PRIMAL-IMPROV: Towards Co-Evolutionary Musical Improvisation

Marco Scirea[1], Peter Eklund[1], Julian Togelius[2], and Sebastian Risi[1]
msci@itu.dk, petw@itu.dk, julian@togelius.com, sebr@itu.dk
[1]Center for Computer Games Research, Robotics, Evolution and Art Lab, IT University of Copenhagen, Denmark
[2]Department of Computer Science and Engineering, New York University, NY, USA

***Abstract*—This paper describes a work in progress on co-evolving Artificial Neural Networks (ANNs) for music improvisation. Using this neuro-evolutionary approach the ANNs adapt to the changes in the human player's music as input, while still maintaining some of the structure of the musical piece previously evolved. The system is called PRIMAL-IMPROV and evolves modules that are composed of two ANNs, one controlling pitch and one controlling rhythm. The results of a quantitative study show that, by only introducing simple rules as fitness functions, the system is able to generate more interesting arrangements than ANNs evolved without a specific objective. The emerging and interesting musical patterns that are produced by the evolved ANNs hint at the promising potential of the system.**

I. INTRODUCTION

Improvements in computer science, evolutionary computation, and music informatics have enabled many creative performance systems producing music. The goal of most such systems is to exhibit musicality, which is determined by a listening audience or a performing musician. Many diverse approaches have been applied to this domain due to the variety of methods in artificial intelligence (reinforcement learning, evolutionary algorithms, statistical modeling, etc), and to different interpretations and genres of music applied to a specific method [1]. Another goal of artistic expression is to create interesting structures that one cannot immediately imagine. In musical history we find plenty of rules and limitations (for example the 12-tone series), which musicians use to create interesting content. In a way, simple limitations help to support the creative process. Interactive evolution, where human aesthetic judgment is included in the fitness function, has often been used to guide evolutionary creative systems. Another approach is to base the fitness function on music theory to emulate the way a human would play/compose. PRIMAL-IMPROV, while still in its infancy, presents a different approach by introducing very simple rules in the fitness function, which do not necessarily need to rely on musical theory. The most interesting feature is how, even though the system generates music according to rules of our own construction, musical structures emerge that are not obvious to human creators. These structures arise from the freedom we give to the Artificial Neural Networks (ANNs) to musically express characteristics of their topology, thanks to the multitude of ways that the fitness functions can be satisfied. Moreover, by avoiding domain knowledge our system can create music that transcends usual harmony, which can lead to unexpected improvisations. Part of the architecture described in this paper is inspired by *MaestroGenesis* [2], a tool for computer-assisted composition that is based on interactive evolution. The main difference between PRIMAL-IMPROV and *MaestroGenesis* is that our system does not require human input, but instead co-evolves its multiple voices according to "primitive" fitness functions. The system presented in this paper is based on the idea of evolving improvisational modules, each capable of creating monophonic melodies. The system can be used to create accompaniments to predefined melodic phrases or, more interestingly, provide an adaptive improvisational companion to a human player. While there are other evolutionary real-time improvisers (GenJam [3], Bown [4]), the proposed system presents a novel modular architecture that allows for the creation of an arbitrary music "instrument". Finally, when considering the real-time application of PRIMAL-IMPROV, we can note how the feedback loop between the player and the system closes: as the musician plays, he/she finds out how the system reacts to the music played; at the same time the system adapts to the music the human is playing, this change will likewise influence the musician, and so on.

II. BACKGROUND

A. Music Generation

Music generation is a field that has received much attention in the last decade [5]. The approaches in the literature are diverse and range from creating simple sound effects, to avoiding repetition when playing human-authored music, to creating more complex harmonic and melodic structures [6], [7]. Wooller [8] divides approaches to procedural music generation into two categories, namely *transformational* and *generative* algorithms. The field is too large to survey in the limited space of a conference paper but we will give some relevant and representative examples. There are many examples of evolutionary algorithmic approaches to generating music, two notable examples are the methods to evolve piano pieces by Loughran *et al.* [9] and Dahlstedt [10], although many more can be found in the *Evolutionary Computer Music* book [11]. A relevant example of improvisation system is Patchet's Continuator [12], which is a Markov system that learns from the musician's input and produces accompaniment accordingly. The main difference between this system and our approach is that we believe PRIMAL-IMPROV gives more

freedom of expression to the modules, creating something that is less "dependent" on the human input.

