are played along with the original MIDI file. That way, outputs are a function of the original MIDI file inputs, forcing synchronization with the MIDI.

To take into account the user's own inclinations, NEAT Drummer allows the user to interactively evolve rhythms from an initial population of drum tracks with the NeuroEvolution of Augmenting Topologies (NEAT) algorithm (Stanley and Miikkulainen, 2002, 2004), which can evolve increasingly complex CPPN-encoded patterns.

The main results are drum tracks for existing songs that tightly follow the contours and idiosyncrasies of individual pieces, yet elaborate and elucidate them in creative and unexpected ways. Even when major transitions occur, because the drum tracks are a function of the music, the drums perfectly follow the transitions.

This functional model of musical composition is then further extended to allow human users to add their own functional influences to create variational motifs outside the confines of the provided song. For example, users can provide a monotonically increasing function (i.e. time), which suggests change over time *even if* the underlying scaffold is repetitive. The result is that the drum track can be made to vary exactly as the user requests, while still seamlessly interweaving with the song. These user-provided functions are called *conductors* in a loose analogy with orchestral conductors, who describe functional contours with their hands that the orchestra follows. The conductors further highlight the simplicity and relative ease of creating subtle overlapping textures through simple functional relationships.

To highlight the importance of scaffolding and conductors, several variants of NEAT Drummer *without* such facilities are compared with NEAT Drummer with its full functionality intact. The result is that the consequent capabilities are significantly impoverished, demonstrating the critical role that scaffolding plays in generating accompaniment.

While the main contribution is a powerful new approach to computer-aided musical creativity, the high-level implication for improvisational accompaniment provides an intriguing clue to how such mechanisms may work in the brain.

The next section provides background for the approach introduced in this paper. This approach is then described in Section 3 and the experimental design is explained in Section 4. Results are disclosed in Sections 5 and 6, and discussed in Section 7.

2 Background

This section first explains interactive evolutionary computation (IEC), which is part of the NEAT Drummer approach, and its application to computer-generated music. The section concludes with a review of the NEAT method.

2.1 Interactive Evolutionary Computation

NEAT Drummer refines its original drum patterns through a process called interactive evolutionary computation (IEC), which means a human, rather than a predefined fitness function, selects the parents of the next generation (Takagi, 2001). IEC implementations typically generate a random initial population. The user then selects the most fit individuals from that population to reproduce, resulting in increasingly complex individuals.

IEC addresses the problem that objective evaluation is difficult in aesthetic or subjective domains such as art and music. By shifting the burden of evaluation to the human, the need to formalize subjective quality is avoided. Richard Dawkins first popularized IEC with Biomorphs, a visual representation of artificial organisms designed to illustrate evolutionary principles (Dawkins, 1986). Biomorphs inspired a proliferation of programs tackling design problems from tool creation (Sato and Hagiwara, 1999) and suspension bridges (Furuta et al., 1995) to education, teaching, and story composition (Kuriyama and Terano, 1997).

Because it can harness subjective preferences, a major application of IEC is art. The power of this approach is evident in visual domains like the L-system-encoded Mutator (Lindenmayer, 1968; Todd and Latham, 1992), Karl Sims' genetic art (Sims, 1991), and Picbreeder (Secretan et al., 2008), a website where users evolve, save, and publish images. Picbreeder evolves its images with NEAT (Section 2.3), the same evolutionary algorithm used by NEAT Drummer.

IEC has also branched into *musical* evolution, such as the Biomorphs-inspired Sonomorphs (Nelson, 1993).The next section reviews several such approaches to computer-generated music, which are often based on IEC.

2.2 Evolutionary Computer Generated Music

The idea that computers might be able to compose music has inspired a diversity of approaches. While this section focuses mainly on evolutionary approaches, a broad review of the area can be found in de Mantaras and Arcos (2002). Often computer generated music utilizes IEC to leverage the subjective capabilities of average human subjects while avoiding the need for musical expertise. For example, among the first IEC music applications is Sonomorphs (Nelson, 1993), which encodes rhythms as bit strings in which a note is either on or off. Direct representation of this type, wherein each note is represented by a single gene, does not attempt to model how humans encode music neurologically. However, the creative evolutionary process is a metaphor for human composition through variation and refinement.

Listeners often feel that computer-generated music sounds artificial and uninspired. Music generators tend to either evolve a solution to fit a particular *a priori* style or improvise pieces that often lack a global structure that holds together the entire song (McCormack, 2005; Husbands et al., 2007).

It is common for music generators, such as Sonomorphs, CONGA (Tokui and Iba, 2000), GP-Music System (Johanson and Poli, 1998), and the GA-based IEC composition system of Onisawa et al. (Onisawa et al., 2000) to focus on composing short phrases rather than on entire songs (Nelson, 1993; Biles, 2007). These short phrases, which are selected and evolved by the user, may be extended through looping or manual juxtaposition, but the overall structure of the song is not itself generated.

A notable example of computational improvisation is GenJam (Biles, 1999), which composes music in the style of jazz in cooperation with a human musician. GenJam listens to human improvisations and interprets and genetically modifies the notes. GenJam can also evolve a soloist that is independent of any particular jazz composition by first training from human input. It integrates its improvisations seamlessly into a musical stream by prescripting when in the song improvisation may occur. In this way, it preserves the overall musical structure provided by the human, although it does not innovate at the level of global structure on its own.

Early connectionist approaches, which also emphasize short phrases, represent change over time through recurrence (Todd and Loy, 1991). Recurrence means that a network can represent a time series through a pattern of cycling activation. Todd and Loy (Todd and Loy, 1991) first applied recurrent ANNs to music generation by training them to reproduce patterns with Real-Time Recurrent Learning (RTRL). Chen and Miikkulainen (Chen and Miikkulainen, 2001) later combined this recurrent learning approach with evolution based on the idea that a simple recurrent network (SRN) can capture a general style of music and then vary it through evolution. This approach succeeded in producing melodies in the style of Bela Bartok on a local level; however, even with recurrence it is difficult to capture global structure.

NEAT Drummer approaches the problem of global structure by generating its rhythms from already-existing instrumental parts that span entire songs, thereby diminishing the need to represent patterns over time through recurrence. The next section describes the NEAT method that implements evolution in NEAT Drummer.

2.3 NeuroEvolution of Augmenting Topologies (NEAT) and CPPNs

NEAT Drummer follows the idea in prior connectionist approaches that an ANN can effectively represent music. Therefore, a method is needed to allow the user to evolve ANNs. The NEAT method, described in this section, is chosen for this purpose in NEAT drummer because it allows ANNs to increase in complexity over generations. In particular, NEAT Drummer evolves a neural-based encoding of drum patterns.

The NEAT method was originally developed to solve difficult control and sequential decision tasks. The ANNs evolved with NEAT control agents that select actions based on their sensory inputs. While previous methods that evolved ANNs, i.e. neuroevolution methods, evolved either fixed topology networks (Gomez and Miikkulainen, 1999; Saravanan and Fogel, 1995), or arbitrary random-topology networks (Yao, 1999), NEAT is notable for beginning evolution with a population of small, simple networks and complexifying the network topology over generations into diverse species, leading to increasingly sophisticated behavior. This section briefly reviews the NEAT method; Stanley and Miikkulainen (2002, 2004) provide complete introductions.

NEAT is based on three key principles. First, to allow ANN structures to increase in complexity over generations, a method is needed to keep track of which gene is which; otherwise, it is not clear in later generations which individual is compatible with which, or how their genes should be combined to produce offspring. NEAT solves this problem by assigning a unique historical marking to every new piece of network structure that appears through a structural mutation. The historical marking is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited during crossover unchanged, and allow NEAT to perform crossover without the need for expensive topological analysis.

Second, NEAT traditionally speciates the population based on topological similarity so that individuals compete primarily within their own niches instead of with the population at large, which protects topological innovations. The historical markings allow structures to be compared for this purpose. However, because the *user* performs selection in interactive evolution instead of the evolutionary algorithm itself, speciation is not applicable in NEAT Drummer and therefore not utilized. Note that in Section 6, variants of regular non-interactive NEAT are compared to NEAT Drummer, and these variants therefore do implement speciation.

Third, unlike other systems that evolve network topologies and weights (Yao, 1999), NEAT begins with a population of simple networks with no hidden nodes. New structure is introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. This way, NEAT searches through a minimal number of weight dimensions and finds the appropriate complexity level for the problem. NEAT Drummer lets the user evolve patterns of increasing complexity through this approach.

Finally, in NEAT Drummer, NEAT evolves a kind of ANN called a Compositional Pattern Producing Network (CPPN), which is designed to compactly represent patterns with regularities, such as pictures and songs (Stanley, 2006, 2007). What distinguishes CPPNs from ANNs is that in addition to traditional sigmoid functions, CPPN hidden nodes can include several classes of functions, including periodic functions (like sine) for repetition and symmetric functions (like Gaussian) for symmetry. An individual network can contain a heterogeneous set of functions in its nodes, which are evolved along with the weights.

To demonstrate the capabilities of such networks, Stanley (2006, 2007) showed how simple canonical functions can be composed to create an overall network

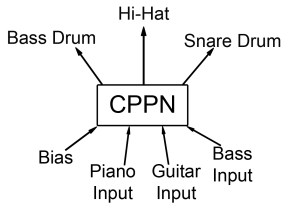


Figure 1: **CPPN Inputs and Outputs** The user selects both a set of inputs from among the channels in the MIDI file and a set of outputs corresponding to specific drums.

that produces patterns with complex regularities and symmetries. Each component function creates a novel geometric *coordinate frame* within which other functions can reside. The idea in NEAT Drummer is that this representation allows drum tracks with regular patterns to be discovered quickly and easily.

