# Learning to Dance through Interactive Evolution

In: *Proceedings of the 8th European event on Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMUSART 2010). New York, NY: Springer, 2010*

Greg A. Dubbin and Kenneth O. Stanley

School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816, USA

**Abstract.** A relatively rare application of artificial intelligence at the nexus of art and music is dance. The impulse shared by all humans to express ourselves through dance represents a unique opportunity to artificially capture human creative expression. In particular, the spontaneity and relative ease of *moving to the music* without any overall plan suggests a natural connection between temporal patterns and motor control. To explore this potential, this paper presents a model called *Dance Evolution*, which allows the user to train virtual humans to dance to MIDI songs *or* raw audio, that is, the dancers can dance to any song heard on the radio, including the latest pop music. The dancers are controlled by artificial neural networks (ANNs) that "hear" MIDI sequences or raw audio processed through a discrete Fourier transform-based technique. ANNs learn to dance in new ways through an interactive evolutionary process driven by the user. The main result is that when motion is expressed as a function of sound the effect is a plausible approximation of the natural human tendency to move to music.

## 1 Introduction

The ubiquity of dance throughout the cultures of the world [1] hints at its deep connection to human creativity and self-expression. The power of music and dance as a tool for self expression is further demonstrated by the popularity of such music and rhythm-oriented games as *Guitar Hero*[1], *Rock Band*[2], and *Dance Dance Revolution*[3]. Yet although in recent years researchers in artificial intelligence (AI) have begun to focus on creativity in music and art [2–5], with few exceptions [6], dance is less explored. Nevertheless, dance can potentially provide insight into how the auditory and motor modalities are connected in creative self-expression. Thus its study is relevant to the enterprise of AI.

Unlike Yu [6], who focused on choreographed dance sequences, the model in this paper investigates the more spontaneous self-expression that results from

[1] Guitar Hero (R) is a trademark of Activision Publishing, Inc.
[2] Rock Band, Rock Band 2 and all related titles are trademarks of Harmonix Music Systems, Inc., an MTV Networks company.
[3] (C) 2008 Konami Digital Entertainment, Inc. "Dance Dance Revolution" is a registered trademark of Konami Digital Entertainment Co., Ltd. KONAMI is a registered trademark of KONAMI CORPORATION.

simply listening to entertaining music, such as in a club setting. In a step toward generating spontaneous dance to arbitrary music, this paper presents a model called *Dance Evolution* in which virtual dancers learn to dance to music encoded as either MIDI or raw audio. In effect, dancers can learn to dance to any song that you might hear in a dance club or on the radio. The model in Dance Evolution can take MIDI sequence data or process raw audio through discrete Fourier transforms to extract an approximation of such data. The resulting temporal progression is input into an artificial neural network (ANN), which outputs a sequence of motor commands that control the body of the virtual dancer. Thus the motion of the dancer becomes a *function* of the beats of the song.

Of course, an important question is how the ANN can learn to make the right moves. However, it turns out that it is possible to quickly discover mappings between audio and motor output that produce movements that appear natural, suggesting that one reason dance is so appealing is that its search space is forgiving. Thus the aim in Dance Evolution is not to learn an *optimal* dance but rather to enable the user to effectively explore the space of possible dances. For this purpose, the user drives an interactive evolutionary algorithm built on the NeuroEvolution of Augmenting Topologies (NEAT) approach to evolving ANNs. In effect, the user *breeds* new dancers from ones that were appealing in the past. In fact, because each evolved ANN embodies the *personality* of a dancer, the same dancer can be transferred from one song to another.

The main insight, that dance can be considerd a function of changing audio over time, suggests a direct coupling between motor control and audio processing. By implementing a model based on this principle, the practical result is an interactive application in which an unbounded space of dance behaviors can be explored and assigned to any song.

## 2 Background

This section reviews foundational technologies to the Dance Evolution approach.

**Interactive Evolutionary Computation** IEC is a growing field within machine learning that takes advantage of humans' ability to make sophisticated subjective judgments [7]. Early IEC implementations include Richard Dawkins's [8] Biomorphs, which evolved visual patterns, and the pioneering work of Sims [5, 9], who evolved both art and virtual creatures. In IEC the user is presented with a set of (usually visual) alternatives and evaluates their fitness. The evolutionary algorithm generates a new generation of candidates based on this feedback. The process repeats until the user is satisfied. While the risk is that the user may become fatigued before finding a satisfactory candidate [7], the hope in Dance Evolution is that watching dancers is sufficiently motivating in part to mitigate the effect of fatigue.