B. Live algorithms

As discussed in the previous section, there are many applications of computers and algorithms to music that use the *computer-as-composer* paradigm. In this section we briefly discuss some of the other paradigms to better situate our approach. *Live coding* is a practice that has become more and more common among computer musicians where algorithms produce music on the fly, while the musician can manipulate these in real-time. We can define such use as *computer-as-instrument*. In this approach the computer is just a tool that relies on human agency. Blackwell *et al.* [13] proposed the concept of *Live algorithms for music* (LAMs) as a way of analyzing live music performance systems that, to some extent, exhibit autonomy or agency. NN Music is a system that follows this concept: it is a performer-machine system for Max/MSP[1] that trains a feed-forward neural network mapped to stochastic processes for musical outputs [14]. Another example, which also uses Genetic Algorithms (GAs), is the system described by Bown and Lexter for generative and interactive musical performance [4]. While these systems present similarities with the described system, exactly because they follow the same LAM philosophy, our system presents a novel approach with co-evolution of different "voices" (the modules).

C. NeuroEvolution of Augmenting Topologies

PRIMAL-IMPROV uses artificial neural networks to produce music, these are evolved using the NeuroEvolution of Augmenting Topologies (NEAT) genetic algorithm developed by Ken Stanley [15]. The novelty of the algorithm is that it evolves both the weighting parameters and structures of networks, attempting to find a balance between the fitness of evolved solutions and their diversity. Three main techniques are key to this algorithm: i) tracking genes with history markers to allow crossover among topologies, ii) applying speciation to preserve innovations, and iii) developing topologies incrementally from simple initial structures.

The specific artificial neural network structure we use is called Compositional pattern-producing networks (CPPN) [16]. CPPNs differ from typical artificial neural networks in their set of activation functions and how they are applied. These kind of ANNs, combined with NEAT, have been used in various research in creating art, including music [17], [18], images [19], [20], and more [21].

D. Co-evolution: cooperative/competitive

Co-evolution is a process of reciprocal genetic change in one species in response to another; this process can be observed in nature where each party exerts selective pressures on the other, as already observed by Darwin in the evolutionary interactions between flowering plants and insects [22]. Co-evolution has also been applied to evolutionary algorithms and in general is divided in two categories: as a competitive arms race [23] or as a cooperative effort. An example of the former is the co-evolution of test cases for a problem (predators) with solutions (prey) [24]. Conversely, in cooperative approaches, the populations can be seen as components of the final solution [25], [26].

An interesting application of co-evolution applied to music can be found in *Living Music* [27], a system where agents are populating a virtual world and have to produce sounds to mate. The aggregation of the sounds leads to the music produced by the system. This system shares the philosophy that for a system to be creative it should create structures that one cannot immediately envision, and that from simple rules complex behaviors can often arise. A key difference between the objective of *Living Music* and the described system is that our focus is on the interaction between human composed music and the system, while *Living Music* creates completely autonomous music.

E. Related systems: MetaCompose, MaestroGenesis

This section clarifies the connections between PRIMAL-IMPROV and two related systems: METACOMPOSE and *MaestroGenesis*. The METACOMPOSE music generator is a compositional, extensible framework for affective music composition [28], [29]. PRIMAL-IMPROV is designed to also be a possible replacement for this "improvisational" part of METACOMPOSE, which is currently delegated to simple human-made algorithms. *MaestroGenesis* [2] is also worth discussing more in depth as it was a great inspiration for PRIMAL-IMPROV. *MaestroGenesis* is an implementation of a computer-assisted approach to music generation called functional scaffolding for musical composition (FSMC), whose representation facilitates creative combination, exploration, and transformation of musical ideas and spaces. Music in FSMC is represented accordingly as a functional relationship between an existing human composition, or scaffold, and a generated set of one or more additional musical voices. Through *MaestroGenesis* a human can explore how the generated voices relate to the scaffolding through an interactive evolution process. Though PRIMAL-IMPROV shares some characteristics of *MaestroGenesis*, it is an autonomous system where the interaction between user and the program is not as explicit.

III. PRIMAL-IMPROV

PRIMAL-IMPROV is an evolutionary system which uses NEAT in combination with co-evolution to create a real-time (and offline) improvisational system. In the current implementation of Primal-Improv two modules are evolved, but the structure of the system will allow for an arbitrary amount of these to be added, possibly creating a very large and diverse instrumentation. The system is developed in C#, using the *SharpNeat*[2] library and the *C# MIDI toolkit*[3].