The next section explains how CPPNs are evolved in NEAT Drummer to produce rhythms.

3 The NEAT Drummer Approach

The main idea in NEAT Drummer is that the temporal patterns of the instrumental parts of a song can be inherited by the drums by making the drums a function of the other instruments. This section begins by explaining how CPPNs encode rhythm and then details how they are evolved interactively.

3.1 CPPN Rhythm Generation

NEAT Drummer begins by generating an initial set of original drum tracks for a provided song. To initiate this first generation, the user must first specify the inputs and outputs of the CPPN (figure 1) through a graphical user interface (GUI) provided by the program (figure 2).

The *inputs* are individual instrumental tracks from the chosen song and the *outputs* are a set of drums that together produce the entire drum accompaniment.

From the inputs the CPPN derives its original patterns, which are therefore *functions* of the original song (i.e. the scaffold) and its structure. In other words, NEAT Drummer generates a rhythm that is a function of these inputs. Thus, it is important to choose instruments that play salient motifs in the song so that the drum pattern can be derived from the richest structures available. Further texture can be achieved by inputting more than one MIDI channel, e.g. bass and guitar.

Thus the user selects any combination of *channels* representing individual instrumental parts from a MIDI file to be input into the CPPN. In this way,

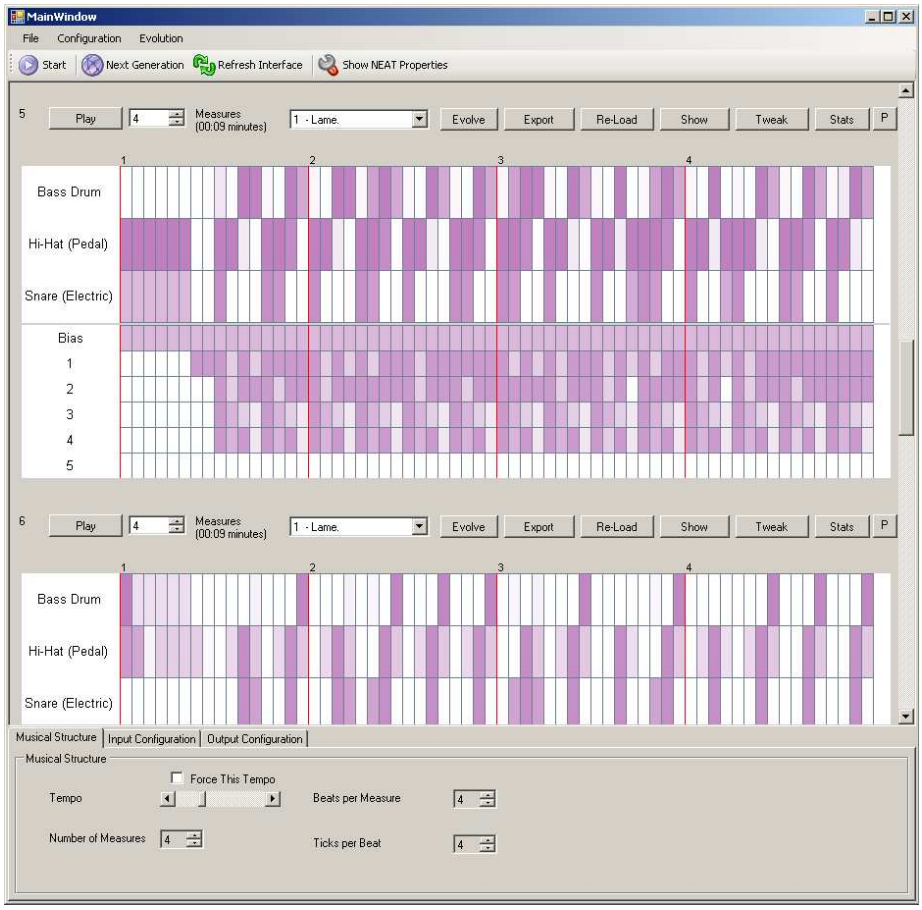


Figure 2: **NEAT Drummer Screenshot** NEAT drummer presents an IEC interface where visual representations of drum patterns help the user to decide whether to listen to each candidate and then select their favorites. This approach, i.e. choosing inputs and outputs and selecting favorites, is designed to allow users to evolve compelling drum tracks without the need for musical expertise.

NEAT Drummer generates rhythms from any MIDI file.

The user also chooses the percussion instruments that will play the rhythm. Each such instrument is represented by a single output on the CPPN. For example, one output may be a bass drum, one a snare, and the final a hi-hat. Any number of drums, and hence any number of outputs, are permissible.

To produce the initial patterns, a set of random initial CPPNs with a minimal initial topology (following the NEAT approach) and the chosen inputs and outputs are generated. The number of inputs in these initial topologies corresponds to the number of instrument channels in the scaffold (e.g. guitar, bass, etc.) plus a bias node. The relationship between the initial topology and the original song is thus established through these inputs, which feed information from the scaffold directly into the network. The number of outputs equals the number of drums in the drum ensemble. The initial minimal topology has random connectivity yet always contains exactly one hidden node. This single hidden node ensures that initial patterns sound more interesting than perceptrons, but are still relatively simple. Note that the *internal* topology is thus unrelated to the scaffold except insofar as it is affected by the number of inputs. Thus the apparent "knowledge" of the provided song in the pattern output by the network is entirely a result of computing a function of the scaffold.

NEAT Drummer then inputs the selected channels into the CPPN over the course of the song in sequential order and records the consequent outputs, which represent drums being struck. Specifically, from time $t = 0$ to $t = l$, where l is the length of the song, the inputs are provided and outputs of the CPPN are sampled at discrete subintervals (i.e. ticks) up to l.

Individual notes input into the CPPN from selected MIDI channels are represented over time as *spikes* that begin high and decrease (i.e. *decay*) linearly (figure 3). The period of decay is equivalent to the duration of the note. That way, the CPPN "hears" the timing information from the supplied channels while in effect ignoring pitch, which is unnecessary to appreciate rhythm. By allowing the spikes to decay over their duration, each note becomes a kind of temporal coordinate frame. That is, the CPPN in effect knows at any time *where* it is within the duration of a note by observing the stage of its decay. That information allows it to create drum patterns that vary over the course of each note.

Interestingly, it is potentially useful also to input temporal patterns that are *not* part of the song itself. Such patterns can provide additional structure to the drums by situating them within coordinate frames that describe how the user wants the song to vary at a meta-level. For example, inputting a simple linear function of time that indicates the *position-in-song* at each tick (figure 4a) *in addition* to the instrument channels means that the output is a function of both the song itself *and* the position-in-song. That way, the CPPN can produce a drum track that shifts gradually from one motif to another over the course of the song.

Similarly, by inputting *position-in-measure* (figure 4b) or *position-in-beat* (figure 4c), the user can bias the output towards progressions across each measure or beat.

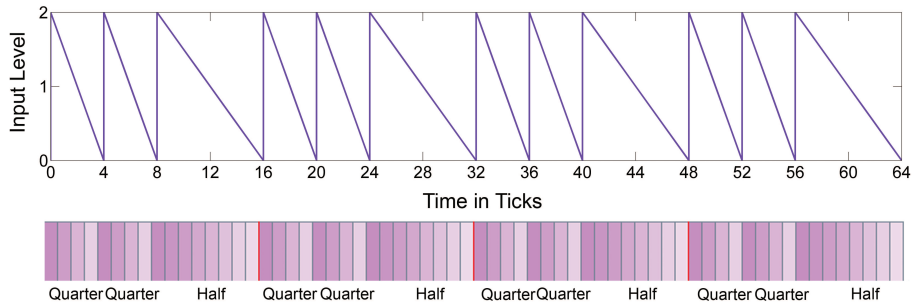


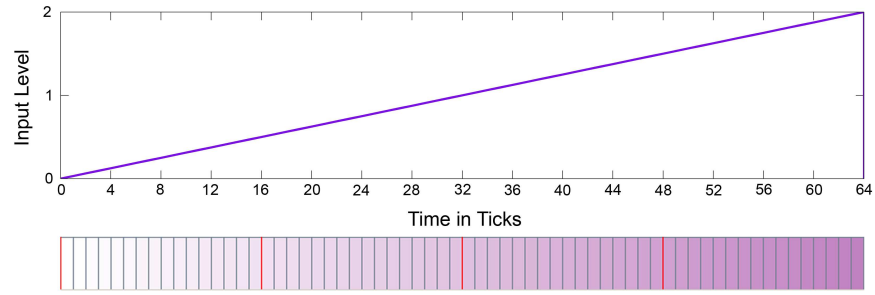
Figure 3: **Channel Input Encoding.** Regardless of the instrument, each note in a sequence in any channel input to the CPPN is encoded as a spike that decays over the duration of note. The pattern depicted in this figure shows how quarter notes decay faster than half notes, thereby conveying timing information to the CPPN, which samples this pattern at discrete ticks. The variable-intensity row of boxes under the spikes depicts the intensity of the spike sampled at discrete time steps (i.e. four per quarter note). The intensity at each timestep is represented by the darkness in its respective column, which indicates how the input channel "sounds" to the CPPN at that moment.

In this paper, these additional inputs are called *conductors* to make a metaphor with the silent patterns expressed to an orchestra by its conductor. Additional inputs that represent desired hidden contours beyond the pattern of the instruments themselves give the user an unprecedented level of control over the nuances of the global output pattern.

