A Framework for Anticipatory Machine Improvisation and Style Imitation

Arshia Cont^{1,2}, Shlomo Dubnov^{2}, and Gérard Assayag^{1}

^{1} Ircam - Centre Pompidou - UMR CNRS 9912, Paris.
{cont,assayag}@ircam.fr
^{2} Center for Research in Computing and the Arts, UCSD.
sdubnov@ucsd.edu

Abstract. We present a first step towards anticipatory machine improvisation systems. The proposed system, based on fundamentals of music cognition, is a multi-agent memory-based collaborative and competitive reinforcement learning architecture, capable of live interaction with a musician or a music score. Results demonstrate the ability to model long-term stylistic planning and need for much less training data than reported in previous works.

1 Introduction

Musical style modeling consists of building a computational representation of the musical data that captures important stylistic features. Considering symbolic representations of musical data, such as musical notes, we are interested in constructing a mathematical model such as a set of stochastic rules, that would allow creation of new musical improvisations by means of intuitive interaction between human musicians or music scores and a machine.

About half a century ago, the musicologist Leonard Meyer drew attention to the importance of *expectation* in the listener's experience of music. He argued that the principal emotional content of music arises through the composer's choreographing of expectation [1]. Most musical improvisation systems (reviewed in section 2) define this expectation effect in terms of prediction process that is based on the past. In this paper we extend this view using the concept of *Anticipation*. Anticipation, in short, is different from prediction in the sense that an anticipatory system is "a system containing a predictive model of itself and/or of its environment, which allows it to change state at an instant in accord with the model's predictions pertaining to a later instant" [2].

In this paper, we present an anticipatory machine improvisation system capable of formal long-term planning of musical material and which handles multiple representations of music signals at the same time. In section 2 we provide pertinent literature background on previous works done on machine improvisation and style modeling, fundamentals of music cognition for this work and some literature review on anticipatory systems. The proposed architecture is based on reinforcement learning with multiple-agents in interaction with an environment, which we overview in section 3. Each agent handles a specific musical attribute and they all collaborate and compete during learning and event generation. Section 4 introduces the representation of musical material in our multi-agent architecture, followed by the definition of interaction and reward and the notion of *guides* in section 5. Section 6 explains the proposed learning algorithm to achieve memory-based collaborative and competitive behavior in a multi-agent structure and finally, we present some result and analysis, followed by discussions and future development of the system.

2 Background

2.1 Machine Improvisation and Style Modeling

Earlier works on style modeling employed information theoretical methods inspired by universal prediction. In many respects, these works build upon a long musical tradition of statistical modeling that began in the 50s with Hiller and Isaacson's "Illiac Suite" [3] and Xenakis using Markov chains and stochastic processes for musical composition [4]. The most prevalent type of statistical model encountered for music are *predictive* models based on *context*; referred to as *context models* implying general Markov models [5]. Universal prediction methods improved upon the limited memory capabilities of Markov models by creating context dictionaries from compression algorithms, specifically using the Lempel-Ziv incremental parsing (IP) [6], and employing probability assignment according to Feder et al. [7]. Music improvisation was accomplished by performing a random walk on the phrase dictionary with appropriate probabilistic drawing among possible continuations [8–10]. Later experiments explored Probabilistic Suffix Tree (PST) [11], and more recently in [12] using Factor Oracle (FO) [13]. Other methods include the use of Genetic Algorithms [14] and neural networks [15] just to name a few. This last group require elaborate training and intricate supervision schemes due to their architecture.

One of the drawbacks of the above methods is lack of responsiveness to changes in musical situations that occur during performance, such as dependence of musical choices on *musical form* or changes in *interaction* between players during improvisation. Interaction has been addressed previously in [10] for PST based improvisation by means of a fitness function that influenced prediction probabilities according to an ongoing musical context, with no consideration of planning or adaptive behavior. Statistical approaches seem to capture only part of the information needed for computer improvisation, i.e. successfully modeling a relatively short term stylistics of the musical *surface*. Although variable Markov length and universal methods improve upon the finite length Markov approach, they are still insufficient for modeling the true complexity of music improvisation.

