

NEAT Drummer: Interactive Evolutionary Computation for Drum Pattern Generation

Amy Kathryn Hoover and Kenneth O. Stanley

Abstract—A major challenge in computer generated music is breaking the barrier between musical novelty and musical quality. Typically, computer music generators produce either genre-specific patterns that lack innovation or patterns that are given too much freedom and lack cohesion. In an attempt to both constrain the musical search space and produce novel rhythms, a program called NEAT Drummer is introduced. NEAT Drummer evolves neural networks with the NeuroEvolution of Augmenting Topologies (NEAT) that produce compelling drum patterns. To constrain the musical search space, NEAT Drummer accepts a base rhythm or motif from the user and through Interactive Evolutionary Computation (IEC), complexifies that pattern with each successive generation. This paper discusses the concepts behind how NEAT Drummer understands and manipulates a base rhythm, which is either predefined by the user through a basic interface or defined by MIDI music file information.

Index Terms—computer generated music, computer generated rhythms, NeuroEvolution of Augmenting Topologies, NEAT, interactive evolutionary computation, IEC

I. INTRODUCTION

Music is an important part of today’s culture, and evidence suggests that of cultures existing in any time or place. Although many people enjoy listening music, the lack of time, musical experience, and creativity restricts many potential music composers. It is therefore not surprising that researchers from a variety of fields have attempted and are currently trying to automate the composition process.

A major problem that still exists in computer generated music is breaking the barrier that lies between musical novelty and musical quality. Although researchers have experimented with a wide variety of algorithmic techniques ranging from random number generators, rule based methods, generative grammars, fractals, and evolutionary computation, computer music generators typically produce either genre-specific patterns that lack innovation or patterns that are given too much freedom and lack cohesion.

In an attempt to both constrain the musical search space and produce novel rhythms, a program called NEAT Drummer is introduced. NEAT Drummer evolves neural networks with the NeuroEvolution of Augmenting Topologies (NEAT) that produce compelling drum patterns. NEAT Drummer evolves these drum patterns through the process of Interactive Evolutionary Computation (IEC), which provides a means for a user

to generate, mate, and evolve different rhythms that they have defined through the easily accessible interface. The user acts as the match-maker or fitness function by ignoring bad drum patterns and evolving the more promising rhythms.

Although the user of NEAT Drummer has more control over how the second generation and their progeny will sound, the only the specified inputs to the ANN determine how the first generation will sound. Since the subsequent generations are based off of the first generation, it is clear that inputs play a vital part in the type and quality of the rhythmic outputs. For this reason, this paper focuses on different types of inputs and methods for controlling their outputs. One type of input accepts a user defined motif that the ANN will use as a basis for composition. Another type of input breaks apart a MIDI music file and generates rhythms based off of the piece of the MIDI selected.

II. BACKGROUND

This section reviews computer generated music, Interactive Evolutionary Computation (IEC), and NeuroEvolution of Augmenting Topologies (NEAT) .

A. Computer Generated Music

Computer generated music, music created by a computer, began around 1956 with *Stochastic Music*. Although Iannis Xenakis used this term to describe his specific composition style that used the FORTRAN programming language to perform statistical calculations of note probabilities, this early random number generator form of composing typified computer generated music of the time. Lajaren Hiller and Leonard Isaacson’s *Illiad Suite for String Quartet*, which was fully composed by a computer, the University of Illinois’s ILLIAC, but not played until later with the development of sound synthesis on the ILLIAC II. Over time, as computer scientists and musicians alike began to realize the limitations of purely statistical analysis in music composition, they began looking at alternative compositional algorithms. These algorithms which were rule, grammatically, chaotically, or evolutionarily based enjoyed moderate levels of success as a novelty, but have yet to produce anything significant.

B. Interactive Evolutionary Computation

Interactive Evolutionary Computation (IEC) is a form of Evolutionary Computation (EC) that allows users to perform the selection of the fitness function [1]. This form of EC which was popularized by Richard Dawkins in his book *The Blind*

⁰Manuscript received July 20, 2007. This work was supported in part by the National Science Foundation IIS-REU-0647018, and IIS-REU-0647120.

Amy K. Hoover is with the School of EECS at the University of Central Florida, Orlando, FL 32816 (e-mail: ahoover@cs.ucf.edu)

Kenneth O. Stanley is with the School of EECS at the University of Central Florida, Orlando, FL 32816 (e-mail: kstanley@cs.ucf.edu)

Watchmaker, has seen the most activity in areas of research where the fitness function is either impossible or too difficult to calculate [2][1].