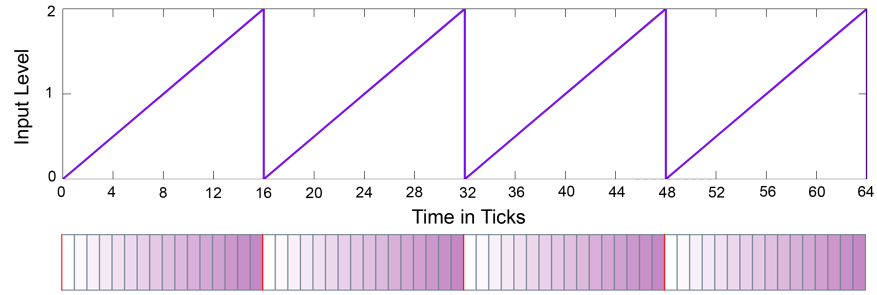
In fact, any arbitrary sequence can be input as a conductor, which in effect simply means a set of note spikes that are never actually heard. Thus the pattern in figure 3, while introduced as an instrumental pattern, could also be a complex conductor pattern that suggests a particular underlying motif that the drums should elaborate. Note that in NEAT Drummer, by convention, conductor inputs that represent time are spikes that start low and *attack*, which conveys the idea of a timing signal, as opposed to notes from scaffolding inputs, which are decaying spikes.

Unlike CPPN inputs, the level of each CPPN output is interpreted as the *volume* (i.e. strength) of each drum strike. That way, NEAT Drummer can produce highly nuanced effects through varying softness. Two consecutive drum strikes one tick after another are interpreted as two separate drum strikes (as opposed to one long strike). To produce a pause between strikes, the CPPN must output an inaudible value for some number of intervening ticks. Because the CPPN has one output for each drum, the end result of activating the network over t ticks is a drum sequence for each drum in the ensemble.

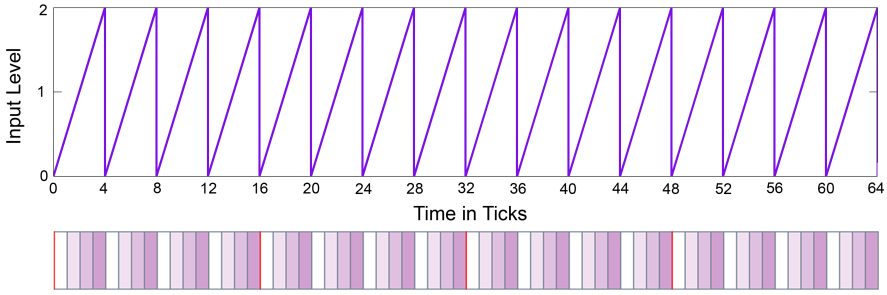
An interesting aspect of this representation is that it does not make explicit use of recurrent connections. While recurrent networks are often noted for their ability to encode temporal patterns (Dolson, 1989; Todd and Latham, 1999;



(a) Position-in-Song



(b) Position-in-Measure



(c) Position-in-Beat

Figure 4: **Potential NEAT Drummer Conductor Inputs.** Each figure depicts four measures of a conductor, which is a temporal coordinate frame optionally provided by the user to provide additional structure to the song. The simplest conductor (a) represents the current position in the song, suggesting a smooth transition across the entire song. Position-in-measure (b) allows the CPPN to know at every moment where it is within the current measure, which allows it to improvise patterns within measures and "understand" the measure structure of the song. Similarly, the time within each four-tick beat can be input as well (c). Conductors offer the user a subtle yet powerful means to influence the overall structure of the rhythm without the need for note-by-note specification.

Chen and Miikkulainen, 2001), it is easier to simply express music as a function of an existing temporal pattern (i.e. the melody and harmony) and thereby affix one pattern to another without needing to *learn* the temporal synchronization itself. Thus, while recurrence is well suited to temporal problems in which the inputs are not known *a priori*, because music is deterministic, recurrence is unnecessary; because the inputs are always the same, the outputs can simply be expressed as a function of the inputs. Thus the CPPN can potentially represent that function without recurrence.

NEAT Drummer generates each of the individuals in the initial population with the same set of inputs and outputs. However, the initial CPPN weights and activation functions for each member of the population are decided randomly. In particular, every input is connected to every output with a randomized weight within $[-2, 2]$. The activation function of each node is assigned randomly from among the following options: sigmoid, binary threshold, Gaussian, linear, multiplication, and sine. To encourage interesting patterns in the initial generation, a single hidden node with a random activation function is also connected into the network by splitting a randomly chosen connection. Each song is divided into ticks (four per beat). At each tick, the vector of note spike values at that discrete moment of time for all the instruments is input. The CPPN is fully activated and the value of each drum output is recorded so that all the generated drum tracks can be visualized or played instantaneously to facilitate interactive evolution, as explained in the next section.

3.2 Drum Pattern Interactive Evolution

As shown in figure 2, NEAT Drummer displays the set of candidate patterns visually after they are generated.

It is important to note that unlike in many evolutionary experiments, patterns in the initial generation *already* sound appropriate. This initial high quality underscores the contribution of the scaffold (i.e. the existing tracks) to the structure of the generated patterns. Thus many appealing patterns already exist in the first generation, demonstrating how quickly appropriate accompaniment can be generated as a function of the source channels. In this way, a major contribution of this research is in showing how rich context can be leveraged by a connectionist system to successfully constrain output to appropriate patterns.

The aim of evolution is thus to elaborate on such patterns. The user can choose to listen to any displayed pattern. When listening, the user can listen to the drum track alone or the drum track with its associated song. The visual presentation allows the user to quickly identify unappealing patterns without wasting time listening to them (e.g. wherein the bass is hit over and over again without pause).

Then either the user rates the individual patterns from which NEAT Drummer chooses parents or the user selects a single parent of the next generation. Further rounds of selecting and breeding continue until the user is satisfied. In this way, drum tracks evolve interactively. Because of complexification in NEAT, they can become increasingly elaborate as evolution progresses.

To encourage rapid elaboration over generations, the probability of adding a connection or node in NEAT was 90%. While this high probability would be deleterious in typical NEAT experiments (Stanley and Miikkulainen, 2002, 2004), because drum tracks tend to follow song structure, this domain supports adding structure quickly. The mutation power, i.e. the maximum magnitude of weight change, was 0.1 and the probability of mutating the weight of an individual connection was 90%.

Finally, it is also important to note that in principle, the idea of representing musical structure in a connectionist system through scaffolding and conductors could be combined with a different evolutionary algorithm, or even a different training mechanism. Thus while NEAT is a robust algorithm from which to demonstrate the power of scaffolding, the benefit of the scaffolding approach is likely compatible with other connectionist training approaches as well.

3.3 Musical Instrument Digital Interface

NEAT Drummer reads its input channels from Musical Instrument Digital Interface (MIDI) files. Standard MIDI format (SMF) is the most common MIDI filetype. SMF format includes any number of tracks, each of which contains a sequence of instrumental events that occur in up to 16 channels. Each channel contains events that tell a particular instrument when and how loudly to play. According to the specification, most of the instrument sounds can occur in any of the 16 channels with the exception of percussion, for which channel 10 is reserved.

NEAT Drummer can input any combination of the 16 channels into the CPPN. That is, given a MIDI song, NEAT Drummer generates a drum pattern as a function of any subset of the preexisting bass, guitar, vocals, etc. The resulting drum patterns are all explicitly functions of the inputs, so if part of a MIDI is input to the ANN, the percussion follows the structure of that part. In this way, NEAT Drummer can generate percussion for MIDI songs based on any subset of the preexisting instrument parts.

4 Experimental Design

This paper includes two sets of experiments. The first set focuses on the capability of the scaffolding approach to generate drum tracks. The second set compares several other approaches with the scaffolding approach, both interactive and supervised, to provide an objective validation of the methodology.

Also, because music appreciation is largely subjective and auditory, the results of NEAT Drummer should be judged in part on that basis. Therefore, MIDI files for every experiment reported in this paper are available online at `http://eplex.cs.ucf.edu/neatmusic/`. We invite readers to listen to the recordings and judge the natural quality of the percussion tracks discussed in Sections 5 and 6.

4.1 Testing Scaffolding

The first set of experiments aim to determine whether drum tracks generated for particular songs are appropriate and nontrivial. The hope is that they respect the structure and transitions of the song yet do not mimic its instruments superficially. Such sophisticated correspondence can confirm the capacity of functional relationships to generate plausible, human-like accompaniment.

Specifically, the first two experiments in this set investigate what happens when salient instrument channels are input alone to the CPPN, which generates drum tracks for the folk songs Johnny Cope and Oh! Susanna. A follow-on experiment explores the consequences of inputting *both* instrument channels *and* conductors for the folk song Oh! Dem Golden Slippers. The question is whether the conductors add a dimension of variation that is seamlessly combined with the structure of the original song in the resultant drum track. A complex conductor is then input by itself into a CPPN to isolate its effects and easily discern the functional relationship between the conductor and its outputs.

4.2 Comparisons

The second set of experiments are designed to scrutinize the power of scaffolding via input from the original song by attempting to achieve comparable output drum tracks *without* providing the original song as input. The aim is to illustrate the contribution of such scaffolding by investigating how other approaches fare without it. To control specifically for the contribution of the scaffold, each such attempt is still a variant of NEAT. That way, differences in performance are attributable to representation and scaffolding.

In this spirit, first, ten 30-generation attempts are made to interactively evolve accompanying drums to Johnny Cope with NEAT *without* the drum channels from the song input into the network. Instead, in the first five attempts, the network is recurrent and inputs only a bias. These attempts compare the capabilities of a recurrent network without any scaffolding to those of the scaffolded networks. In the last five attempts, the network is feedforward and provided only position-in-song as input. Typical best results are presented.