A. Module architecture

A module is defined by a collection of components needed to produce a monophonic melody in response to the environment's

[1]http://www.cycling74.com

[2]http://sharpneat.sourceforge.net/

[3]https://www.codeproject.com/articles/6228/c-midi-toolkit

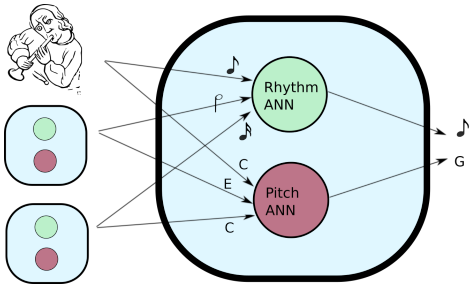

Fig. 1. Module components: the rhythm ANN that controls the durations of the notes, and the pitch ANN that controls what pitch the new notes should have. Each module *listens* (has as inputs) to the human-made melody and to all the other modules to decide what to play (output).

state (see Figure 1). Each module "listens" to both the human and all the other modules currently active, and according to such inputs, decides what to play next.

Rhythm neural network: this ANN controls the duration of the notes played by the module. The note duration is represented as a decaying function, where the decay depends on the length of the note (see the *Representation* subsection for a detailed explanation). The ANN has as many inputs as the number of currently active modules, plus an additional one for the human player. It presents only one output, which can be represented as the same type of function as those in input. A real-time peak recognition algorithm (described below) is applied to this function to identify when a note should end, in order to allow the next note to begin.

Pitch neural network: this ANN determines the pitch that should be played when the *rhythm ANN* determines that a new note should be played. The ANN has as many inputs as the number of currently active modules + one additional input for the human player. These inputs are a representation of the pitches being currently played. Note that the pitches are represented as absolute pitches (without octave information), meaning that a C4 and a C6 would have the same representation.

B. Evolutionary Algorithm

In this section we describe the function of the GA, with the assumption that only two modules are being evolved, yet the structure should be expandable to an arbitrary amount of these, in the *Conclusion* section we discuss what kind of challenges will need to be addressed to achieve this.

For each module the system has to run two instances of a NEAT GA: one for the *Rhythm ANN* and one for the *Pitch ANN*. In the described case, this means the system is running four separate evolutionary algorithms. These four are almost identical, apart from the fitness functions, which differ between *Rhythm* and *Pitch* ANNs. As stated before, this is a cooperative co-evolutionary algorithm, which means that the individuals of the four populations are not evaluated by themselves: individuals are chosen from each of the populations, the sampled ANNs are used together to produce music, this is evaluated, and the computed fitness value is propagated to the ANNs.

To illustrate via a practical example: each generation the population of *Pitch ANNs for module 1* will be used in conjunction with the highest scoring *Pitch ANNs for module 2* to create some outputs using the following formula:

$$f(P_i^{'}) = \frac{\sum_{j=1}^{n} f_{Pitch}(P_i^{'}, PChamp_j^{''})}{n},$$

where f is the fitness function of a specific individual, $P_i^{'}$ is an individual i of the *module 1* Pitch GA, f_{Pitch} is the fitness evaluation of the pitches produced by the input ANNs, and $PChamp_j^{''}$ is one of the n champions (best individuals from the previous generations) of the *module 2* Pitch GA.

Likewise, the fitness for specific individual of the Rhythm GAs is calculated as:

$$f(R_i^{'}) = \frac{\sum_{j=1}^{n} f_{Rhythm}(R_i^{'}, RChamp_j^{''})}{n},$$

where $R_i^{'}$ is an individual i of the *module 1* Rhythm GA, f_{Rhythm} is the fitness evaluation of the durations produced by the input ANNs, and $RChamp_j^{''}$ is one of the n champions of the *module 2* Rhythm GA.

The fitnesses for the *module 2* GAs are constructed analogously using the champions of the *module 1* GAs.

As stated earlier, the fitness functions are calculated on the outputs of the Rhythm ANNs (and Pitch ANNs) of all the modules. These fitnesses are calculated using a human-made reference phrase, in the real-time application: this is a recording of the last notes played by the user, while in the static case it uses the first five notes of a provided piece. We decided to keep these functions simple to observe emergent behaviors and give more freedom to the ANNs.