**NeuroEvolution of Augmenting Topologies** Dance Evolution encodes dance policies as ANNs that "hear" input from a music file and output requests for limb movement. The ANNs in Dance Evolution are trained through the NeuroEvolution of Augmenting Topologies (NEAT) method, which has proven successful in a variety of control and decision making tasks [10–12]. Thus NEAT makes a good platform for evolving controllers for dancers.

NEAT begins evolution with a population of small, simple networks and grows the network topology over generations, leading to increasingly sophisticated behavior. For evolving dance, this process means that dances can become more elaborate and intricate over generations. Stanley and Miikkulainen [12] provide complete introductions to NEAT.

**Music Processing** If dancers can only dance to processed MIDI files, the user can only enjoy a limited library of songs. To broaden the possibilities, this paper takes the significant step of extending the functionality to the popular MPEG-1 Audio Layer 3 (MP3) music format. This capability allows the user to enjoy a normal library of songs. Thus the problem is to parse raw sound in such a way that the ANN can interpret it. The approach in this paper is inspired by an algorithm described by Scheirer [13] that can determine the beat from raw musical input.

## 3 Approach

Dance Evolution was implemented in the Panda3D[4] simulator, which was chosen for its rapid prototyping capabilities and ability to simulate articulated bodies. This section details the main approach, starting with inputs.

### 3.1 ANN Inputs

To allow the ANN to "hear" the music, data from the song is input over time as a vector representing the current pulses as the song unfolds. For MIDI (i.e. *musical instrument digital interface*) files, this vector contains four parameters. The first three are periodic functions of the beat, measure, and length of the song. The last signal in the vector represents the volume of the *low drum* track, which adds song-specific variation. In this way, the ANN is aware of the major temporal building blocks of music and generates dance as a function of that information. Because MIDI files represent this data explicitly, it is easily extracted and input into the ANN. However, the MIDI format only allows a proof of concept of the technique because most popular songs are available only in raw audio. Thus a key focus of this research is to extend the capability to dancing to raw audio.

### 3.2 Audio Processing

As Scheirer [13] explains, a beat is a repeating pulse with a regular period throughout a piece of music. To discover the beat, these pulses must be identified;

[4] Panda 3D Software Copyright (c) 2000-2005, Disney Enterprises, Inc. All rights reserved

however they can appear distinctly within different *subbands* of the song. Therefore, to recognize pulses, the song must first be decomposed into its subbands. This task is accomplished efficiently with a Fast Fourier Transform (FFT).

The FFT in Dance Evolution decomposes the amplitude vector formed from 1,024 samples of the song, which combined represent approximately $\frac{1}{40}$ of a second, into multiple subbands that represent the average energy of the song in several frequency-ranges. In particular, Dance Evolution decomposes the song into 15 subbands whose ranges are $\left[\frac{(\frac{N}{15}*i+4)*f_e}{N}, \frac{(\frac{N}{15}*(i+1)+4)*f_e}{N}\right]$ Hz with $i \in [0, 14], N = 1,024$, and $f_e$ set to the sampling rate of the song, which is usually about $44,100$ Hz (the constant 4 is the remainder of dividing 1,024 by 15). As noted by Scheirer [13], differing range divisions do not have a significant effect on results. The FFT also calculates the direct current (DC) term, i.e. the non-periodic component of the vector that represents the average energy of the song.

A beat occurs when the energy in a subband increases dramatically with respect to the average energy of the subband. The FFT of the 1,024 samples represents the near-instantaneous energy of each channel. To search for a beat, this energy is averaged with the previous 43 samples, which represents about one second of time for the common sampling rate of 44,100 Hz. This interval was selected to allow real time processing while maintaining sufficient fidelity.