Second, three *target-based* experiments form a more objective comparison. In these target-based runs, the aim is to reproduce a specific drum track that was previously evolved with NEAT Drummer (i.e. with the scaffold provided) as an accompaniment to Johnny Cope. This drum track is set as the target for the target-based experiments, which do *not* have access to the scaffold.

Target-based runs rely on the same NEAT algorithm as NEAT Drummer; however, the computer performs selection instead of a human user. Selection is performed as in regular NEAT, wherein each individual in the population is assigned a *fitness* based on the sum-squared error between the target pattern and the attempted output:

$$f = \frac{\sqrt{\sum_{t=0}^{t=l} M^2 - \sum_{t=0}^{t=n} (x_t - y_t)^2}}{lM^2}, \tag{1}$$

where M is the maximum possible error at any tick t, l is the number of ticks, x_t is the target note value at tick t, and y_t is the output value of the network at tick t. Note that if there are multiple output tracks, this expression is applied to each and the fitness is the average. This fitness function is designed to approach 1.0 the better the output matches the target.

The main question is how hard it will be for NEAT to evolve the very same rhythm that it evolved with the scaffold. Three alternative representations are tested in this way:

- recurrent neural networks with only a bias input,
- feedforward networks with only a position-in-song conductor input, and
- feedforward networks with *both* a position-in-song conductor and a position-in-measure conductor input.

In these target-based experiments, NEAT is run with typical successful parameter settings for regular non-interactive evolution (Stanley and Miikkulainen, 2002, 2004). In particular, the population size was 100 and probability of adding a connection or node in NEAT was 3% and 5%, respectively. The mutation power, i.e. the maximum magnitude of weight change, was 0.1 and the probability of mutating an individual connection was 80%. The compatibility coefficients for determining to which species individuals belong (Stanley and Miikkulainen, 2002) were $c_1 = 1.0$, $c_2 = 1.0$, and $c_3 = 0.4$. The compatibility threshold C_t was adjusted dynamically in increments of 0.5 to maintain a stable equilibrium of eight species.

If it turns out that any of these variants can evolve the target drum track, it will show that the scaffold is not necessary to provide a context. On the other hand, if none of the representation can evolve the target, it shows the critical contribution of the scaffold.

In summary, experimental results are divided into two parts: First, the power of scaffolding is tested through interactive evolution; second, scaffolding is compared to several variants of NEAT Drummer without scaffolding. The next section details the results of evolving interactively with the scaffold.

5 Scaffolding Results

While NEAT Drummer can theoretically input a drum channel from a MIDI file and thereby generate variations of the percussion, this section focuses on drum tracks generated from inputting non-percussion instruments, like guitars and bass. Thus the MIDI songs input in this section do not include drums in their original form.

Results in this section are reported through figures that are designed to demonstrate the relationship between the CPPN inputs and outputs as the song progresses over time. In the figures that follow, the inputs are arranged in rows at the bottom of each depiction and the outputs are the rows above. Time

moves from left to right and each discrete column represents a tick of the clock. No instrument can play at a rate faster than the clock ticks. There are four ticks per beat in all songs tested. A slightly thicker dividing line between columns denotes a measure break. While all drum tracks include bass, snare, and hi-hat outputs, the number and types of drum outputs is unlimited in principle as long as the right sounds are available.

Recall that inputs are *spikes*; in the figures, their decays are depicted as decreasing darkness. In contrast, outputs represent *volumes*, wherein darker shading indicates higher volume. The main difference between inputs and outputs is that a single note in the input may straddle several columns during its decay. Outputs on the other hand are played as *separate* notes for every solid column. For an output drum to last for more than a single tick before the next drum attack, it must be followed by white (empty) columns.

5.1 Inputting Instrument Channels Alone

Figure 5 shows individuals from generations one and 11 generated for the folk song *Johnny Cope*. The relationship between the the bass, hi-hat, and snare and the three input channels is complex because each drum is related to all three inputs. Note however that the *instrumental* patterns in measures three and four are highly related though not identical. Slight differences exist between the piano pattern in measure three and measure four; this difference is reflected in the snare in both generations one and 11, which both slightly differ between the early parts of measures three and four. Thus, the drum pattern's subtle variation is *correlated* to the music because of their coupling, which evokes a subjective sense of appropriate style.

At measure 23, the song changes sharply by eliminating the piano part. Consequently, the CPPN outputs also diverge from their previous consistent motifs. This strongly coupled divergence that is carried both in the tune and in the drums creates a sense of purposeful coordination, again yielding a natural, sophisticated feel. In this way, the functional relationship represented by the CPPN causes the drums to follow the contours of the music seamlessly.

Generation 11, which evolved 12 additional connections and six additional nodes, reacts particularly strongly to the elimination of the piano by significantly altering its overall pattern. In generation one, the shift is less dramatic, showing how the user interactively pushed evolution towards a sharper transition over those ten generations. Generation 11 also elaborates on the snare, making it harder-hitting in the later measures than in earlier ones.

Results from generation 25 of *Oh! Susanna* are shown in figure 6. NEAT Drummer produces similarly natural and style-appropriate rhythms for this song as well, suggesting its generality. Because style is inherent in the original song's channels, it transfers seamlessly to the drum track without any need for explicit stylistic rules. The result is an entertaining sound that could be attached to the original instrumental tracks without raising suspicion.

It is interesting to *listen* to the songs with their generated drum tracks, which makes it possible to judge their subjective quality (a critical aspect of

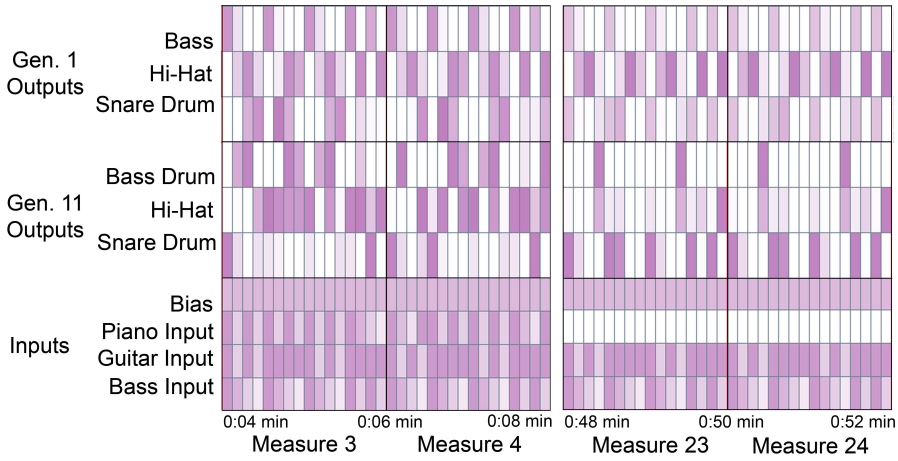


Figure 5: **Johnny Cope Results.** Results are depicted from two different generations at two different parts of the song. The inputs from the original song, which are always the same, are shown at bottom. Note the relationship between the inputs and the outputs, and between the first generation and the eleventh, which elaborates on the former. The motif in measures three and four is typical of the first part of the song until measure 23, when it switches to a different motif in both generations. Thus, the figure gives a sense of the two predominant drum riffs exhibited in both generations. The main conclusion is that the output is a function of the input that inherits its underlying style and character. (These tracks are available at `http://eplex.cs.ucf.edu/neatmusic/`)

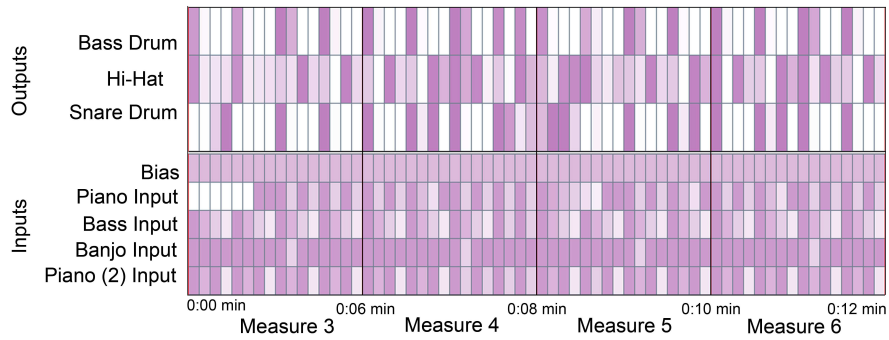


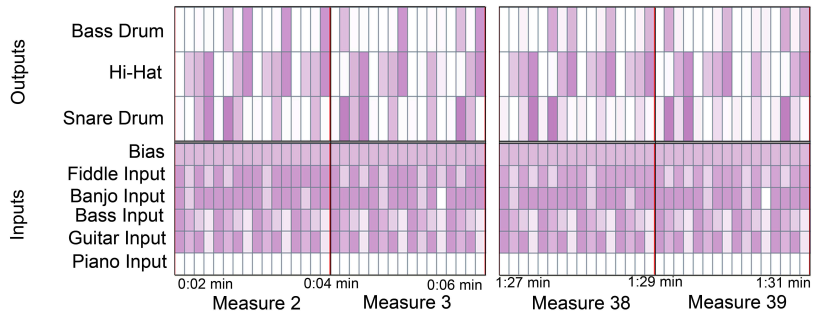
Figure 6: **Oh! Susanna Outputs.** This pattern from measures three through six of *Oh! Susanna* is from generation 25 of evolution. The network evolved 15 hidden nodes and 41 connections. Near the end of measure four is a particularly improvisational riff in the snare that transitions to measure five. This riff is caused by variation in the piano and other inputs at the same time. As with Johnny Cope, the drum pattern sounds natural and styled correctly for this upbeat song. (This track is available at `http://eplex.cs.ucf.edu/neatmusic/`)

musical appreciation). In the authors' experience (which the reader can also judge), the generated tracks sound natural and lack the usual "mechanical" quality of computer-generated music. Rather than repeating stock patterns, core motifs subtly vary and are interspersed with occasional unique flourishes. The personality of these variations is a byproduct of the personality that is implicit in the song itself, simply functionally transformed into a different local motif. This result further demonstrates that it is possible to *inherit* the natural character of one pattern by deriving another from it. Of course, the evolved song also in part reflects the tastes of the human user.