The fitness for the Rhythm ANNs is calculated as:

$$f_{Rhythm}(R_i^{'}, RChamp_j^{''}) = -(Few + SameDuration),$$

where *Few* is a function that measures if too few notes have been produced (in the current implementation the minimum threshold is three), and *SameDuration* is a function that discourages the modules from producing the same rhythmic pattern.

The fitness for the Pitch ANNs is calculated as:

$$f_{Pitch}(P_i^{'}, PChamp_j^{''}) = -(SamePitch + Identity + OutOfKey),$$

where *SamePitch* is a function that discourages the modules from producing the same pitches, *Identity* counts the number of times the pitches produced by each module are the same, and *OutOfKey* counts the times the produced pitches are out of key, with respect to a predefined key.

SharpNeat represents neural networks as collection of nodes and weights connecting the nodes. As SharpNeat has not been changed in this project, we refer the reader to the SharpNeat page for more details.

What is more interesting is how the system interprets inputs and outputs: the Pitch ANNs translate the absolute pitches in values in the range $[0, 1]$: $PitchInput(P) = P/12$, where P is an absolute pitch, which can range from $[0, 11]$ and $\in N$, this

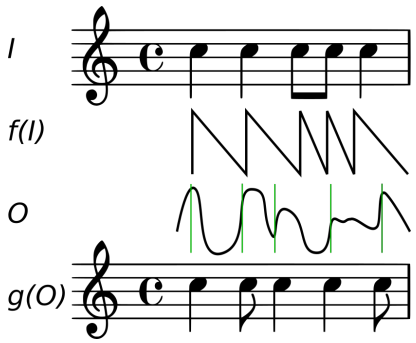

Fig. 2. Representation of rhythm, from the top: input rhythm, decaying function $f(I)$ representing the input, output of the network (in green the recognized peaks), output rhythm $g(O)$.

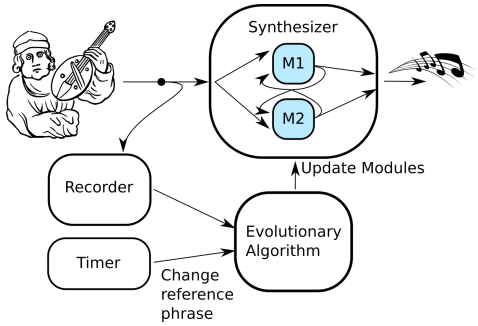

Fig. 3. Representation of the Real-time architecture of Primal-Improv. The human player is recorded by a component which periodically updates the reference phrase driving the evolutionary algorithm. At the same time the modules in the GA are passed to the synthesizer to use until the next time interval has passed.

procedure divides the interval in twelve parts, each of them representing one of the twelve notes commonly used in Western music. In this way a function to interpret the outputs can easily be determined: $\mathit{PitchOutput}(o) = 12o$, where $\mathit{PitchOutput}(o) \in N$, o is the output value of the ANN which has range $[0, 1]$.

Durations are instead represented as a sawtooth function, which generates a spike that decays according to the duration of the note (Figure 2). This is a function of time, which is represented as MIDI ticks (24 per quarter note). This representation is inspired by Hoover's *functional scaffolding for composing additional music voices* [2], which allows the networks to learn when new notes are played while allowing for more freedom in creating non-standard durations. The decay corresponding to the input duration can be calculated as:

$$\mathit{Decay} = \frac{1}{\mathit{duration} \cdot PPQN \cdot 4},$$

where PPQN is the ticks (or pulses) per quarter note.

Once the decay is calculated, the values for each tick can be found by subtracting the decay from the previous value: $x_n = x_{n-1} - \mathit{Decay}$.

The Rhythm ANN outputs a value for each tick, and peaks in this function are interpreted as the start of a new note. The peak detection algorithm is called *Smoothed z-score*[4]. It is based on the principle of dispersion: when a data-point is x standard deviations away from the moving mean a new peak is signaled. The algorithm creates a separate moving mean and standard deviation, meaning that the signals do not risk corrupting the threshold.

C. Real-Time

In the real-time application of the system the user is able to use a computer keyboard to play notes; as notes are played the modules receive them as inputs for the Rhythm and Pitch ANNs and calculate if and what notes to play. As the user is playing, a *recording* module keeps a queue of the last notes played and, after a predefined passage of time, a message is sent to update the reference phrase in the evolutionary algorithm with said recording. At the same time, as the reference phrase is updated, the current best individuals are pushed to the synthesizer where they will replace the previous best individuals. The program allows for minimal user control over the system: mainly switching on/off the modules. The current implementation keeps the last ten notes played by the user and switches the reference phrase every ten seconds.