The ratio of the instantaneous energy to the average energy amplifies differences between them such that a high ratio means that the pulse may be a beat. This pulse information can be input directly into an ANN, causing a spike in the input to correspond to a pulse in the music. This procedure both reduces time spent processing the music and preserves non-beat pulses, which may still be important to the dance. The non-beat pulses have a similar affect to the drum track information from the MIDI files. 16 values representing the pulses from each of the 15 subbands as well as the DC component are input into the ANN. The input vector is calculated and fed into the ANN in real time in each frame.[5]

To make pulse data suitable for ANN input, it must be (1) normalized and (2) lengthened. Because the magnitude of a raw audio pulse is unbounded, the sigmoid function $f(x) = \frac{1}{1+e^{-\alpha(x-\beta)}}$ is applied to each input to both limit the pulse values to $[0, 1]$ and further separate the strong pulses from the weak ones, in effect filtering out the weak signals. Based on an analysis of several songs, $\alpha$ and $\beta$ are set to 1 and 6 respectively to keep the point of inflection above the bulk of the weaker pulses.

The second problem is that the pulses occur over short periods of time, which does not give the ANN enough time to properly react before the input pulse terminates. Dances produced with such short pulses thus appear jerky. To address this problem, the ANN is provided with a "memory" of the last few inputs by adding a fraction of the previous input into the current one such that $I_t = f(x) + \gamma \cdot I_{t-1}$. This final step provides the ANN with smooth decay curves following each pulse. The result is smooth, natural-appearing motion.

[5] The engine updates the visual display each such frame.

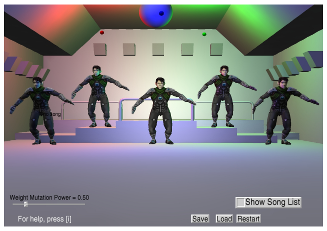


**Fig. 1. Dance Evolution User Interface.** The user can click on dancers to breed them and control the degree to which they change with the mutation slider in the lower-left.

### 3.3 ANN Outputs

The ANN controls three-dimensional models within an interactive environment. The output nodes of the ANN request the actuation of the joints of the model. The models' joints are chosen from the head, spine, arms, hips, and legs such that each joint has enough mobility to noticeably affect the dances. Each output affects the angle of an axis (heading, pitch, or roll) of one joint, for a total of 34 outputs. Every frame, the outputs of each ANN are queried and the angles of the models are updated accordingly. Activation travels through the the ANN at a rate of one link per frame. It is important to note that because ANNs evolved by NEAT can contain recurrent connections, in principle it is possible to react to more than just the current beat.

To display the dance, the outputs of the ANN must be converted into actuation commands to move the models. Each frame, the next input vector is calculated from the most recently heard music and loaded into the ANN. Each output of the ANN is in the range $[0, 1]$ and scaled linearly into $[-.2, .2]$. This activation is scaled according to the frame rate so that dancers are consistant regardless of frame rate. Finally, the joint is moved along a sine curve between the physiological limits at a rate determined by the activation, creating a rhythmic motion that avoids being stuck at the limits. That is, the limbs continually oscillate between their limits with a frequency determined by the activation, such that higher activation causes faster movement.

In this way, because each input represents the most recent pulses in the song as heard by the user, the ANN is able to react in real time to the music.

### 3.4 ANN Training

ANNs in Dance Evolution are trained through IEC, which allows the user to direct their evolution intuitively. Left-clicking on one of the five model dancers produces a new generation of mutants of that model's ANN. The model that is clicked keeps its ANN, which ensures the favorite dancer is not lost. Right-clicking creates a set of hybrids by crossing over the clicked model's ANN with each of the other models' ANNs following the NEAT method, and replacing the old ANNs of each unclicked model with the new offspring. The user can control the mutation power through a slider provided at the bottom of the screen, which gives the user significant control over evolution. The interface is shown in figure 1. The population can be initialized randomly or from ANNs saved from previous sessions. In addition to saving trained dancer ANNs, the user can load any previously saved ANN.

Users can choose to play any song in their library. Interestingly, ANNs can react to any such song, including those for which they were not trained. In this way, the same dance *personality*, which is embodied in an ANN, can be transferred from one song to another.

## 4 Experiments and Results

This section is divided into two parts: a brief analysis of learning to dance to MIDI and the more ambitious goal of dancing to raw audio. Please note that the text in this section is accompanied with video demonstrations at:
`http://eplex.cs.ucf.edu/dance-evolution-videos.html`.