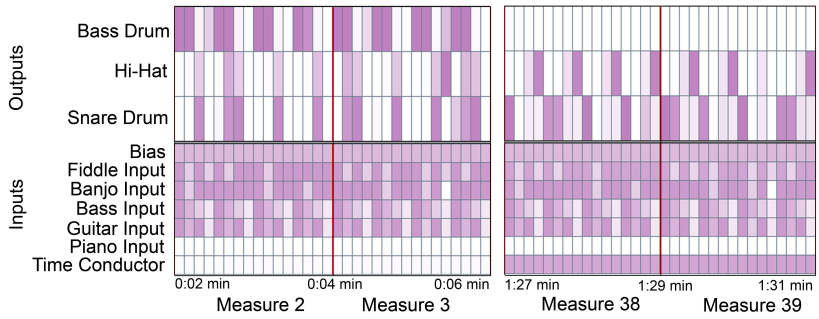
5.2 Inputting Instrument Channels and Conductors

Figure 7 highlights the effect of a conductor input on drum tracks produced for the song Oh! Dem Golden Slippers, which has a very similar beginning and end; all the measures in these parts are similar. Thus the question is whether a conductor can introduce a sense of progression into the drums even though the song itself undergoes little discernable progression between the start and finish.

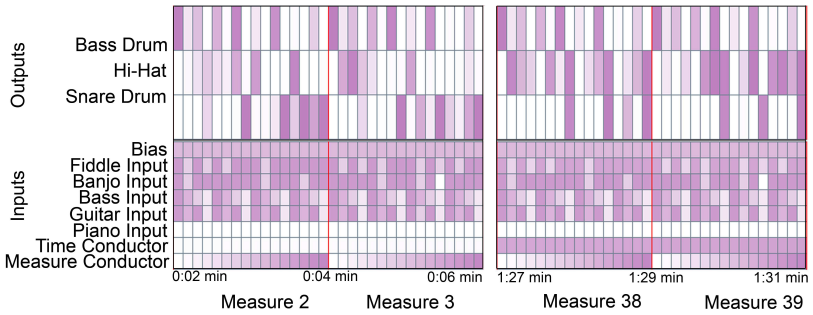
Figure 7(a) shows example drum output for this song *without* any additional conductor. Thus, with only the song's channels as inputs, the resultant drum pattern is also highly repetitive; the pattern in measures two and three is largely preserved much later in measures 38 and 39 (figure 7a). Yet when a *position-in-song* conductor (figure 4a) is added as an input, the difference in drum patterns between measures two and three and measures 38 and 39 is dramatic, showing the powerful effect of the simple conductor (figure 7b). Nevertheless, even



(a) Without Conductor



(b) With Position-in-song Conductor



(c) With Position-in-song *and* Position-in-measure Conductors

Figure 7: **Oh! Dem Golden Slippers with and without Conductors** Output drum patterns are shown for the song in one case when no conductor is input (a) and in the other where a *position-in-song* conductor is input (b, shown at bottom). The difference in resultant drum patterns shows that the conductor imposes a temporal progression on the drum track that does not derive from the structure of the song itself, demonstrating the power of conductors to subtly shape the structure of music. Finally, when conductors indicating *both* position-in-song and position-in-measure are input simultaneously (c), progression is enhanced both throughout the song and within each measure. (These tracks is available at `http://eplex.cs.ucf.edu/neatmusic/`)

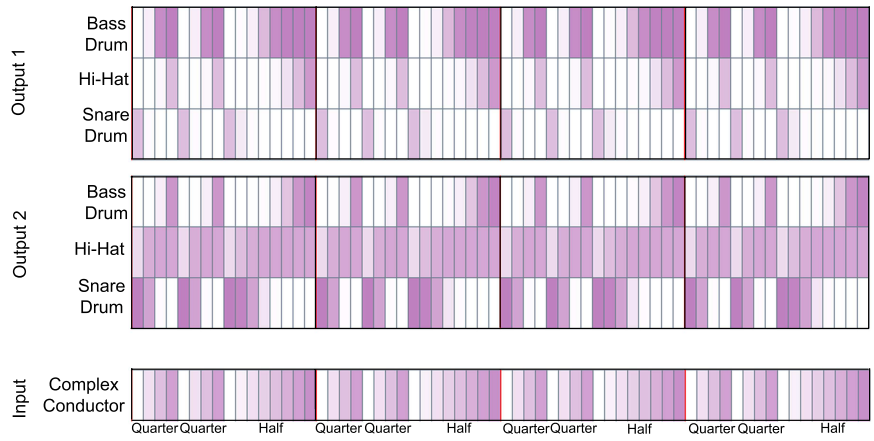


Figure 8: **Complex Conductor.** The conductor, which follows the pattern *quarter quarter half*, is shown at bottom. Two three-part drum tracks that are functions this conductor are shown above it. While both drum tracks are different, they are also both constrained by the underlying motif of the conductor. (These tracks are available at `http://eplex.cs.ucf.edu/neatmusic/`)

though the drum pattern exhibits a sharply different motif at these two similar parts of the song, it sounds appropriate and sophisticated in both parts because it is a function of *both* the conductor and the instrument channels. Thus it is a seamless variation on both influences simultaneously.

It is also possible to combine multiple conductors to affect the structure of the output in more than one way. Figure 7(c) shows the impact of inputting both the time in the song (figure 4a) and the time in the measure (figure 4b) together. The result is that not only do the later drum patterns differ from the earlier ones, but the interior of each measure varies in part independently of the instrumental scaffold. This effect is subtle because there are five instrumental tracks already influencing the pattern in each measure. However, closely comparing the drum measure patterns in 7(c) to 7(a) and 7(b) does reveal a discernable difference.

Finally, figure 8 isolates the effect of a single complex conductor. The aim is to show explicitly how the output of the CPPN is influenced by the incoming conductor, which expresses the same *quarter quarter half* pattern as in figure 3. Thus, the outputs of two CPPNs that both take the same conductor are displayed for comparison. The main result is that the patterns of the two three-part tracks are both closely tied to the pattern of the conductor, wherein two short events are always followed by a long one. Yet, within that framework, the patterns nevertheless differ significantly, illustrating the idea that a conductor is an implicit guide above which the pattern is realized, even if there is no explicit song at all.

The next section exhibits the evolved CPPNs that produce the drum tracks in this section.

5.3 Evolved CPPNs

Figure 9 shows the CPPNs that were evolved for each of the evolved drum tracks in Sections 5.1 and 5.2. These networks range in complexity from 1 to 15 hidden nodes. Interestingly, the subjective *quality* of the accompaniment does not seem to correlate to the complexity of the network. This perception makes sense because the functional relationship to the original instrument channels guarantees a tight coordination between drums and instruments. Thus creating a plausible coordination does not require significant complexity. Furthermore, if the underlying instrument channels themselves embody complex motifs and progressions, then the drums *inherit* the same complexity even if the CPPN that relates them is not itself complex.

What CPPN complexity affords, rather, is a more complex relationship that is realized through more elaborate covariation. This subjective effect is subtle yet perceptible, suggesting that more sophisticated compositions may suggest to the human ear the complexity of the function relating their parts.

Yet the most important conclusion is that complexity is not essential to the CPPN that relates one part to another because the complexity need only exist originally in the preexisting parts. To a large extent, that original complexity is transferred through the CPPN to any affiliated drum pattern.

The next section presents the results from the comparative experiments.

6 Comparison Results

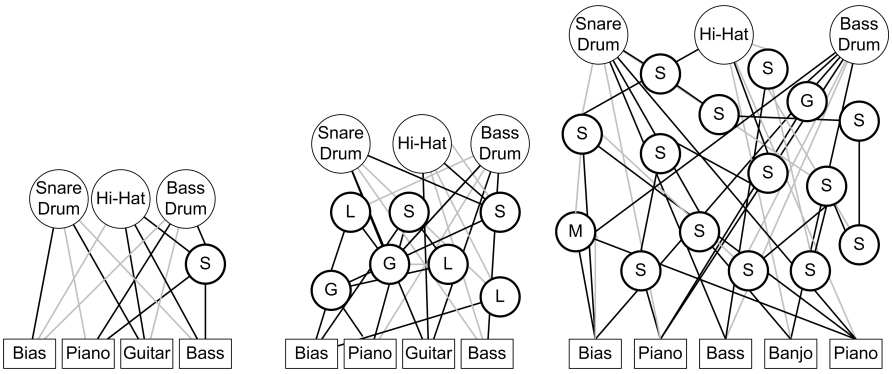
While the results in Section 5 establish the quality of the tracks produced through scaffolding and the power of conductors to shape the output pattern, the question remains what is lost if the scaffold is not provided, as in prior automated music generation techniques. Can similar accompaniment be produced without a scaffold? This section validates the role of the scaffold by answering this question.

Typical results reported in this section can be heard at `http://eplex.cs.ucf.edu/neatmusic/`.

