Signal Complexification using Frequency Modulation and Neuroevolution

Brendan Moore and Wade Spires
School of Computer Science
Orlando, FL 32816-2362

December 21, 2006

Abstract

The automatic generation of dynamic or complex audio signals has a wide range of applications, from sound effects design to virtual instrument implementation. Techniques for the creation of content in these domains typically involve searching though large input spaces in order to find combinations that produce interesting results. These input spaces can be viewed as all the possible input signals, filter types, filter settings, etc., that might be used in order to generate a new signal. Our goal here is to automate this search process. This paper presents an audio generation system that combines simple FM Synthesis with a complexifiying neural network based on the NEAT system. Interesting results were found with our system that provide insight and motivation for future work on this topic.

1. Introduction

The creation of synthesised audio signals has a long history dating back as far as 1876 with the creation of Elisha Grays "Musical Telegraph"[3]. There have been countless examples of its uses. From the work of synthesizer designer Robert Moog to the film audio of sound designer Walter Murch, the need to effectively manipulate audio in order to produce new signals has a wide range of applications. Several techniques for audio synthesis have been developed. Some of the more popular methods include Subtractive synthesis, Additive synthesis, Wavetable synthesis, and Frequency Modulation synthesis. Each of these techniques produce sounds that are indicative of their methodology. For example by using Subtractive synthesis, which uses filters to subtract certain sine waves from a signal, we can approximate the sound of acoustic instruments like horns and strings. Whereas signals produced by the application of Frequency Modulation, a technique in which a Carrier signal is modulated by a Modulator signal, tend to be more *bell-like* or percussive. Despite their differences, each technique essentially relies on the principal of using one signal to modify another, either directly or indirectly. For the purposes of this paper we have decided to use Frequency Modulation as the chosen method of audio synthesis.

There is a natural analogy between the building blocks of Frequency Modulation synthesis and the structure of a neural network. It is easy to see each of the input nodes of a neural network as signal generators, or oscillators. We can then think of the link and link weights of the neural network as controlling which signals take part in modulation or filtering operations. Hidden nodes then become the location in which signals are *modulated* or filtered. The resulting *modified* signal is then passed on through the rest of the network to be modulated and filtered based on the topology (i.e., nodes, weights, and activation functions) of the network. Therefore, we would like to adopt a neural network design that allows for topological change. NEAT is one such system.

The next section gives a brief description of Frequency Modulation and describes some of the key features of the NEAT system. We then describe our approach to the combination of audio synthesis and NEAT. From there, we present several experiments including a test of the system using the XOR problem as a benchmark. Finally, we discuss out results and present ideas for future work.

1.1. Background

The concept of audio synthesis by way of Frequency Modulation (FM) was originally proposed by John Chowing [1]. The basic idea behind the FM Synthesis process is to modulate the frequency of one sine wave by another. The interaction of the two signals, one called a Carrier signal and the other called a Modulator signal will produce a new signal (see Figure 1) that tends to be more complex in comparison to its two parent signals. If the carrier signal is in the audible range, from 60 Hertz (Hz) to 20 Kilohertz (KHz), this complexity can be seen as a change in the tone of the signal or in this case, sound. The frequency relationship between the Carrier signal and the Modulator signal completely characterizes the new signal. If the Modulator frequency is an integer multiple of the Carrier signal, the new signal will be called harmonic.