### 4.1 Dancing to MIDI

Learning to dance to MIDI is significantly easier than learning with raw audio because the beat and measure are provided explicitly by the MIDI format. Showing what is possible under such conditions provides a context for the results of the greater challenge of learning to dance to raw audio.

MIDI dancers were evolved to MIDI dance songs composed by Bjorn Lynne (Shockwave-Sound.com). The main result is that a wide variety of dances evolved that tightly follow the rhythm of the song, demonstrating that the idea that dance can be generated as a function of temporal information in music can work in principle (see "MIDI Sequence" video). While some of the specific bodily motions sometimes appear unrealistic, it is possible to evolve toward more realistic motions. The quality of results with MIDI was sufficient to win the Best Student Video Award at the AAAI video competition [14]. The dancers perform the most dramatic movements and changes to the beat of the song. The next section investigates the performance of the approach when confronted with the greater challenge of raw audio.

### 4.2 Dancing to Raw Audio

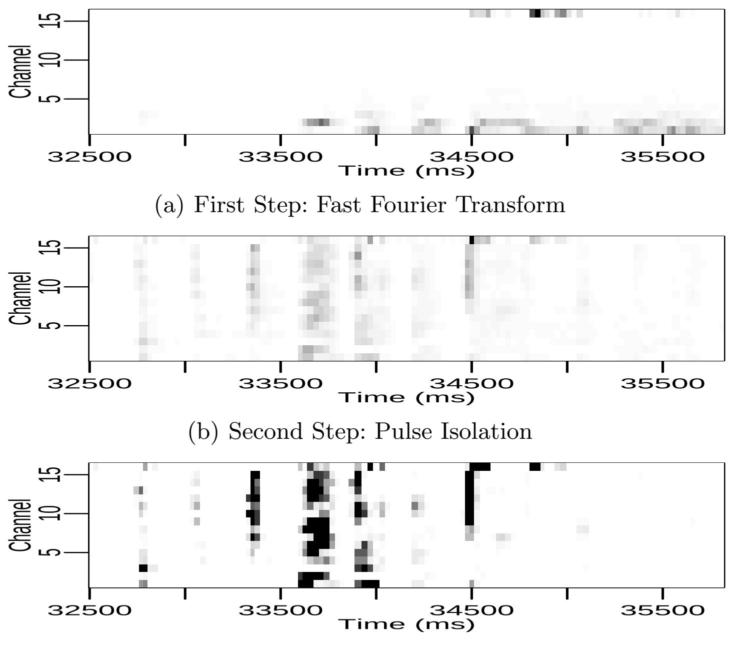
This section analyzes raw audio performance in detail through a variety of specific experiments. The primary aim is to provide significant insight into *how* the results follow from the raw audio processing procedure, from which it will be possible to expand and improve further in the future.

A brief sequence from the song *Tubthumping*[6] by Chumbawamba is chosen to illustrate in detail the responsiveness of the algorithm to pertinent components of raw music. Specifically, the inputs resulting from processing and the

[6] *Tubthumping* is from the album "Tubthumper," released by EMI Group Ltd.

corresponding ANN outputs are graphically analyzed to test whether processing captures important signals from the music and whether the ANN reacts accordingly. *Tubthumping* has a very distinct chorus in which the lyrics state "I get knocked down, but I get up again." These lyrics provide a reference point in the analysis of the network outputs.

Figure 2 depicts the initial three steps of the raw audio processing of the segment with the lyrics, "I get knocked down, but I get up again," which is completed before inputting data into the ANNs (see "Tubthumper Clip" video for actual footage). Figure 3 demonstrates the movement during this segment as well as the inputs and outputs of the ANN that generated this dance.



(a) First Step: Fast Fourier Transform

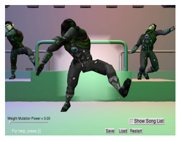
(b) Second Step: Pulse Isolation

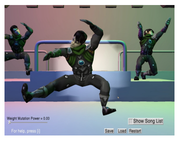
(c) Third Step: Sigmoid Transformation

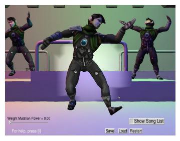
**Fig. 2. Processing Raw Audio.** The first three steps of processing the song *Tubthumping* are shown. The FFT results are shown for each channel (a). The ratio of the instantaneous to the average FFT isolates pulses in the music (b). The sigmoid function then filters out weak pulses and normalizes the input (c).

