Scaffolding for Interactively Evolving Novel Drum Tracks for Existing Songs

In: *Proceedings of the Sixth European Workshop on Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMUSART 2008).* New York, NY: Springer, 2008
Winner of the Best Paper Award

Amy K. Hoover, Michael P. Rosario, and Kenneth O. Stanley

Evolutionary Complexity Research Group
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32836
`ahoover@cs.ucf.edu, michael.rosario@yahoo.com, kstanley@cs.ucf.edu`
http://eplex.cs.ucf.edu/neatdrummer

Abstract. A major challenge in computer-generated music is to produce music that sounds natural. This paper introduces NEAT Drummer, which takes steps toward natural creativity. NEAT Drummer evolves a kind of artificial neural network called a Compositional Pattern Producing Network (CPPN) with the NeuroEvolution of Augmenting Topologies (NEAT) method to produce drum patterns. An important motivation for this work is that instrument tracks can be generated as a *function* of other song parts, which, if written by humans, thereby provide a *scaffold* for the remaining auto-generated parts. Thus, NEAT Drummer is initialized with inputs from an existing MIDI song and through interactive evolution allows the user to evolve increasingly appealing rhythms *for that song.* This paper explains how NEAT Drummer processes MIDI inputs and outputs drum patterns. The net effect is that a compelling drum track can be automatically generated and evolved for any song.

Key words: compositional pattern producing networks, CPPNs, computer-generated music, interactive evolutionary computation, IEC, NeuroEvolution of Augmenting Topologies, NEAT

1 Introduction

Because music is at heart patterns and motifs that vary and repeat over time, it should be possible to generate such patterns automatically. Yet in practice, the most significant problem with computer-generated music is that it sounds *artificial* [1]. Subjectively, this problem is plain to the human ear. Yet its cause is subtle and difficult to establish objectively, making the solution elusive.

This paper takes a first step toward confronting the problem by showing that natural-sounding musical parts *can* be generated when they are constrained by a *scaffold*, i.e. an existing support structure, that is itself produced by competent humans. In particular, in experiments in this paper, the instrumental parts of an existing song are supplied as the scaffold upon which drum tracks are automatically generated, that is, the drum tracks are a *function* of the instrumental parts. The result is original drum sequences for existing songs that are often indistinguishable from human art.

All music has rhythm [2] and because rhythm is simpler than melody or harmony, rhythm generation is an appropriate place to begin to investigate the problem of producing natural patterns. The novel method for generating rhythms in this paper is implemented in a program called NEAT Drummer. It accepts existing human compositions as input to a type of artificial neural network (ANN) called a Compositional Pattern Producing Network (CPPN) 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, which can evolve increasingly complex CPPN-encoded patterns.

The results are drum patterns that sound like genuine human compositions when played with their associated songs. The ability to automatically generate musical parts that sound natural demonstrates an important lesson: As long as *part* of a song sounds natural and other instrument tracks are represented as functions of that part, they can *inherit* the plausibility inherent in the original design, suggesting a new path for computer-generated music.

2 Background

This section explains how interactive evolutionary computation (IEC) relates to computer generated music and introduces 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 [3]. Richard Dawkins first popularized IEC with Biomorphs, a visual representation of designs based on an L-System encoding [4, 5]. Like most IEC programs, to start a Biomorph run, a randomized population of biomorphs are generated. The user then selects which individual should be allowed to reproduce to evolve increasingly complex creatures. This idea naturally led to other visual IEC like the L-system-encoded Mutator [6], Karl Sims' genetic art [7], and eventually the first musical IEC application, Sonomorphs [8].

2.2 Computer Generated Music

Although music generated by computers can be interesting, typically it also sounds artificial, often because it lacks a global structure that progresses from the beginning to the end of the song [1, 9].

It is common for music generators, such as Sonomorphs, the first Biomorph-inspired music evolver, to focus on short phrases rather than on the entire song [8, 10]. These short phrases, which are selected and evolved by the user, may be extended through looping or manual juxtaposition.

Early connectionist approaches also evolved short phrases and represent change over time through recurrence [11]. Todd and Loy [11] first applied recurrent ANNs to music generation by training them to reproduce patterns with Real Time Recurrent Learning (RTRL). Chen and Miikkulainen [12] 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 avoids this problem by generating its rhythms from already-existing instrumental parts that span entire songs, thereby precluding 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)

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 [13, 14], or arbitrary random-topology networks [15], NEAT is the first to begin evolution with a population of small, simple networks and complexify the network topology over generations into diverse species, leading to increasingly sophisticated behavior. This section briefly reviews the NEAT method; Stanley and Miikkulainen [16, 17] 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.