For example, if the Carrier frequency is set to 440Hz and

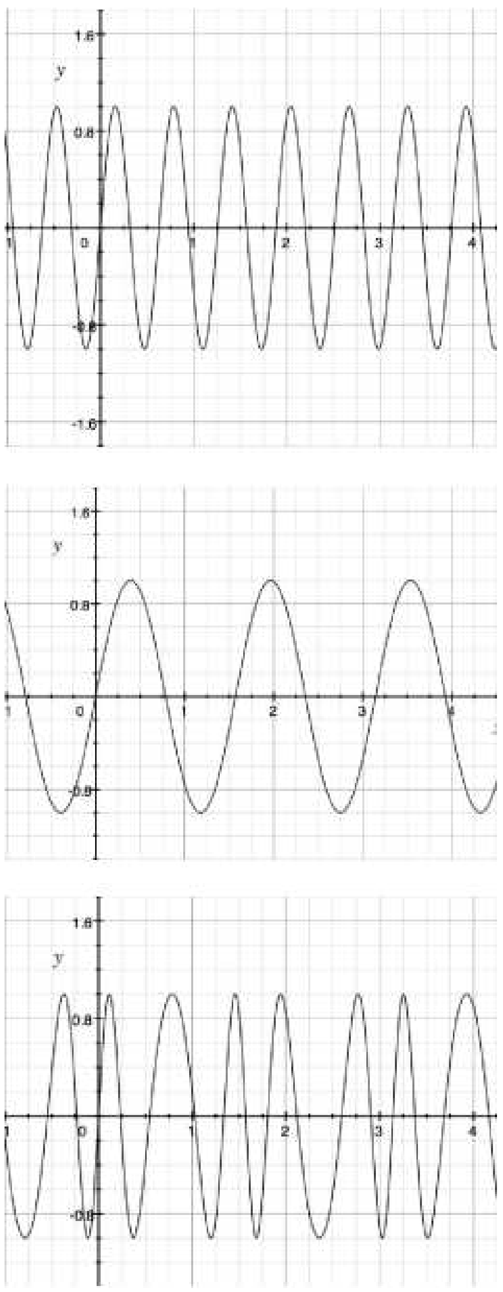


Figure 1: Carrier (top), Modulator (middle), and modulated (bottom) signals

the Modulator is set to 880Hz, we see that the Modulator frequency can be factored as 2·440Hz and is therefore a harmonic of the Carrier. In this example, the Modulator is actually called the second harmonic, and the Carrier is called the fundamental frequency. In terms of signals in the audible range, the resulting sound can be described as "glassy" or "hollow". If the Modulator signal is not an integer multiple of the Carrier signal, the new signal will be called inharmonic or non-harmonic. Using our previous example, if the Carrier frequency is again set to 440Hz and the Modulator is set to 883Hhz, we see that we cannot factor the Modulator frequency as a direct multiple of the Carrier frequency. Signals in the audible range having an in-harmonic relationship tend to be more percussive or bell-like. More formally, the general equation for Frequency Modulation can be described as

$$f(t) = sin(2 \cdot \phi \cdot \alpha \cdot t + I \cdot sin(2 \cdot \phi \cdot \beta \cdot t)) \qquad (1)$$

where ϕ is the phase of both the Carrier and the Modulator; α is the frequency of the Carrier signal; β is the frequency of the Modulator signal; t is time; and I is a scaling constant of the Modulator signal that determines its overall effect on the Carrier. Upon closer inspection of Equation 1 we see that we are really modulating the phase of the Carrier signal, i.e., by the addition of the Modulator signal. But this continuous change in phase, known as Phase Modulation will manifest itself as a continuous change in the frequency of the Carrier signal. Therefore, Frequency Modulation and Phase Modulation can be though of as equivalent operations noting that if the Phase Modulator is defined as

$$m(\alpha) = sin(\alpha),$$

then the Frequency Modulator is

$$f(\alpha) = \nabla m(\alpha) = cos(\alpha) = sin(\alpha - \frac{\pi}{2}).$$

This shows that Frequency Modulation is equivalent to a shift in the frequency used in Phase Modulation. With this brief introduction to FM Synthesis, we move our attention to the Neural Network component of this system.

The underlying neural network is an implementation of the NeuroEvolution of Augmenting Topologies (NEAT) [4] system. NEAT is a neural network design that employs the idea of *complexification* in order to evolve network weights and topologies in an effort to converge to some optimal solution space. There are three main components in the NEAT system:

1. **Crossover and mutation**—NEAT uses a structured process for genetic crossover and mutation. This is accomplished though *Historical Markers.* By assigning the same marker to equivalent genes the history of the genes can be tracked and therefore like genes can be aligned in order to mate two Genomes correctly.