Another significant challenge faced with music signals arises from the need to simultaneously represent and process many attributes of music information. Previously, this problem has been tackled by *cross-alphabet* models [9, 12, 10] where each parsed signal represents an alphabet as a vector of multiple attributes and *multiple viewpoints* [16], by deriving individual expert models for any given representational viewpoint and then combining the results obtained from each model. Cross alphabet methods have proven to be useful for single attribute information types such as text, and are intractable when the dimension of attributes increases as in the case of music. Researchers have considered various membership functions to allow for these context dependencies through various heuristics [12, 10]. Such heuristics might make the system dependent upon the style of music being considered or reduce generalization capabilities. Multiple viewpoint models are more expressive than cross-alphabet models since by combining models we allow the system to reach parts of the hypothesis space that the individual models would not be able to reach. However, learning requires huge amounts of training data and this representation might be extremely redundant in view of the *repetitive* structure of music information.

2.2 Psychology of music expectation

Expectations imply some sort of *mental representation*. On the other hand, one of the main characteristics of music information is its multi-dimensional aspect. Huron, in his recent book, suggests that brain uses a combination of several underlying representations and expectation plays a major role in realizing which representation to use, and provides evidence for a system of rewards and punishment that evaluates the accuracy of our unconscious predictions about the world [17]. Our mental representations are being perceptually tested by their ability to usefully predict ensuing events, suggesting that *competing and concurrent representations* may be the norm in mental functioning. This view is also strongly supported by the *neural Darwinism* theory of Edelman [18]. According to Huron, it is during the *prediction response* that the reinforcement feedback is provided and serves at least three functions: *motivation*, *preparation* and *representation*.

Another concept in music perception important for this work is the role of memory. Snyder in [19] proposes an auditory model of memory that consists of several layers, from which we consider *feature extraction*, Long Term Memory (LTM) and Short Term Memory (STM). Feature extraction is some sort of perceptual categorization and grouping of data. Events processed at this stage can activate those parts of LTM activated by similar events in the past. Not all LTM at this point become conscious, but form a *context* for current awareness. This context takes the form of *expectations* that can influence the direction that current consciousness takes. LTM acts like a filter determining which aspects of our environment we are aware of at a given time. LTM that reaches this higher state of activation can then persist as current STM. Information in STM might be repeated or rehearsed. This rehearsal greatly *reinforces* the chances that the information being circulated in STM will cause modifications in permanent LTM.

2.3 Anticipatory Systems

There has been collective interest in the recent years on simulating anticipatory behavior in robotics and animat design among others [20]. These works are commonly based on foundations from psychological research of behaviorists and/or latent learning in animals. In general, anticipatory behavior research is interested in systems that base their action decisions on future predictions as well as past inference and simulate adaptive frameworks in the light of different anticipatory behavior in animals [21].

Davidsson in [22] proposes a framework for preventive anticipation where he incorporates collaborative and competitive multiple agents in the architecture. While this has common goals with the system proposed in this paper, our proposals are different since the system in [22] uses rule-based learning with ad-hoc schemes for collaboration and competition between agents, which is not the case here. Recently, in the computer music literature, Dubnov has introduced an anticipatory information rate measure that when run on non-stationary and time varying data such as audio, can capture anticipatory profile and emotional force data that has been collected using experiments with humans [23].

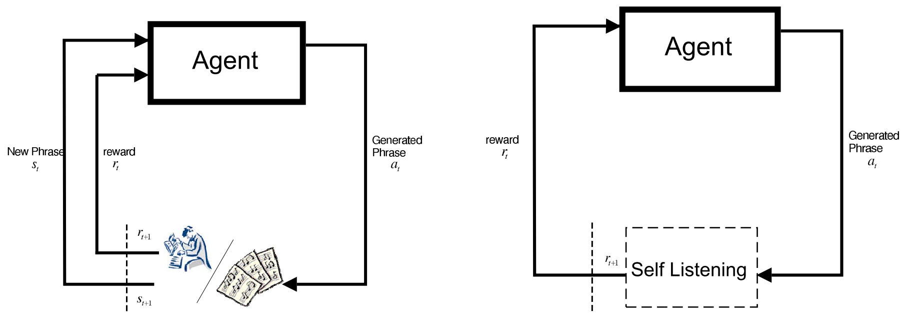
3 General Architecture

The most common methods to simulate anticipatory behavior has been reinforcement learning (RL) and learning classifier methods [20]. Here, we adopt an RL view of musical interaction in an anticipatory framework.