Second, NEAT traditionally speciates the population so that individuals compete primarily within their own niches instead of with the population at large, which protects topological innovations. 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.

Third, unlike other systems that evolve network topologies and weights [15], 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.

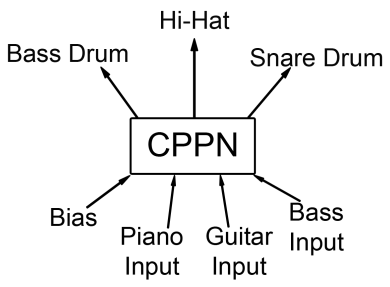
Finally, in NEAT Drummer, NEAT evolves a kind of ANN called a Compositional Pattern Producing Network (CPPN), which is designed to compactly represent patterns with regularities, such as pictures and songs [18]. What distinguishes CPPNs from ANNs is that in addition to traditional sigmoid functions, 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. In this way, drum tracks with regular patterns are easily discovered quickly.

3 Approach

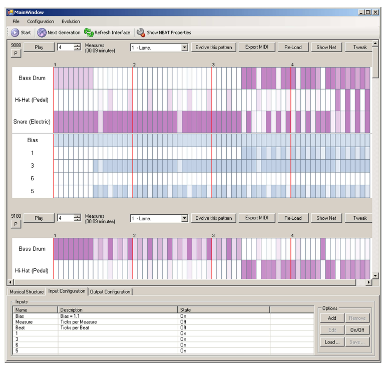
NEAT Drummer produces drum tracks for songs through two primary functions. First, it generates an initial set of candidate songs from a set of user-defined inputs. Second, it allows the user to further evolve the initial candidates.

3.1 CPPN Rhythm Generation

To generate the initial drum tracks, the user first specifies the inputs and outputs of the CPPN (figure 1a) through a GUI provided by the program (figure 1b).



(a) Inputs and Outputs



(b) NEAT Drummer Screenshot

Fig. 1. Using NEAT Drummer. (a) 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. (b) NEAT drummer presents an interactive evolution interface where visual representations of drum patterns help the user to decide whether to listen to each candidate and then select their favorites.

Because NEAT Drummer is based on the philosophy that a natural sound can be produced from a scaffold that is human-produced, the CPPN inputs must provide this scaffolding. From these inputs the CPPN derives its original patterns, which are therefore *functions* of the scaffolding.

For drum tracks in particular, the most natural scaffolding is the music itself (e.g. melody and harmony), from which the drum pattern can be derived. Thus, the user selects any combination of *channels* from a MIDI file to be input into the CPPN. These channels represent individual instrumental parts from the song. The main idea is that NEAT Drummer generates a rhythm that is a function of these channels, such that it is constrained by, though not identical to, their intrinsic patterns. 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 channel, e.g. bass and guitar, so that the rhythm is a function of both.

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. 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 channels are represented over time as *spikes* that begin high and decrease (i.e. *decay*) linearly (figure 2). 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.

The level of each CPPN output, on the other hand, 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.

3.2 Drum Pattern Interactive Evolution

As shown in figure 1b, NEAT Drummer displays the set of candidate patterns visually after they are generated. It is important to note that unlike in many

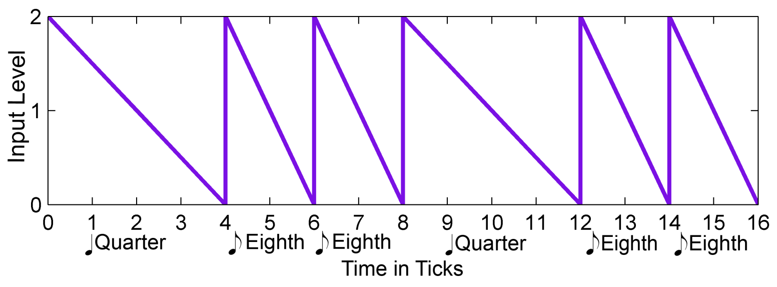


Fig. 2. Channel Input Encoding. Regardless of the instrument, each note in a sequence in any channel input to the CPPN is encoded as a spike that decays over the duration of note. The pattern depicted in this figure shows how eighth notes decay faster than quarter notes, thereby conveying timing information to the CPPN, which samples this pattern at discrete ticks.

evolutionary experiments, patterns in the initial generation *already* sound appropriate because they are functions of other parts of the song. This initial high quality underscores the contribution of the existing tracks to the structure of the generated 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 these high probabilities would be deleterious in typical NEAT experiments [16], because drum tracks are constrained to follow the music no matter what, 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 an individual connection was 90%.