[4]https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data/22640362

TABLE I

NUMBER OF CORRECT, INCORRECT AND NEUTRAL ANSWERS TO OUR CRITERIA FOR THE COMPLETE SYSTEM AGAINST THE VERSION WITH RANDOM FITNESS FUNCTION. ALSO INCLUDED THE p-VALUES CALCULATED USING A TWO-TAILED BINOMIAL TEST AND THE BINOMIAL EFFECT SIZE DISPLAY (BESD). NOTE THAT IN THE CASE OF THE RANDOM CRITERIA THE LISTENER IS ASKED TO SELECT THE CLIP THAT HE/SHE FEELS THE MOST RANDOM, SO IT IS EXPECTED THAT LESS PEOPLE PREFERRED THE COMPLETE SYSTEM.

Choice	Pleasing	Random	Interesting	Harmonious
Preferred the complete system	307	185	261	294
Preferred the random fitness	151	226	151	164
Neutral answer	77	124	123	77
Total non-neutral answers	458	411	412	458
Binomial test p-value	2.60E-13	0.04836	6.62E-08	1.30E-09
BESD	34.10%	-10%	26.70%	28.40%

IV. EVALUATION

This section describes a quantitative user study designed as a preliminary evaluation of the musical product of the system. As music is (at least partly) a subjective matter, we evaluate it through a preference questionnaire to listeners distributed over the Internet. Participants are asked to indicate which of two pieces of music they prefer; one of these is generated with the complete system (as described in the previous section) and the other using a random fitness function. We realize that comparing against a random fitness function doesn't prove music quality, but allows us to identify (i) if the emergent music patterns are caused by the architecture of the system and (ii) if the introduction of small evolutionary pressure is enough to perceptively improve the result. Quality is evaluated according

to four criteria: *pleasantness*, *randomness*, *harmoniousness* and *interestingness*. We chose these criteria to cover different aspects or dimensions of perceived quality of music. Our intended interpretation of the criteria is presented below. Note that no definition of these terms is offered in the survey, and there is therefore no guarantee that participants interpret these criteria the same way we do.

Pleasantness intends to measure how pleasing to the ear the piece is, but this alone is not sufficient to describe the quality of the music produced. There are countless pieces of music that do not sound pleasant, but may nonetheless be considered by the listener as "good" music. In fact, in music, often uncommon (and even discordant) chord sequences or intervals are introduced to express different features in a score, such as affect, as well as other narrative information. Also note that some alterations or passages can be specific of a music style or genre. Moreover, discordant intervals are more acceptable to the ear the more often they are repeated (see dodecaphonic music [30] for example).

Interestingness is introduced to overcome the just described limitations of the *pleasantness* criteria: in this way we intend to test if one of our "broken" scores might introduce something more interesting to the listener, even when the composition is not necessarily pleasant or harmonic. Note that this is a very subjective measure, as most people have a different opinion about how interesting they perceive a score to be.

On the other hand, *harmoniousness* might be confused with pleasantness, but we hope that it will be seen as a somewhat more objective measure: less of a personal preference and more of a measure of the listener's ability to recognize the presence of dissonances and harmonic passages.

Finally, *randomness* intends to gather a measure of how structured the music sounds to the listener. It is not only a measure of dissonance, but also of the extent the music seems to have a cohesive quality and coherent internal structure. Examples of coherent internal structure are: (i) voices working together well (ii) coherent rhythmic structure (iii) chord sequences presenting tension building and eventual resolution.

An online survey was developed with HTML and PHP, using a MySQL database to hold the data collected. Participants were presented with pairs-wise music clips and asked to evaluate them using the four criteria described. Each of the four criteria has a multiple choice question structured as:

> Which piece do you find more pleasing? *"Clip A"/"Clip B"/"Neither"/ "Both Equally"*

Where the last word (e.g. "pleasing") is dependent on the criteria. We also include the more neutral answers "Neither" and "Both Equally" to avoid randomness in the data from participants who cannot decide which clip satisfies their evaluation, according to the criteria better or worse. Other benefits of doing this are: avoiding participant frustration, and giving us potential information on interesting individual pairs, where the pieces are considered equally good/bad.