IEC was first applied to computer generated music in 1993 with the development of *Sonomorphs* [3]. Typical of most IEC programs, *Sonomorphs* started off with a random initial population. From there, the user picked a pair of individuals to evolve from, parents, although the number can vary in other IEC applications and created the next generation, children of these parents, based off of the parental encoding.

C. *NeuroEvolution of Augmenting Topologies*

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 [4], [5], [6], or arbitrary random-topology networks [7], [8], [9], [10], [11], [12], [13], NEAT is the first to begin evolution with a population of small, simple networks and complexify the network topology over generations, leading to increasingly sophisticated behavior. Compared to traditional reinforcement learning techniques, which predict the longterm reward for taking actions in different states [14], the recurrent networks that evolve in NEAT are robust in continuous domains and in domains that require memory, making many applications possible. This section briefly reviews the NEAT method; Stanley and Miikkulainen [15], [16] provide complete introductions. NEAT is based on three key principles.

First, in order 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. That way, genomes of different organizations and sizes stay compatible throughout evolution, solving the previously open problem of matching different topologies [17] in an evolving population. Second, NEAT speciates the population so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before competing with other niches in the population. NEAT uses the historical markings on genes to determine to which species different individuals belong.

Third, unlike other systems that evolve network topologies and weights [18], [10], [13], [11], [10], NEAT begins with a uniform 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. This process of complexification has important implications for search. While it may not be practical to find a solution in a high-dimensional space by searching in that space directly, it may be possible to find it by first searching in lower dimensional spaces and complexifying the best solutions into the high-dimensional space.

III. APPROACH

This section explains how the NEAT Drummer produces drum patterns using an ANN and the two types of inputs available in NEAT Drummer.

A. *Rhythm Generation with an ANN*

NEAT Drummer represents drum patterns through the discrete sampling of the underlying ANN over the length of the song. Each rectangular box in the NEAT Drummer drum pattern represents the value of the ANN of the particular output at a specific time. Darker shaded regions represent louder, more forceful notes whereas the lighter regions are meeker notes or some not even played at all. The arrangement of these outputs depends on the input structure.

Inputs to the NEAT Drummer ANN require certain musical elements such as time signature, the number of beats per measure, and the ticks per beat, where ticks per beat is the number of times the ANN is sampled in the time period of a beat. Time signature indicates the number of beats per measure when it is written in simple time, but ticks per beat must be either selected by the user or set to a default value. Also, from these two variables, the duration for the ranges of notes can be set. For example, 4/4 time indicates four quarter notes are in a measure and every quarter note counts as a beat such that there are four beats in a measure. Suppose ticks per beat is then assigned a value of 2. The ticks per beat and beats per measure can then be used to calculate the total number of ticks per measure. Then, it follows that there are eight ticks in a measure, and a quarter note and quarter rest span two ticks each.

As illustrated above, it is quite simple to find the duration of a quarter note or quarter rest given the time signature and the amount of ticks per beat. For the conductor to be able to interact with the ANN, however, it is not just necessary to know the duration of a quarter note, but also durations for every other type of note and rest. To find these values from the duration of a quarter note, the number of ticks must be doubled for a half note, quadrupled for a whole note, and so on. In the previous example then, a half note would last four ticks and a whole note would last eight ticks

Inputs operate through the concept of musical measure. 1 is an ANN interpretation of the measure progression throughout the song. At time 0, the song has just begun and continues to play through point time 16 where the first measure ends and continues on to the second. This illustration represents a song that spans the length of four measures, and shows how the four measures are input interpreted by the ANN.