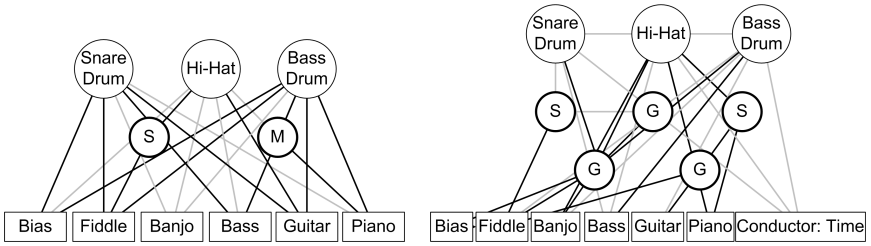
6.1 Interactive Evolution Without Scaffolding

As described in Section 4.2, in the first set of comparisons, recurrent ANNs with only a bias input and feedforward ANNs with position-in-song as input were evolved interactively with no other inputs to accompany Johnny Cope. Five 30-generation runs of both configurations were completed.

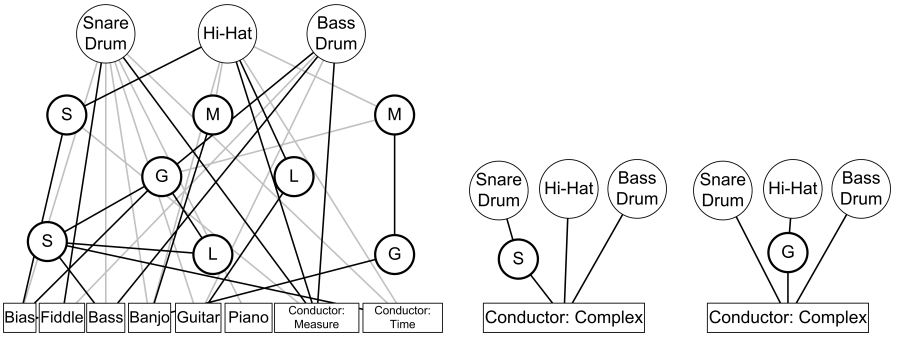
Figure 10 shows typical best results from these runs. The best results from each set reveal a distinct difference between feedforward functions of position-in-song and evolved recurrent networks. The feedforward patterns never develop beyond monotonous unbroken gradients that gradually vary from loud to soft, and sometimes back again (figure 10a). These gradients often span the length of the song, or a large extent of it, and do not respect the measure structure. Thus, overall, position-in-song alone is not enough to allow interactive evolution to



(a) Johnny Cope Gen. 1 (b) Johnny Cope Gen. 11 (c) Oh! Susanna

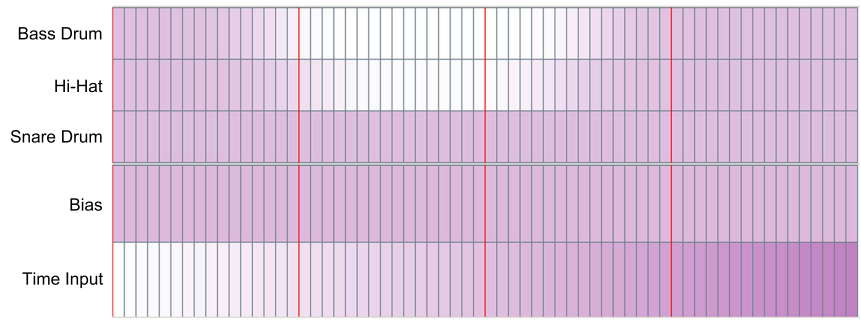


(d) Oh! Dem Golden without Conductors (e) Oh! Dem Golden with the Time Conductor

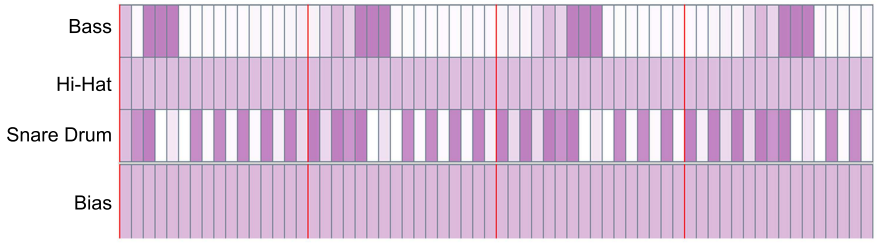


(f) Oh! Dem Golden with Measure and Time (g) Quarter Quarter Half Conductor (h) Quarter Quarter Half Conductor

Figure 9: **CPPN Drum Track Generators.** The evolved CPPNs that produce every drum track shown in Sections 5.1 and 5.2 are depicted. While the complexities vary, the quality of the output is similar because each produces a function of a preexisting song, thereby inheriting its qualities. *Activation functions are denoted by S for sigmoid, M for multiplication, G for Gaussian, and L for linear.*



(a) Typical Best Feedforward



(b) Typical Best Recurrent

Figure 10: **Typical Best Results from Interactive Evolution Without Scaffolding.** Best results from both types of representations tested are depicted. The feedforward network only inputs position-in-song and produces unremarkable gradient patterns that do not follow the structure of music nor the Johnny Cope song. The recurrent network produces repeating motifs, but they are not synchronized with the measure and they do not vary with the song. In this way, removing the scaffold removes a major advantage of NEAT Drummer.

compete with evolving networks with a better scaffold. This result makes sense because the only input is the position in the song, so the only way to develop a significant number of changes is to add many hidden nodes, which would take far longer than 30 generations. Also, because CPPNs have no knowledge of when measures begin or end, the changes that do occur are difficult to evolve to align with the measure structure.

In contrast, patterns interactively evolved with recurrent networks *do* display more complexity and more frequent changes over time (figure 10b). Because feedback can lead to complex oscillations without the need for many hidden nodes, recurrent networks are better suited to producing complex variation early in evolution. However, the drum patterns are difficult to evolve to align with the contours of the music itself because the recurrent network is unaware of the music without the scaffold. Furthermore, these networks suffer from the same problem with measure structure as the feedforward networks: Even though

some motifs repeat, they repeat at haphazard times relative to each measure, producing a disorganized aesthetic. For example, the bass drum in figure 10(b) is hit several times at the start of the first measure, but by the fourth measure, this motif has moved to the middle of the measure.

The overall result is that while the recurrent networks produce more complexity, the patterns evolved by both networks are not synchronized with Johnny Cope and therefore sound disjointed, highlighting the critical role of the scaffold in tethering the accompaniment to the song itself.

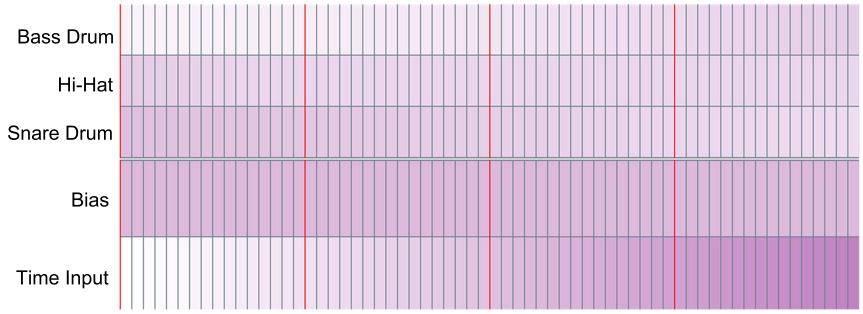
6.2 Targeted Evolution Without Scaffolding

In this experiment, the same two types of networks as in the previous section, i.e. one recurrent and one feedforward with position-in-song input, were evolved to match a *target* (figure 11c), which is a rhythm for Johnny Cope output by NEAT Drummer *with* the scaffold in the 11th generation. Clearly, the scaffold provides an advantage, but the question addressed by this experiment is how hard it is without the scaffold to approximate the same output even when the precise target rhythm is known a priori. Because it is also known that the target rhythm took exactly 11 generations to evolve when the scaffold was provided, the number of generations it takes these variant networks to produce the same output can be compared. Each variant attempted to match the target in 20 separate runs.

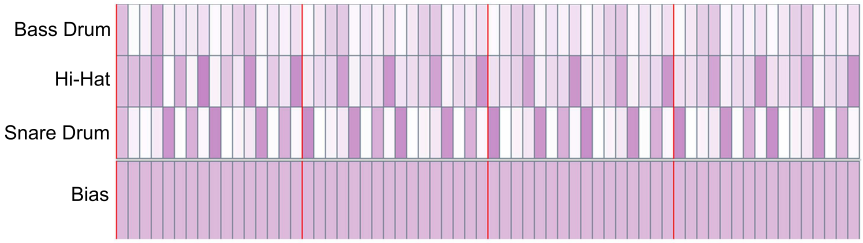
Figure 11 shows typical best results from these runs after $1,600$ NEAT generations with a population size of 100. It turns out that the feedforward network suffered similar problems breaking away from simple gradients as in interactive evolution. While such a network theoretically *could* approximate the target pattern, it always became trapped on a local optimum because it is easy to reach a fairly high fitness simply by approximating the general loudness of drums over large contiguous periods of time. In other words, instead of attempting to discover every individual beat, it discovers their average energy and paints that energy level across large swaths of time. Thus this type of network is demonstrably ill-suited to producing such temporal patterns, either through interactive evolution or target-based evolution.