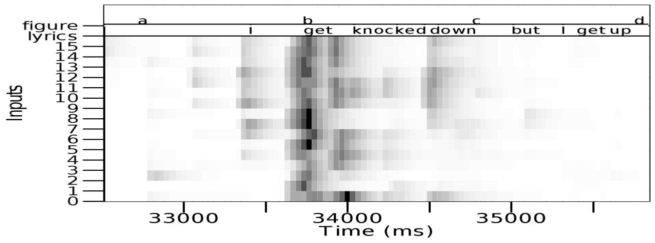
The ANN that controls the dancer in figure 3 evolved the humorous behavior of bobbing down when the chorus sings "I get knocked down." Spikes in the processed input can be observed at each of these lyrics. Note that these spikes represent instrumental beats in various frequencies that happen to correspond to the timing of the lyrics, which is how Dance Evolution learns to respond. Other spikes representing a variety of instrumental sounds are also apparent, indicating

(a) Time = 32727 ms

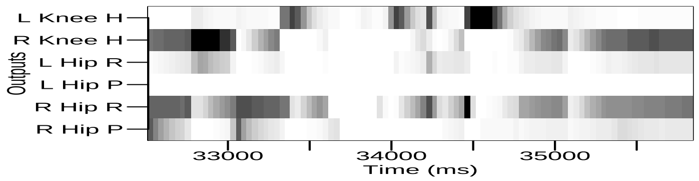

(b) Time = 33726 ms


(c) Time = 34764 ms


(d) Time = 35726 ms



(e) Input Strength for each Channel Over Time



(f) Upper and Lower Leg Outputs Over Time

**Fig. 3. Dancing to Music.** An ANN trained with Chumbawamba's *Tubthumping* dances to the chorus of "I get knocked down, but I get up again". The sequence (a–d) are screen shots taken at regular intervals during the dance. Over this time, the ANN sees the inputs processed from this part of the song (e). The outputs for the left and right legs (f) cause the dancer's legs to change direction quickly at the words "I" and "down," corresponding to pulses in the input (e).

that the inputs of the ANN indeed receive spikes in the music to which a human would react. Correlated with this input, the output in figure 3f shows that the ANN requests the right and left knees to change direction (i.e. the sign switches) when the lyrics, "I get knocked down" are sung (approximately 33,300 ms into the song). The act of both legs changing direction simultaneously causes the dancer to dip. Thus figure 3 confirms that, with user input, NEAT could find a policy to express the desired behavior as a function of change in subbands.

A single ANN can transfer some of its characteristic behavior between songs, but also exhibits new behavior resulting from different active frequency subbands in different songs. Demonstrating this capability to transfer, a single ANN was trained on George Clinton and the Parliament's *Give Up the Funk (Tear the Roof off the Sucker)*[7] and transferred to to Billy Joel's *Piano Man*[8]. Characteristic

[7] *Give Up the Funk (Tear the Roof off the Sucker)* is from the album "Mothership," released by Casablanca Records.

[8] *Piano Man* is from the album "Piano Man," released by Columbia Records.

behaviors such as spreading the arms remain throughout, although the dance slows to match the tempo of *Piano Man* (see "transition clip").

There are generally several interesting dances each generation; therefore, further evolution is most naturally directed toward curiosity-driven exploration rather than a defined goal. Thus an entertaining strategy for Dance Evolution is undirected search, i.e. selecting the most interesting or fun behavior.

Accordingly, five runs, evolved over two songs each, were executed with such a strategy. Every initial ANN, with no prior training, contains 51 nodes and 578 connections. The chance of adding a node is 15% and the chance of adding a connection is 25%. After 10, 20, 30, and 40 generations, the average number of nodes and connections was 54.48 nodes and 579.76 connections, 54.68 nodes and 583.56 connections, 55.28 nodes and 585.88 connections, and 56.08 nodes and 588.36 connections, respectively. Because the songs were on average 4.1 minutes, the average number of generations per minute was 6.1, which means approximately one generation every 9.8 seconds. This short duration suggests that subjective impressions of dance can form quickly. The dances generated (see "variety clip" featuring *Walk Like an Egyptian*[9]) included splits, intricate arm and leg movements, head bobbing, toe tapping, and more, usually in concert.