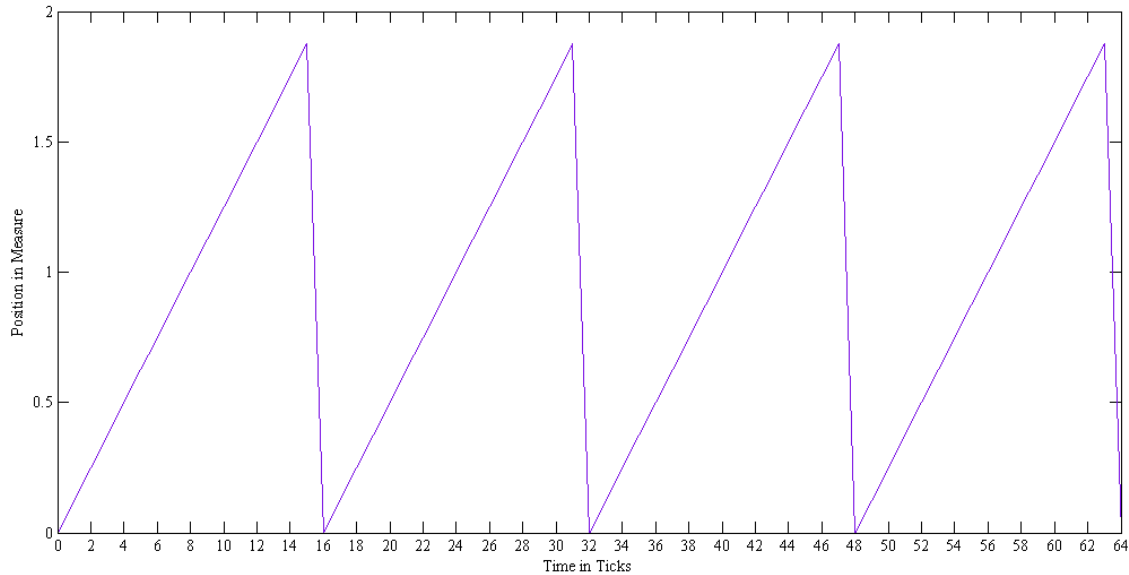


Fig. 1. Four measures as interpreted by the ANN.

B. Conductors: Inputs to the ANN

The *conductor*, much like an orchestral conductor, helps control the drum pattern but doesn't dictate it. Conductors act as links between the user, who supplies the ANN with a simple motif, and the ANN which interprets this pattern and relays this interpretation to the drums. Conductors provide a means for a person with relatively little musical experience to take a vague notion of a rhythm, or motif, and instructs NEAT Drummer to consider this motif when it creates drum patterns.

From the very first generation of individuals in the first drum pattern population to the final generation, the user directs the rhythmic evolution to create a drum pattern inspired by their own creation.

Conductors as the ANN sees them are a series of notes, rests, or a combination of notes and rests, where the duration of the notes and rests. The amount of ticks each note is held is determined by the user specified musical structure which is set before the generation of the rhythm can begin.

2 shows the same four measures of 1, but now with a conductor in 4/4 time on top of the first measure representation. This conductor is composed of two quarter notes and a half note where one quarter note is receiving four beats.

3 shows the same three measures of ??, but now with a conductor in 4/4 time on top of the first measure representation. This conductor is composed of two quarter notes and a half note where one quarter note is receiving four beats.

C. Rhythms from MIDI Files

MIDI inputs, like conductors, are a way for the user to communicate with the ANN. The user is able to pick a song, or part of a song, and send it to the network for interpretation. The network then returns a rhythm that corresponds to that MIDI which can be evolved interactively and can be played along side the MIDI, or stand alone.

Although it is necessary for the user to know a little bit about music to manipulate conductors since it uses the concept of quarter note, eighth notes, and so on, all the user has to do for the MIDI inputs is select a channel number or a specific part and instruct the network to use that part when generating rhythms. Instead of requesting that the user chooses a particular motif, which may or may not exist, MIDI inputs allows the user to pick which part of the MIDI to input to the ANN, e.g. horns, piano. To generate these rhythms, the program subtracts the original percussion from the base MIDI file and selectively parses the specified channel information.

IV. EXPERIMENTAL RESULTS

This section illustrates the different inputs and the types of drum patterns they produce.

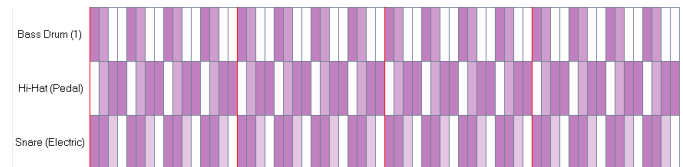


Fig. 4. Example of isolated conductor composed of four quarter notes

The conductor in 3 shows how the ANN can represent the idea of a quarter note, and illustrates how the network understands it. The conductor contains four quarter notes in 4/4 time were entered as an input and the output is pictured in 4. Recall that in 4/4 time, there are 4 quarter notes in a measure which is separated from the other measures by a line extending through all of the instruments, and that the quarter note is a beat. Since every quarter note is then 4 ticks, every four rectangles represent a quarter note. It's clear that the network