However, interestingly, unlike in the 30-generation interactive experiment, after 1600 generations the recurrent network typically produces a repeating pattern that *does* synchronize with the measure structure. Thus one conclusion is that recurrent networks can learn musical structure given sufficient time. However, unlike the target pattern, which displays distinct variations (e.g. in the latter half of figure 11c), the pattern produced by the best recurrent networks tend to repeat the same measure pattern throughout the song after the first generation (figure 11b). Also, this repeating motif is only vaguely reminiscent of the target, which is likely because the recurrent network has trouble producing the subtle repetition with variation that the target inherited from its scaffold when it was evolved.

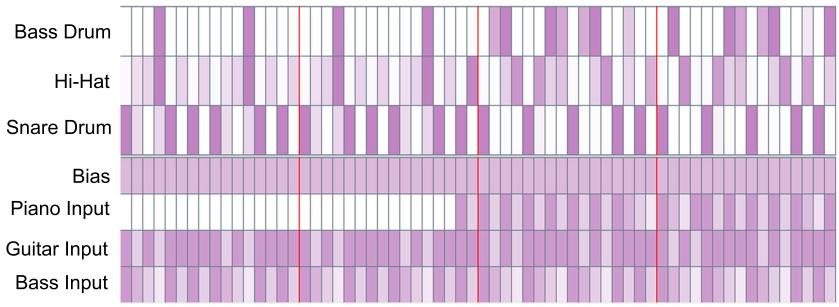
Because the feedforward results were disappointing, a third set of 20 runs was attempted with a feedforward network that receives *both* position-in-song



(a) Typical Feedforward Champion



(b) Typical Recurrent Champion



(c) Target Pattern

Figure 11: **Typical Best Results from Target-based Evolution Without Scaffolding.** Feedforward (a) and recurrent (b) results are depicted and the target pattern is shown in (c). The aim was to match the target. As this figure shows, neither variant successfully matched the target (which was evolved in NEAT Drummer in 11 generations) after 1,600 generations, although the recurrent variant evolved more complex patterns. This result confirms again the importance of the scaffold.

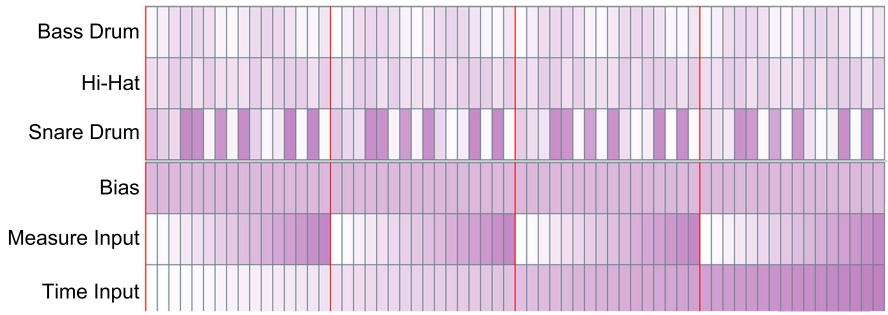


Figure 12: **Typical Result from Target-based Evolution with Position-in-song and Position-in-measure Inputs.** The improvement in pattern complexity, and the adherence to measure structure, are apparent in comparison to figure 11a. By providing position-in-measure as input, evolution can easily produce patterns that follow the timing of measures, demonstrating the power of conductor inputs. However, without the scaffolding inputs from Johnny Cope, the drum pattern still does not match the target despite its regular structure.

and position-in-measure conductors as input. The idea is to relieve the network of the need to discover the measure structure of music on its own, exploiting the power of conductors. In fact, as figure 12 shows, providing position-in-measure typically dramatically improves the complexity of the output and allows it to break out of the local optima that trap such networks without position-in-measure. Indeed, the feedforward output resembles the output of the recurrent network and respects measure structure, demonstrating the contribution of the additional conductor. Yet because even such conductors do not contain the same song-specific information as the scaffold, like the recurrent network, the output pattern is only marginally reminiscent of the actual target pattern, and is also more repetitive.

Thus, after 1,600 generations, none of the variant networks are able to successfully match a target pattern that was discovered in only 11 generations. Figure 13 summarizes this result by depicting fitness over time in the three variants, each averaged over 20 runs. Whereas a fitness of 1.0 denotes a perfect match, none of the variants reach a fitness of even 0.7. Interestingly, despite the apparent aesthetic inferiority of the feedforward network with only position-in-song, on average its fitness approaches that of the other two variants, demonstrating why local optima characterized by smooth volume gradients attract it.

Although their final fitness levels are not far apart, the differences between some of the variants are significant. In particular, recurrent networks produce significantly higher fitness than position-in-song alone throughout the run ($p < 0.05$), and feedforward with the position-in-measure input outperforms feedforward without it ($p < 0.05$). However, interestingly, by the end of each run, recurrent networks on average are *not* significantly better than feedfor-

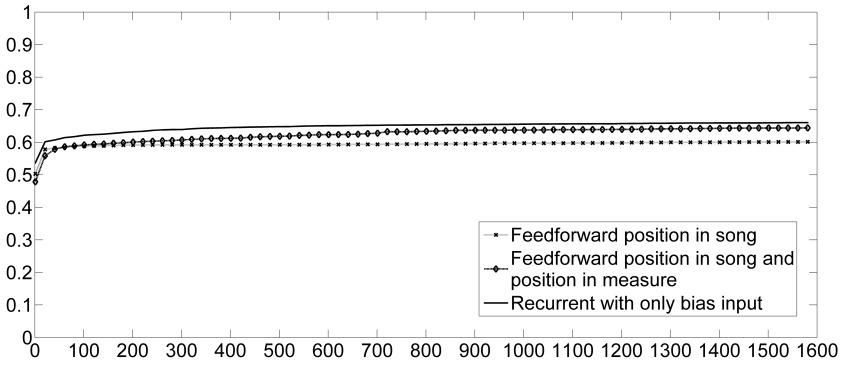


Figure 13: **Fitness Over Time of the Three Variants in Target-based Evolution.** The increase in fitness over 1,600 generations of evolving the three variant representations is shown, averaged over 20 runs for each. A perfect fitness of 1.0 would mean the target is matched perfectly. None of the variants exceed 0.7 fitness because the search is too unconstrained without the scaffold.

ward with position-in-measure, showing that feedforward networks can match the performance of recurrent networks on this task when provided information on the structure of music. However, most significantly, none of the variants can produce a pattern close to the target within 1,600 generations.

The main conclusion from both interactive and target-based comparisons is thus that the scaffold provides critical infrastructure. In effect, it constrains the search to patterns that relate to the original song. If accompaniment is to be evolved for an existing song, inputs from that song should be provided as a scaffold. Without such context, the accompaniment becomes decoupled from the song regardless of the representation.

A further result is that conductors make it easier to discover patterns that respect musical structure. While the recurrent network does eventually discover a measure-synchronized motif in the target-based runs, it takes hundreds of generations to achieve such synchrony. On the other hand, when time-in-measure is provided as a conductor, measure structure is respected from the start.

Overall, this set of experiments confirms the contribution of the scaffold and suggests that it should be a standard facet of any network-based attempt to generate musical accompaniment.

7 Discussion

While the sheer size of a composition containing multiple interrelated tracks suggests its complexity, the results with NEAT Drummer show that by encoding some parts as functions of others, much of the apparent complexity is removed. While each drum track contains hundreds of drum strikes over several minutes,

the networks that represent them contain on average only 22 connections. The most salient feature of drum tracks generated in this way is that they *sound* natural, hinting that the functional relationship may reflect a realistic aspect of human musical creativity.

7.1 Implications of Scaffolding

The idea of scaffolding, i.e. deriving several parts from a preexisting pattern, means that the most profound effort in musical creativity can be largely concentrated on a relatively small part of the overall composition, which can then provide a scaffold for other parts. The interrelationships among the different instruments of a song can be expressed as functions of one or more scaffold tracks, which is the approach taken in this paper. While effective, it is interesting that the direction of the relationship could also go the other way: Melody and harmony can by expressed as a function of rhythm. In fact, harmony can also be a function of melody and vice versa. Thus there is no apparent *essential* starting point, though some may be more natural than others depending on the context. Nevertheless, it is clear that *something* must form the scaffold from which all other accompaniment is derived. Thus, as long as some seed is introduced, the accompaniment can follow smoothly.

The main contribution of this paper is thus to advance automated music generation by introducing an effective method for representing some parts of a song with respect to others in a connectionist framework. This approach significantly simplifies the problem of constraining accompaniment appropriately. Nevertheless, of course, different *styles* of accompaniment may be more or less difficult to discover, even with the scaffold. For example, can convincing jazz walking bass be generated even in the context of other jazz instrumental tracks? Certainly it is *possible* that the interactive evolution process can discover a function that expresses a particular style, yet the likelihood of such a discovery depends on to what extent the style is already embodied by the existing scaffold. The extent to which the scaffold contains essential stylistic cues, combined with the complexity of the function that would create the right style in the absence of such cues, determines the difficulty of the discovery. Thus this work does not diminish the considerable human effort required to acquire specialized styles of accompaniment.

Another intriguing possibility is that specific conductors can be developed that constrain output to a desired style. Thus, while discovering a style based on the scaffold alone may be difficult in some cases, providing *both* the scaffold and carefully chosen conductors can potentially simplify the search, just as conductors in this paper convey a priori measure and beat structure.

Interestingly, once a CPPN is found that yields a particular accompaniment style, it can potentially be *reused* with other tracks. Thus, once discovered, stylistic accompaniment may be transferable, which is an important topic for future research.

Overall, then, while some styles of accompaniment are probably harder to discover than others, the scaffold reduces the space that must be searched.