2. **Speciation**—*Speciation* is used to protect innovation. NEAT facilitates innovation by forcing Genomes that differ by some set amount to be segregated into their own Species. By doing so, the new Genomes are given a chance to compete and innovate without being compared to Genomes that might be better *only* because of their age. This incubates the new Genomes, protecting them from being lost due to premature truncation.

3. **Growth from minimal structure**—Rather than beginning optimization using networks of high dimension NEAT begins the process with simple uniform topologies, i.e., networks of small dimension. If structure is added, it is only kept if it improves the overall performance of the network. This way NEAT only searches in the minimal space to converge to a solution.

2. Approach

The first step in the construction of an audio synthesis system is the generation of the basic signals used in the system. The Synthesis Toolkit [2], an open source signal processing and algorithmic synthesis library, was used for building audio synthesis and processing. Since audio signals can be decomposed via a Fourier transform into their component sine waves, we chose to start the creation of our signals with sine waves. The Synthesis Toolkit provides functionality for the creation of sine wave lookup tables which became the inputs to our neural net. For each input node in the network, a sine wave object is created. Each sample value is then iteratively sent through the network with the value at the output node being read back into a new sine wave object. Since the values for a normal sine wave range from -1 to 1, each sample is linearly interpolated to a value between 0 and 1 prior to begin sent into the neural net. Once the value at the output node is read it is interpolated back to the appropriate sine value. This process is allowed to happen over a series of generations in which the network topology and link weights are changed via genome mutation and reproduction.

Additionally, the accumulation done at a node for incoming links was expressed as a product instead of as a summation:

$$y = f(\prod_{i=1}^{n} w_i \cdot x_i) \tag{2}$$

In this case, the action of the network is similar to the convolution operation in the frequency domain. Convolution is a mainstay of signal processing as it is the process by which nearly all modulation and filtering is done. Node values were initialized to one instead of zero to ensure the values were not decimated.

The process of fitness assignment is determined by the user. The user selects the sounds that they like, and these will be passed on to the next generation. A sound file is either accepted or rejected, no gradation of ratings is allowed in the current system. The number of inputs is held constant. If, for instance, five choices are presented to the user, and two are accepted, then each accepted choice is incrementally added to the next generation until the population has reached five again. Hence, we would have three clones of the first accepted genome and two clones of the second accepted genome in the new population for this example. Mutation later on ensures that the population is does not consist of only clones.

3. Experiments

We performed several experiments to test the potential of our system. As a baseline test, we solved the XOR problem. We used a population size of 100 for all XOR tests. Dynamic compatibility thresholding was used with a target species size of 5. We tried two different settings for the mutation probabilities. The first series of tests used a probability of .06 for adding a new link, .03 for adding a new node, and .91 for updating a weight. We then tested the system using roughly equal probabilities: .33, .33, and .34 for adding a new link, adding a new node, and updating a weight, respectively. Sample outputs are shown in Figure 2.

For the audio tests, we used the interface we developed for our system shown in Figure 3. The user is presented with output Carrier and Modulator waves and can load their own wave file. The user can then rate each audio track and run evolution with these choices.

As a test of the network's ability to perform simple FM, an "Identity" genome was created. This genome consisted of two input nodes and one output node. Each input was connected directly to the output node. The weights on each link were set to one, and the output node's activation function was set to the identity function, $f(x) = x$. Figure 4 shows the two input signals and the output of the network.

Our next test used a network with the same configuration but with a change in the activation function on the output node. For this test, a binary sigmoid function was used as the activation function on the output node. Figure 5 shows the output of the network.

Next, we experimented with adding input nodes. For this test, a network was configured with five input nodes and one output node. The network was run for one generation. Figure 6 shows the five input signals and the output signal.