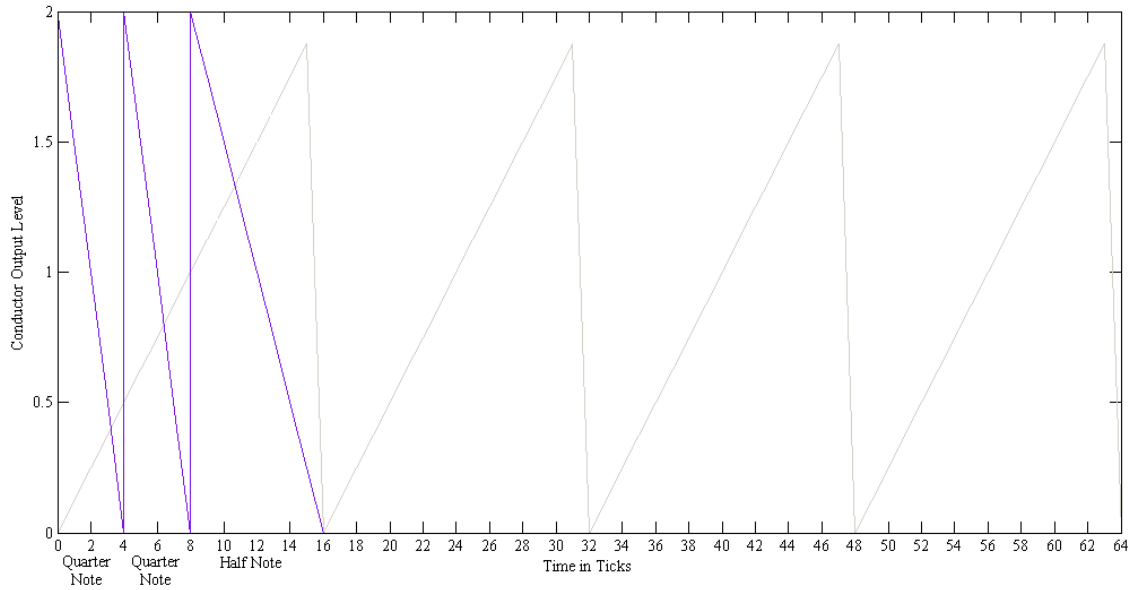


Fig. 2. Example of a conductor composed of two quarter notes and a half note represented in one measure visually, but repeated for every measure

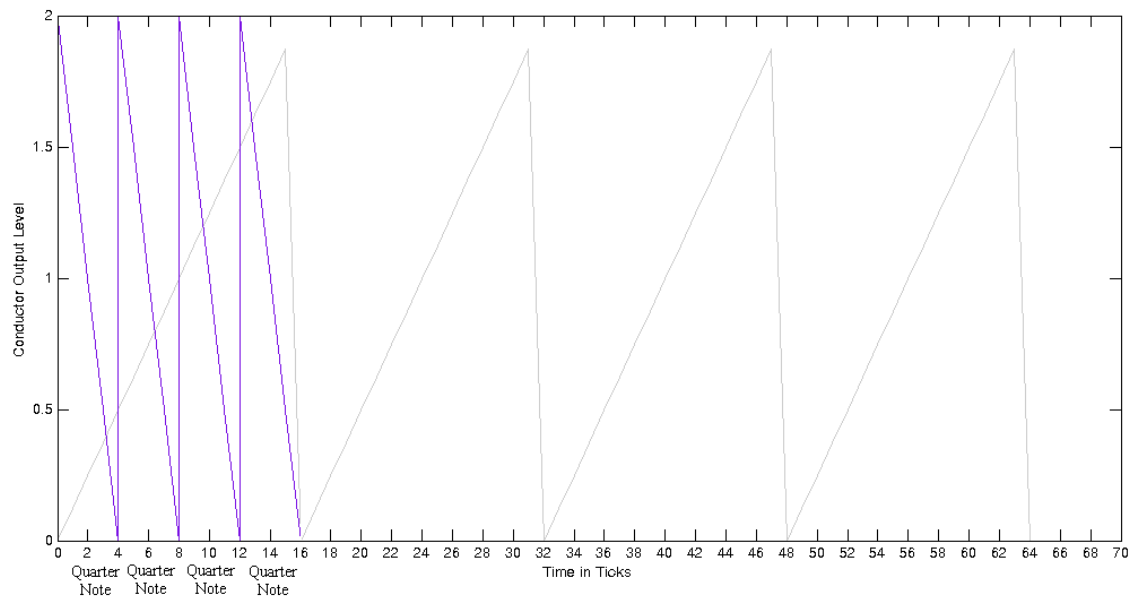


Fig. 3. A conductor spanning one measure over a total of four measures

is receiving four equal, sequential signals, each one of which is representing a quarter note. Each quarter note representation either hits and decays, or this is inverted from quiet notes to a final loud note.