Nevertheless, although automated music generation may benefit from this principle, it still does not solve the fundamental problem of generating the scaffold itself. What kind of process can create the initial pattern? Interestingly, it is possible that even an individual instrumental part can be generated from an even more abstract underlying scaffold, i.e. one that is never actually *heard*, like the conductors in this paper. These abstract patterns represent musical structures below the level of the explicit notes and pauses. Rather, they are the shape of the fabric upon which such notes are woven. An interesting hypothesis is that human composers and songwriters construct at a cognitive level such hidden "conductors" *before* the salient musical pattern emerges as notes and rhythms. It may be difficult to discern, but several simple underlying functional motifs that cannot be expressed in musical notation may underlie apparently richly textured musical masterpieces. Perhaps this hidden factor plays a role in our appreciation of music; when the many threads of a symphony stand out in their synchronized majesty, perhaps the brain is appreciating the simple hidden scaffold that unifies them in purpose.

Another complementary possibility is that even a long serial progression of notes is actually a series of motifs that are functions of each other. This view suggests that a scaffold or conductor underlying a long melody may itself be short and compact. These considerations lead to the philosophical question of how little is necessary to encode a "human essence" that functionally-related parts can inherit. Perhaps only a very simple and short hidden function is all that is essential to the subsequent unfolding of a symphony.

The size of the smallest necessary seed is practically important because the smaller it is, the easier it will be in the future to create entire compositions automatically. For example, if an entire complicated melody can be generated from a simple initial function, and the rhythm and harmony can be generated from the melody, then a future system might require a human to merely suggest the barest motif, such as a gradual attack followed by several step-wise decays, and from that point elaborate an entire composition through scaffolding.

Another important ingredient of NEAT Drummer that is not automated is the user's input. The product of a NEAT Drummer session thus in part reflects the creativity of the *user*, and not just the search algorithm. It is an interesting question whether the user can be entirely eliminated, allowing the computer to compose completely on its own. However, while NEAT Drummer does not eliminate the need for human input, what it does eliminate is the need for human *expertise* by shifting the creative focus from composition to opinion (i.e. what sounds the best). In this way, a significant obstacle to widespread, high-quality musical creativity is removed.

Thus the promise of this work is that it opens a promising new avenue to computer-generated music that raises interesting questions about how music is encoded and generated by humans.

7.2 The Role of Recurrence in Connectionist Music

The experiments comparing recurrent networks to scaffolded CPPNs in Section 6 yield interesting insight into the capabilities and limitations of recurrent neural networks applied to generating musical accompaniment. In particular, recurrent ANNs do produce complex motifs, but it is difficult to synchronize them to musical structure. While a CPPN with a position-in-measure conductor can right away produce patterns that respect measures, it takes hundreds of generations to achieve the same with recurrent networks, which is too long for a human performing interactive evolution.

This result raises the question of whether it is biologically realistic in connectionist models of music generation to rely upon recurrence as the main mechanism of musical encoding. It is also plausible that the brain stores music in part as simple functions that are layered and composed one upon another. It is notable that evolving a recurrent network (figure 11b) *and* a feedforward network that is a function of both position-in-song and position-in-measure (figure 12) produced results of similar quality. Thus the question remains open whether the best infrastructure for generating temporal patterns is recurrence, or whether it is two simple timing signals upon which feedforward functions can be built. It may also be a combination of the two.

8 Conclusion

This paper argued that the reason human musicians can improvise and compose vast and complex accompaniments almost instantaneously is that they are in effect generating a simple function that relates one instrument's part to another's. This idea was implemented in a program called NEAT Drummer that generates novel drum tracks for existing MIDI songs. Furthermore, the idea of a *conductor*, i.e. a simple hidden function that affects the overall pattern of music, was introduced. The results demonstrated the viability of this model of musical creativity, producing drum tracks that tightly follow the contours of real songs yet still produce nontrivial accompaniment. The conductors seamlessly introduced variational motifs over and above those already in the existing song structure, creating a new way for humans with little musical expertise to control the overall structure of a song. The main conclusion is that a significant portion of musical creativity may be explained by the functional relationships between the different parts of a song.

Acknowledgments

Special thanks to Michael Rosario for his prior work at the University of Central Florida creating the software infrastructure for NEAT Drummer. Special thanks also to Barry Taylor for granting special permission to utilize his own MIDI productions of folk music in this work. Barry Taylor originally sequenced Johnny Cope, Oh! Susanna, and Oh! Dem Golden Slippers (all without percussion),

which are the three songs for which drum tracks were generated in Sections 5 and 6. This research was supported in part by NSF grants IIS-REU: 0647120 and IIS-REU 0647018.

References

Barrett, F. J. (1998). Coda: Creativity and improvisation in jazz and organizations: Implications for organizational learning. *Organization Science*, 9(5):605–622. Special Issue: Jazz Improvisation and Organizing.

Berliner, P. F. (1994). *Thinking in Jazz: The Infinite Art of Improvisation.* (CSE) Chicago Studies in Ethnomusicology. The University of Chicago Press, Chicago.

Biles, J. A. (1999). Life with genjam: interacting with a musical iga, systems, man, and cybernetics. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 652–656.

Biles, J. A. (2007). Evolutionary computation for musical tasks. In Miranda, E. R. and Biles, J. A., editors, *Evolutionary Computer Music*, chapter 2, pages 28–51. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Chen, C.-C. and Miikkulainen, R. (2001). Creating melodies with evolving recurrent neural networks. In *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, pages 2241–2246.

Dawkins, R. (1986). *The Blind Watchmaker.* Longman, Essex, U.K.

de Mantaras, R. L. and Arcos, J. L. (2002). Ai and music from composition to expressive performance. *AI Mag.*, 23(3):43–57.

Deutsch, O. E. (1965). *Mozart: A Documentary Biography.* Standford University Press, Standford, California. 59.

Dolson, M. (1989). Machine tongues xii: Neural networks. *Computer Music Journal*, 13(3):28–40.

Furuta, H., Maeda, K., and E.Watanabe (1995). Application of genetic algorithm to aesthetic design of bridge structures. *Microcomput. Civil Eng*, 10(6):415–421.

Gomez, F. and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *IJCAI-99*, pages 1356–1361, KAUF-ADDR. KAUF.

Husbands, P., Copley, P., Eldridge, A., and Mandelis, J. (2007). An introduction to evolutionary computing for musicians. In Miranda, E. R. and Biles, J. A., editors, *Evolutionary Computer Music*, chapter 1, pages 1–27. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Hymer, S. (1990). On inspiration. In Stern, E. M., editor, *Psychotherapy and the Widowed Patient*. The Haworth Press, Inc., New Rochelle, New York.

Johanson, B. and Poli, R. (1998). Gp-music: An interactive genetic programming system for music generation with automated fitness raters. In *Proceedings of the Third Annual Conference: Genetic Programming*, pages 181–186.

Katz, P. and Longden, S. (1983). The jam session: A study of spontaneous group process. In Middleman, R., editor, *Activities and Action in Groupwork*, chapter 3, pages 37–52. The Haworth Press, Inc, Binghamton, NY.

Kuriyama, K. and Terano, T. (1997). Interactive story composition support by genetic algorithms. In *World Conf. Artificial Intelligence in Education*, page 615617, Kobe, Japan.

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development parts I and II. *Journal of Theoretical Biology*, 18:280–299 and 300–315.

McCormack, J. (2005). Open problems in evolutionary music and art. In *Proceedings of Applications of Evolutionary Computing, (EvoMUSART 2005)*, volume 3449 of *Lecture Notes in Computer Science*, pages 428–436, Berlin, Germany. Springer Verlag.

Nelson, G. L. (1993). Sonomorphs: An application of genetic algorithms to growth and development of musical organisms. In *4th Biennial Art and Technology Symp.*, pages 155–169.

Oliver, M. (2006). On inspiration. *Contemporary Music Review*, 25(5/6):457 – 459.

Onisawa, T., Takizawa, W., and Unehara, M. (2000). Composition of melody reflecting users feeling. *IEEE Int. Conf. Industrial Electronics, Control and Instrumentation*, page 27382743.

Ruppel, R. R. (1998). *Gottfried Keller and His Critics: A Case Study in Scholarly Criticism*. Camden House, Columbia, SC.

Saravanan, N. and Fogel, D. B. (1995). Evolving neural control systems. *IEEE Expert*, pages 23–27.

Sato, T. and Hagiwara, M. (1999). Tool creating support system using evolutionary techniques. *Faji Shisutemu Shinpojiumu Koen Ronbunshu*, 15:363366.

Schuller, G. (1968). *Early Jazz*. Oxford University Pres, New York.

Secretan, J., Beato, N., D'Ambrosio, D. B., Rodriguez, A., Campbell, A., and Stanley, K. O. (2008). Picbreeder: Evolving pictures collaboratively online. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1759–1768, New York, NY, USA. ACM.

Sims, K. (1991). Artificial evolution for computer graphics. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*, pages 319–328, New York, NY. ACM Press.

Stanley, K. O. (2006). Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*, Menlo Park, CA. AAAI Press.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127.

Stanley, K. O. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *JAIR*, 21:63–100.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

Todd, P. M. and Loy, D. G. (1991). *Music and Connectionism*. MIT Press, Cambridge, MA.

Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers*. Academic Press, London.

Todd, S. and Latham, W. (1999). *The Mutation and Growth of Art by Computers*, chapter 9, pages 221–250. Morgan Kaufmann, San Mateo, California.

Tokui, N. and Iba, H. (2000). Music composition with interactive evolutionary computation. In *Third International Conference on Generative Art*, pages 215–226, Milan Italy.

Weick, K. E. (1998). Introductory essay: Improvisation as a mindset for organizational analysis. *Organization Science*, 9(5):543–555. Special Issue: Jazz Improvisation and Organizing.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.