***Music Piece Generation**:* Eight clips were created for each group (*normal* and *random fitness*), for a total of 16 pieces[5]. The system was allowed to evolve for 100 generations, using the first ten notes of the melody as a reference point for the evolution. The clips used in the experiment were not selected from a pool, but were the result of a singular execution of the program. The melodies used are a mix of classical melodies, traditional songs and one melody composed by the author.

***Results and Analysis**:* The data collected amounts to 538 answers for each of the four evaluation criteria from 87 participants. Table I shows how many responses were obtained for each criteria and how many neutral answers were collected. For now we only consider definitive answers (i.e. the participant chooses one of the music clips presented); we will look at the impact of the neutral answers at the end of this section. Under the definite choice constraint, the data becomes boolean: the answers are either *"user preferred the complete system"* or *"user preferred the random fitness function"*. To analyze this data we use a two-tailed binomial test, which is an exact test of the statistical significance of deviations from a theoretically expected random distribution of observations in the two categories. The null hypothesis is that both categories are equally likely to occur and, as we have only two possible outcomes, that probability is 0.5. The Binomial Effect Size Display (BESD) [31] is another way of looking at the effects of treatments by considering the increase of success through interventions. This is an interesting measure, as it elucidates how much of an effect is created, in our case, by the introduction of even very simple fitness functions.

As can be seen in Table I, there is a strong statistical significance ($p < 0.05$) for all the criteria, although there is a clearly smaller effect on the *randomness* criteria ($p \approx 0.048$) This means that the null hypothesis can be refuted and a difference in distribution can be inferred between choosing the music generated with (and without) the described fitness functions.This indicates that music generated by the complete system is considered more pleasing, less random, more interesting and more harmonious than music generated by the degenerate system with random fitness. The BESD values reflect what can be inferred from the p-values; it can be observed that for the *randomness* criterion the effect is small (-10%) confirming that, while the p-value is significant, people can not easily distinguish emergent musical structures. This result could be explained by the high freedom left to the ANNs and by the short size of the generated pieces, which might not have been long enough for participants to discern emergent patterns and make a judgment.

***Demographics**:* Our participant's population is composed by 62 males, 14 females, and 7 people that did not specify their gender. The average age is 37.46 (stdev 17.72). Participants were asked to rate their skill with a music instrument and their knowledge of music theory according to a five point Likert scale (0 to 4). The participants reported a similar level of musical training (avg: 1.11, stdev: 1.01, mode: 0) and instrument skill (avg: 1.36, stdev: 1.19, mode: 0). The homogeneity of

[5]The list and the generated music for the various groups can be accessed at *https://goo.gl/2cqZL9*.

the population may explain how, however we partition the population, we find no significant difference in the results.

V. Conclusions

This paper describes a system for co-evolution of melody-producing modules, which can work both as an arranger for a predefined melody, or a real-time improvisational system. The main features of the system are: emergence of musical structures out of very simple rules, not requiring direct human input, and adaptiveness to a human musician. A quantitative user study has been described, comparing the complete system with one using a random fitness function. As described in the *Results and Analysis* session we have shown a statistical significant preference in all criteria (*interestingness*, *less randomness*, *pleasingness* and *harmoniousness*) for the complete system. This shows that the fitness function evaluated, while extremely simple, does guide the evolution of the modules to create something more interesting than the networks themselves could otherwise produce. Clearly this does not prove that the system actually produces interesting music, but we believe it is a first step in showing the potential of PRIMAL-IMPROV.

This system is still in its infancy and there are many improvements that can be implemented. An example is that the current implementation of the system is limited to two modules, it would be interesting to find out how the system reacts with a larger number. The fitness functions will likely require adjustment to avoid a cacophony effect, due to too many modules playing simultaneously. An interesting phenomenon that appears in some of the music pieces created for the user study is a "ringing" effect (like an old time phone): this is due to the neural networks producing very short notes (1/96th of a measure) repeatedly. This is a very "non-human" way of playing and, while emulating a human player is not an objective of the system, this effect seems to be negatively perceived by listeners, so it might need to be addressed and eliminated. We plan to conduct qualitative experiments with musicians playing with the system in real-time, in order to find out how the interaction between the players unfolds, and what kind of creative improvisations can emerge.