In general, as evolution progresses, behaviors appear more correlated and smooth; which is both a function of selection and the added network structure.

## 5 Discussion

Music is composed of a series of periodic spikes in sound energy of different frequencies. Humans dance by reacting to these events by creatively mapping them to motor outputs. Similarly, Dance Evolution evolves ANNs that react to audible spikes of different frequencies through motor outputs.

Dance Evolution applies AI to a unique domain. Both the domain and the interactive interface minimize the risk of user fatigue, allowing Dance Evolution to exploit human intuition to solve an otherwise poorly defined problem.

A promising future extension to Dance Evolution is to embed knowledge of the inherent symmetry of the body into the genetic encoding. Such *indirect encoding* [15, 16] would bias the networks toward producing dance sequences that are partially symmetric, which may increase their natural appeal.

One significant practical application of this achievement is within the video game industry. Games based on music such as *Guitar Hero* and *Rock Band* are becoming increasingly popular. For game designers, the capability to produce a large variety of dancers to raw audio recordings can potentially enhance such games significantly and speed up their development.

Perhaps most interesting is that the functional perspective has also been shown plausible in music (as a function of time [2]), and art (as a function of space [4]). By demonstrating that this model can work, Dance Evolution suggests that human creative expression is indeed possible to model in AI.

[9] *Walk Like and Egyptian* is from the album "Different Light," released by Columbia Records.

## 6 Conclusion

This paper presented Dance Evolution, a program that takes advantage of IEC and ANNs to approach the subjective problem of learning to dance. Additionally, the paper described the algorithm implemented in Dance Evolution to process raw audio files into a form suitable for ANN controllers. Dance Evolution proved capable of training both specific and varied dances on a range of songs. Because these ANNs simply express functions of the music, the dances that they produce are naturally coupled to the music.

## References

1. Kurath, G.P.: Panorama of dance ethnology. Current Anthropology **1**(3) (1960) 233–254
2. Hoover, A.K., Rosario, M.P., Stanley, K.O.: Scaffolding for interactively evolving novel drum tracks for existing songs. In et. al., M.G., ed.: Proc. of the Sixth EvoMUSART 2008, Springer (March 2008) 412–422
3. Romero, J., Machado, P., eds.: The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Natural Computing Series. Springer Berlin Heidelberg (November 2007)
4. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Stanley, K.O.: Picbreeder: Evolving pictures collaboratively online. In: CHI '08: Proc. of the 26th annual SIGCHI, New York, NY, USA, ACM (2008) 1759–1768
5. Sims, K.: Artificial evolution for computer graphics. In: Proc. of SIGGRAPH, ACM Press (July 1991) 319–328
6. Yu, T., Johnson, P.: Tour jeté, pirouette: Dance choreographing by computers. In: GECCO. (2003) 156–157
7. Takagi, H.: Interactive evolutionary computation: Fusion of the capacities of ec optimization and human evaluation. In: Proc. of the IEEE. (2001) 1275–1296
8. Dawkins, R.: The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design. W. W. Norton (September 1986)
9. Sims, K.: Evolving virtual creatures. In: SIGGRAPH '94: Proc. of the 21st annual conf. on Comp. graphics and interactive techniques, New York, NY, USA, ACM (1994) 15–22
10. Taylor, M.E., Whiteson, S., Stone, P.: Temporal difference and policy search methods for reinforcement learning: An empirical comparison. In: Proc. of the Twenty-Second Conference on AI. (July 2007) 1675–1678 Nectar Track.
11. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Evolving neural network agents in the nero video game. In: Proc. of the IEEE 2005 CIG05. (2005)
12. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10** (2002) 99–127
13. Scheirer, E.D.: Tempo and beat analysis of acoustic musical signals. Journal of Acoustical Society of America **103**(1) (January 1998) 588–601
14. Balogh, J., Dubbin, G., Do, M., Stanley, K.O.: Dance evolution. In: Proceedings of the Twenty-Second National Conference on Artificial Intelligence AI Video Competition, AAAI-Press (2007)
15. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based indirect encoding for evolving large-scale neural networks. Artificial Life **15** (2009)
16. Stanley, K.O., Miikkulainen, R.: A taxonomy for artificial embryogeny. Artif. Life **9**(2) (2003) 93–130