Once we established that the network was correctly performing FM, mutations were turned back on. Mutations were set to the following: New link probability was set to .03; New node probability was set to .06; and Weight update probability was set to .91. The initial network was given

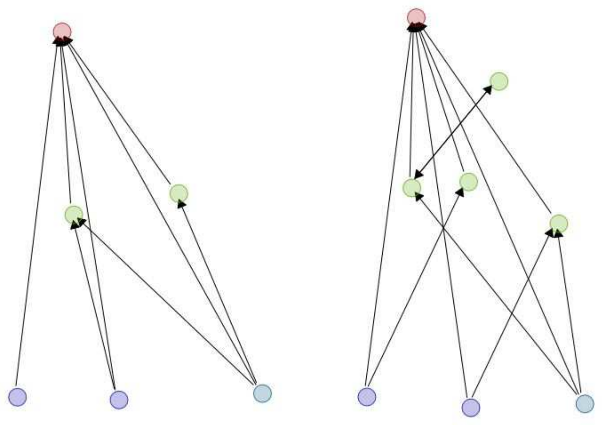


Figure 2: Two solutions found by our system that solve XOR. The genome on the left was created with high weight mutation rate. The genome on the right has nearly equal probabilities for link, node, and weight mutations. Input, bias, hidden, and output nodes are, blue, blue-gray, green, and red, respectively.

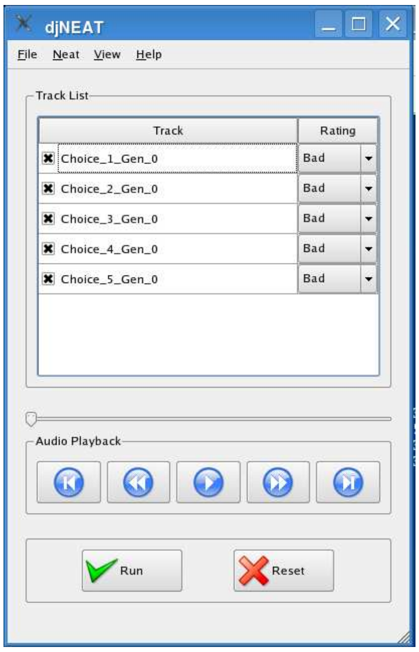
three inputs and was allowed to evolve over 5 generations. Figure 7 shows the resulting output.

3.1. Results and Discussion

Our results are presented in this section. Our system was able to successfully solve XOR. However, the results for our main task, audio creation, were mixed.

Figure 2 shows our XOR results. Champion genomes from two separate runs are shown. The genome on the left was the champion produced after three generations and was created using parameters that gave weight update mutations a very high probability. We see that a new hidden node was added in order to solve XOR. On the right, we see another genome that solved XOR. This genome was produced after eight generations. All mutations received the same probability for this run. We see that the this genome is far more complicated than the previous one. We think that the additional structural complexity allows the network to solve the XOR problem since the weights are not as flexible. Another thought is that the structure beyond the first hidden node is superfluous and that this genome was selected as soon as its weights were properly updated.

In terms of the audio tests, the network was able to successfully perform basic FM synthesis as shown in Figure 4. This is promising because it proves that the basic structure of a neural net is suitable for audio processing. Furthermore, we were able to get more “complex” sounds as we increased the number of input signals. Unfortunately, this has more to do with the basic FM process rather than any complexifiying that is done by the network. When we allowed the network to complexifiy over several generations, the re-



Track	Rating
Choice_1_Gen_0	Bad
Choice_2_Gen_0	Bad
Choice_3_Gen_0	Bad
Choice_4_Gen_0	Bad
Choice_5_Gen_0	Bad

Figure 3: System interface for audio tests.