In summary, we present (i) a novel co-evolutioary system for improvisation and melody arrangement and (ii) a quantitative study showing how even from very simple rules some interesting content seems to emerge.

References

[1] P. M. Todd and G. M. Werner, "Frankensteinian methods for evolutionary music," *Musical networks: parallel distributed perception and performace*, pp. 313–340, 1999.

[2] A. K. Hoover, P. A. Szerlip, and K. O. Stanley, "Functional scaffolding for composing additional musical voices," *Computer Music Journal*, 2014.

[3] J. A. Biles, "Genjam: A genetic algorithm for generating jazz solos," in *ICMC*, vol. 94, 1994, pp. 131–137.

[4] O. Bown and S. Lexer, "Continuous-time recurrent neural networks for generative and interactive musical performance," in *Workshops on Applications of Evolutionary Computation*. Springer, 2006, pp. 652–663.

[5] E. R. Miranda, *Readings in music and artificial intelligence*. Routledge, 2013, vol. 20.

[6] M. Edwards, "Algorithmic Composition: Computational Thinking in Music," *Commun. ACM*, vol. 54, no. 7, pp. 58–67, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/1965724.1965742

[7] D. Cope, "Algorithmic Music Composition," in *Patterns of Intuition*, G. Nierhaus, Ed. Springer Netherlands, 2015, pp. 405–416, dOI: 10.1007/978-94-017-9561-6_19.

[8] R. Wooller, A. R. Brown, E. Miranda, J. Diederich, and R. Berry, "A framework for comparison of process in algorithmic music systems," in *Generative Arts Practice 2005 — A Creativity & Cognition Symposium*, 2005.

[9] R. Loughran, J. McDermott, and M. O'Neill, "Tonality driven piano compositions with grammatical evolution," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 2168–2175.

[10] P. Dahlstedt, "Autonomous evolution of complete piano pieces and performances," in *Proceedings of Music AL Workshop*. Citeseer, 2007.

[11] E. R. Miranda and A. Biles, *Evolutionary computer music*. Springer, 2007.

[12] F. Pachet, "The continuator: Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[13] T. Blackwell, O. Bown, and M. Young, "Live algorithms: towards autonomous computer improvisers," in *Computers and Creativity*. Springer, 2012, pp. 147–174.

[14] M. Young, "Nn music: improvising with a livingcomputer," in *International Symposium on Computer Music Modeling and Retrieval*. Springer, 2007, pp. 337–350.

[15] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[16] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic programming and evolvable machines*, vol. 8, no. 2, pp. 131–162, 2007.

[17] P. A. Szerlip, A. K. Hoover, and K. O. Stanley, "Maestrogenesis: Computer-assisted musical accompaniment generation," 2012.

[18] B. T. Jónsson, A. K. Hoover, and S. Risi, "Interactively evolving compositional sound synthesis networks," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 321–328.

[19] J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley, "Picbreeder: evolving pictures collaboratively online," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008, pp. 1759–1768.

[20] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley, "Combining search-based procedural content generation and social gaming in the petalz video game." in *Aiide*. Citeseer, 2012.

[21] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 167–174.

[22] C. Darwin, "The origin of species by means of natural selection: or, the preservation of favored races in the struggle for life," 1859.

[23] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary computation*, vol. 5, no. 1, pp. 1–29, 1997.

[24] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 228–234, 1990.

[25] C. C. Coello and M. R. Sierra, "A coevolutionary multi-objective evolutionary algorithm," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 1. IEEE, 2003, pp. 482–489.

[26] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary computation*, vol. 8, no. 1, pp. 1–29, 2000.

[27] P. Dahlstedt and M. G. Nordahl, "Living melodies: Coevolution of sonic communication," *Leonardo*, vol. 34, no. 3, pp. 243–248, 2001.

[28] M. Scirea, J. Togelius, P. Eklund, and S. Risi, "Metacompose: A compositional evolutionary music composer," in *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 2016, pp. 202–217.

[29] ——, "Affective evolutionary music composition with metacompose," *Genetic Programming and Evolvable Machines*, pp. 1–33, 2017.

[30] G. Perle, *Serial composition and atonality: an introduction to the music of Schoenberg, Berg, and Webern*. Univ of California Press, 1972.

[31] R. Rosenthal and D. B. Rubin, "A simple, general purpose display of magnitude of experimental effect." *Journal of educational psychology*, vol. 74, no. 2, p. 166, 1982.