4 Experimental Results

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 both 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.

To appreciate the results in this section it is important to judge the subjective natural quality of the generated rhythms. Thus MIDI files for every experiment in this section are available at `http://eplex.cs.ucf.edu/neatdrummer/`.

Figure 3 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 strong subjective sense of appropriate stylistic sophistication and creativity.

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 1, 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 4. NEAT Drummer produces similarly natural and style-appropriate rhythms for this song as well, suggesting its generality. Because style is inherent in the scaffold, 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.

Interestingly, *listening* to the songs with their generated drum tracks produces a surprisingly natural feel, lacking the usual "mechanical" quality of

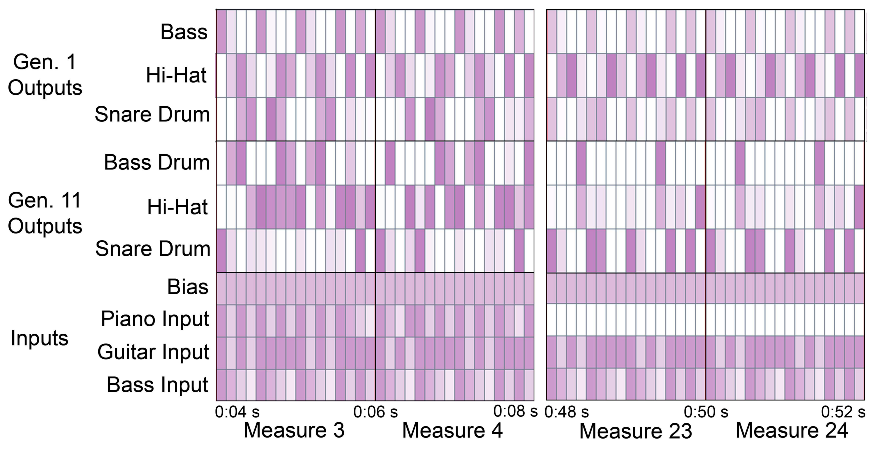


Fig. 3. Johnny Cope results. Results are depicted from two different generations at two different parts of the song. The inputs from the original song are shown at bottom. The motif in measures three and four is typical of the first part of the song until measure 23, when it switches to a different motif in both generations. 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/neatdrummer/`).

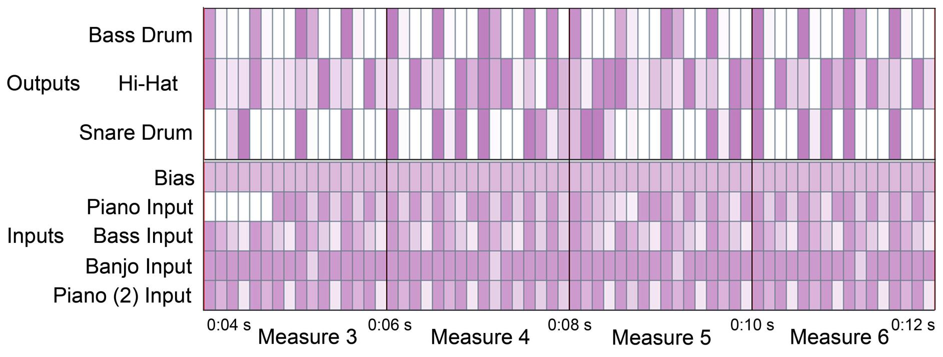


Fig. 4. Oh! Susanna outputs. This pattern from measures three through six of *Oh! Susanna* is from generation 25 of evolution. The network evolved 15 hidden nodes and 41 connections. Near the end of measure four is a particularly improvisational riff in the snare that transitions to measure five. This 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/neatdrummer/`).

computer-generated music. The rhythms appear to exhibit creative flourishes and interjections that suggest personality. Yet these are byproducts 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.

5 Discussion and Future Work

NEAT Drummer exploits the fact that the different parts of a musical piece are functionally related, which is what allows the output to sound natural and song-appropriate. The implications of this functional relationship are significant because it means that many parts of a song are less complex than they seem to be when put in the context of the rest of the instruments.