The types of rhythms the ANN will produce, not the rhythms themselves, start to follow a pattern. One of the reasons for implementing conductors is to describe and classify these types of patterns. By examining the outputs from particular conductors, observations are made as to what conductors produce which classes of drum patterns.

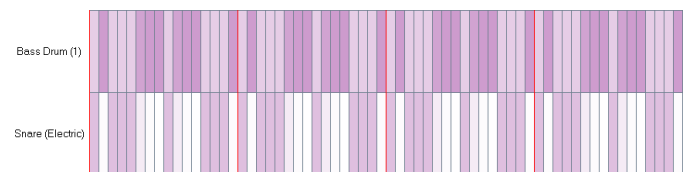


Fig. 5. Example of a rhythm with notes on the off beats

One observation in particular was that conductors with an

eight note and eighth rest combination could produce rhythms similar to those heard in Latin music and in blues. These syncopated rhythms, like the one pictured in IV all have a note on the offbeat where it is typically not expected.

V. CONCLUSIONS

Researchers in computer generated music are still looking for a way to make music that is structurally sound yet innovative. A variety of algorithms have been introduced to the problem, with marginal results at best. In this paper, NEAT Drummer was introduced as producing both novel and structured drum patterns. Experiments with the inputs of the ANN show that conductors can be classified at this preliminary stage into syncopated or not syncopated, a characteristic of Latin music or blues music. Thus, NEAT Drummer, a way to evolve rhythms that are both innovative and structured. Future research will focus further identifying constraining drum patterns to restrict the classes of drum patterns into genres.

ACKNOWLEDGMENT

Special thanks to Michael Rosario who created the initial framework for NEAT Drummer and for allowing me to continue developing his project, and to Dr. Michael Georgiopoulos, Dr. Georgios Anagnostopoulos, Dr. Mingyu Zhong, and Jimmy Secretan for their continued support over the course of the REU.

REFERENCES

- [1] H. Takagi, "Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.
- [2] R. Dawkins, *The Blind Watchmaker*. Norton, 1987.
- [3] G. L. Nelson, "Sonomorphs: An application of genetic algorithms to growth and development of musical organisms," in *4th Biennial Art and Technology Symp.*, March 1993, pp. 155–169.
- [4] F. Gomez and R. Miikkulainen, "Solving non-Markovian control tasks with neuroevolution," 1999, pp. 1356–1361.
- [5] N. Saravanan and D. B. Fogel, "Evolving neural control systems," *IEEE Expert*, pp. 23–27, June 1995.
- [6] A. Wieland, "Evolving neural network controllers for unstable systems," pp. 667–673.
- [7] H. Braun and J. Weisbrod, "Evolving feedforward neural networks," 1993.
- [8] J. C. F. Pujol and R. Poli, "Evolution of the topology and the weights of neural networks using genetic programming with a dual representation," School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK, Tech. Rep. CSRP-97-7, 1997.
- [9] J. C. Bongard and R. Pfeifer, "Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny," 2001, pp. 829–836.
- [10] F. Gruau, D. Whitley, and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. MIT Press, 1996, pp. 81–89.
- [11] B.-T. Zhang and H. Muhlenbein, "Evolving optimal neural networks using genetic algorithms with Occam's razor," vol. 7, pp. 199–220, 1993.
- [12] D. W. Opitz and J. W. Shavlik, "Connectionist theory refinement: Genetically searching the space of network topologies," vol. 6, pp. 177–209, 1997.
- [13] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
- [14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 1998.
- [15] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, pp. 99–127, 2002.
- [16] —, "Competitive coevolution through evolutionary complexification," vol. 21, pp. 63–100, 2004.
- [17] N. J. Radcliffe, "Genetic set recombination and its application to neural network topology optimization," *Neural computing and applications*, vol. 1, no. 1, pp. 67–90, 1993.
- [18] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," vol. 5, pp. 54–65, 1993.