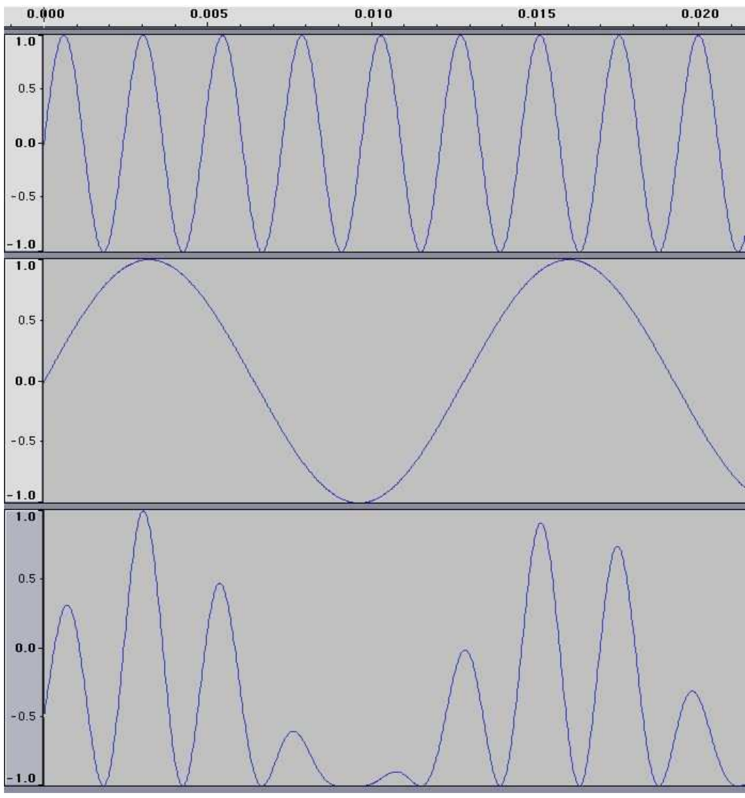


Figure 4: The output of the Identity network. The first two signals are the inputs to the network, and the bottom signal is the output.

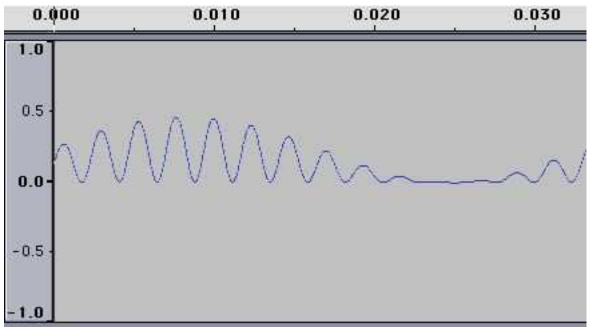


Figure 5: The output of network with the output node's activation function set to a binary sigmoid.

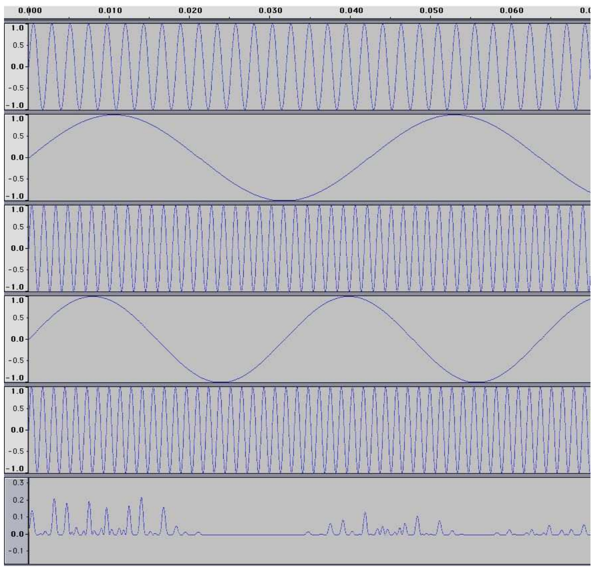


Figure 6: Mutiple input test. Five inputs and one output were used. The bottom signal is the network output.