Formally, any pattern over time can be described as a function of time, $f(t)$. However, a good pattern for a particular drum alone may be highly complex with respect to t, making its discovery prohibitive. Yet given another part $p(t)$ that varies over time, because the parts of a song are related to each other, it may be easier to discover $g(p(t))$ than $f(t)$ even though they produce the same output. In effect, p provides a *scaffold* upon which other parts can be attached.

In this way, the drums are a function of the melody and the harmony, and, interestingly, the melody and harmony are also an inverse function of the drums. The important implication is that as long as *something* is human-generated, the rest will naturally inherit its intangible natural quality on their own. Furthermore, they will automatically inherit the *style* intrinsic in the scaffold, removing the need for style-specific considerations. Finally, interactive evolution allows the user to refine and elaborate the initial track in an unlimited variety of ways.

The next step is to move from generating percussion to generating bass, harmony, and eventually melody from each other. In the longer term, we aim to explore the minimum amount of prior information that is necessary to form a scaffold upon which other parts are generated. If this basis can be reduced to a very simple core encoding that even laymen can specify with ease, then perhaps it will become possible for almost anyone to generate complete songs with little human effort or expertise.

6 Conclusion

This paper introduced NEAT Drummer, a program that can generate novel drum tracks for songs sequenced in MIDI. The main idea is to input instrumental tracks directly into a kind of neural network called a Compositional Pattern Producing Network (CPPN), which produces a pattern that is a function of its input. In this way, the drum sequences output by the CPPN are related to the song inputs, which thereby supply a scaffold for the drums. Human users can then evolve and thereby elaborate the track to satisfy their specific tastes. Drum tracks generated for two popular folk songs exhibit a natural style appropriate to each song that lacks the familiar computerized feel of computer-generated music. In the future, the scaffolding approach may apply to generating instrumental tracks as well, and eventually may enable nearly-autonomous music generation.

Acknowledgments

Special thanks to Barry Taylor for granting special permission to utilize his own MIDI productions of folk music in this work. Barry Taylor originally sequenced both Johnny Cope and Oh! Susanna (both without percussion), which are the two songs for which drum tracks were generated in Section 4. This research was supported in part by NSF grants IIS-REU: 0647120 and IIS-REU 0647018.

References

1. McCormack, J.: Open problems in evolutionary music and art. In: Proc. of Applications of Evolutionary Comp., (EvoMUSART 2005). Volume 3449 of Lecture Notes in Computer Science., Berlin, Germany, Springer Verlag (2005) 428–436
2. London, J.: Rhythm. http://www.grovemusic.com (2007)
3. Takagi, H.: Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. Proc. of the IEEE **89**(9) (2001) 1275–1296
4. Dawkins, R.: The Blind Watchmaker. Longman, Essex, U.K. (1986)
5. Lindenmayer, A.: Mathematical models for cellular interaction in development parts I and II. Journal of Theoretical Biology **18** (1968) 280–299 and 300–315
6. Todd, S., Latham, W.: Evolutionary Art and Computers. Academic Press, London (1992)
7. Sims, K.: Artificial evolution for computer graphics. In: Proc. of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91), New York, NY, ACM Press (1991) 319–328
8. Nelson, G.L.: Sonomorphs: An application of genetic algorithms to growth and development of musical organisms. In: 4th Biennial Art and Technology Symp. (1993) 155–169
9. Husbands, P., Copley, P., Eldridge, A., Mandelis, J.: 1. In: Evolutionary Computer Music. Springer London (2007)
10. Biles, J.A.: 2. In: Evolutionary Computer Music. Springer London (2007)
11. Todd, P.M., Loy, D.G.: Music and Connectionism. MIT Press, Cambridge, MA (1991)
12. Chen, C.C.J., Miikkulainen, R.: Creating melodies with evolving recurrent neural networks. In: Proc. of the 2001 Int. Joint Conf. on Neural Networks, Washington, D.C., IEEE Press (2001) 2241–2246
13. Gomez, F., Miikkulainen, R.: Solving non-Markovian control tasks with neuroevolution. (1999) 1356–1361
14. Saravanan, N., Fogel, D.B.: Evolving neural control systems. IEEE Expert (1995) 23–27
15. Yao, X.: Evolving artificial neural networks. Proc. of the IEEE **87**(9) (1999) 1423–1447
16. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10** (2002) 99–127
17. Stanley, K.O., Miikkulainen, R.: Competitive coevolution through evolutionary complexification. **21** (2004) 63–100
18. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines Special Issue on Developmental Systems **8**(2) (2007) 131–162