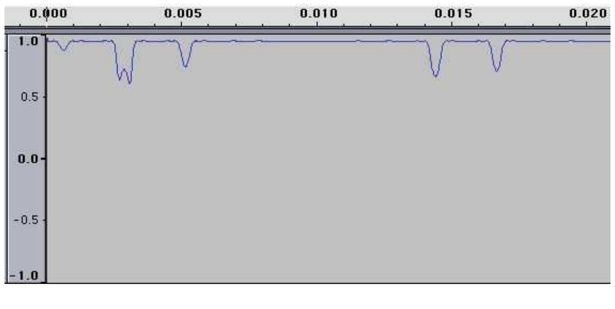


Figure 7: Output of a network with three inputs after five generations.

sulting signals tended to become noise. It appears that the complexification of the inputs signals at the "sample time" granularity is part of the problem. Since we are activating the network at the sample level, the signals tend to converge to some maximum point quickly. This can be seen in the output in Figure 7. As this graph shows the signal is essential pinned at 1.0, and this is only after five generations. There needs to be a mechanism for a more gradual change in the output signal.

Additionally, each signal is represented by a network in the total population. This is fine for the first one or two generations but after that the signals themselves become too "busy" for use in FM synthesis. Since the backbone of FM synthesis is a product function, the product of signals that are already busy will result in an even busier signal. Therefore, these signals tend to converge to noise very quickly. This implies that we need to have simple signals and, therefore, simple networks as part of our population at all times. This is somewhat contrary to the idea of complexification. Perhaps this could be solved by developing a measure for complexity and taking this into account as part of the fitness evaluation.

Furthermore, the standard activation function (i.e., sigmoid) did not enhance the signal; it simply acted to constrain the output of a node. The hope was that this function would add some level of non-linearity to the resulting output. This was not the case. It would seem that that the classical idea of a fitness function is somewhat unsuitable for processing audio in this manner. Better results might be obtained by the replacement of the fitness function with that of a standard audio filter, such as an ADSR filter.

The neuroevolution process could also be improved by an improved fitness evaluation procedure. First, rather than providing only two choices to the user for rating each sound file, many levels of ranking could be given. To provide further differentiated output choices, an automatic method of fitness evaluation could be employed as well. Such a system could apply evolution for some fixed number of generations before presenting choices to the user. Fitness might be hard to define in the case of automatic audio creation; however, this might be feasible for other tasks, such as noise removal. For instance, if we *a priori* know that an audio signal has been corrupted with Gaussian noise, then we may be able to retrieve the original signal back, assuming such a noise model. This would be comparable to function regression.

4. Summary and Future Work

We presented a system that dynamically changes sound using frequency modulation. The sound creation process was done using neuroevolution and was guided by the user. Complexification was further augmented by the use of FM synthesis in our audio processing. In such a process, a sine wave known as the Carrier signal has its frequency altered by a Modulator signal to produce a more complex sound. The structure of the neural network was changed in order to accommadate signal handling in the frequency domain through the use of products instead of summations inside the nodes.

This work raises many important questions and motivations for future work. First, the use of different synthesis techniques, such as Subtractive synthesis, could lead to interesting and richer results. As our work was entirely in the frequency domain, exploitation of time domain information could be useful as well. Sensor selection for a neural network is a delicate art, so it is possible that neural networks are better equipped to handle time domain values instead of frequencies. Finally, a more indirect approach could be adopted. Instead of attempting to directly process the input audio signals, a system could be designed to control the parameters used as part of a particular synthesis technique. We are currently investigating this parameter-based approach.

References

[1] J. Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21(7), 1973.

[2] Cook and Scavone. The synthesis toolkit (stk). In *Proceedings of the 1999 International Computer Music Conference*. IEEE, 1999.

[3] Edward A. Evenson. *The Telephone Patent Conspiracy of 1876: The Elisha Gray-Alexander Bell Controversy*. McFarland, 2000.

[4] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.