3.1 RL view of Musical Interaction

In a RL system, *rewards* are defined for goal-oriented interaction. In musical applications, defining a *goal* would be either impossible or would limit the utility of the system to certain style. In this sense, the *rewards* used in our interaction are rather *guides* to evoke or repress parts of the learned model in the memory. We define two execution modes for our system demonstrated in Figure 1. In the first, referred to as *Interaction mode*, the system is interacting either with a human performer (live) for machine improvisation or with music score(s) for style imitation and occurs when external information is being passed to the system from the environment (either human improvisor or music score). During the second mode, called *self listening mode*, the system is in the generation phase and is interacting with itself, or in other words it is listening to itself, in order to decide how to proceed.

The agents in both modes are model-based RL frameworks. They include an internal model of their environment plus a reinforcement learner for planning. This internal model plays the role of *memory* and *representation* of input sequences from the environment. Our model, presented in section 4, leads to multiple agents which collaborate and compete during learning and planning. This aspect is quite important since it reduces the cost of learning and storage, overcomes the curse of dimensionality, and makes it possible for the system to interact in realtime with small data available. During the *self-listening* mode reinforcement learning is model-free, meaning that the internal model of the environment rests intact but the planning is influenced by the latest musical sequence that has been generated by the system itself, thus, the idea of *self listening.*



(a) Interaction Mode (b) Self Listening Mode

Fig. 1. Machine Improvisation modes diagram

3.2 Anticipatory Framework

At each interaction with the environment, agents receive *immediate* scalar rewards, r, which serve as *guides* in stimulating the internal memory of agents (section 5). This way the *reward* would be the manner in which this new information reinforces the current stored model in the agent. In an anticipatory system, we are interested in the impact of future predictions on the current state of the system. In a RL framework, this means that the reward for a state-action pair would correspond to future predicted states. Equation 1 demonstrates the RL reward used in different stages of our system to be discussed in section 6 with γ as a discount factor and s_t as the chosen state by the system at time t.

$$R(s_t) = \sum r(s_t) + \gamma r(s_{t+1}) + \cdots + \gamma^m r(s_{t+m}) + \cdots \tag{1}$$

Rewards or guides are calculated the same way for both modes of the system described before with a small difference. We argue that the rewards for the *interaction mode* (Figure 1(a)) correspond to a *psychological attention* towards appropriate parts of the memory and guides for the *self-listening mode* (Figure 1(b)) correspond to a *preventive* anticipation scheme. This means that during interaction with a human improvisor or a score, the system needs to be attentive to (new) input sequences from the environment and during self-listening in needs to be preventive so that it would not generate the same (optimal) path over and over. This is achieved by treating the same environmental rewards with positive and negative signs appropriately. Following the distinctions of Butz et al. in [21], the proposed framework attempts to model two types of anticipations: *state anticipation* in which predictions about future states directly influence current behavioral decision making, and *payoff anticipation* where the influence of future predictions on behavior is restricted to payoffs during self-listening.

4 Musical Representations

In this section, we present how the music data is represented and stored in the agent. The input of the system is polyphonic MIDI signals, sliced vertically between successive onsets. The unit of time used for the system is assumed to be the smallest significant time interval that a musical event would occupy during a piece (referred to as *tatum*). In our multiple-agent framework, each agent learns and stores a model for a specific attribute of music signal using the *Factor Oracle (FO)* algorithm [13]. FO has been previously used for machine improvisation using cross-alphabets in [12].

We give a short description of the properties and construction of FO and leave the formal definitions in [13]. Basically a factor oracle is a finite state automaton constructed in linear time and space in an incremental fashion and can be learned online. A sequence of symbols $A = a_1 a_2 \cdots a_n$ is learned in such an automaton, whose states are $S_0, S_1, S_2 \cdots S_n$. There is always a transition arrow labeled by symbol a_i going from state S_{i-1} to state S_i. Depending on the structure of A, other arrows will be added to the automaton. Some are directed from a state S_i to a state S_j, where $0 \leq i < j <= n$. These also belong to the set of transitions and are labeled by symbol a_j. Some are directed *backward*, called suffix links, and bear no label. The transitions model a factor automaton, that is every factor p in A corresponds to a unique transition path labeled by p, starting in S_0 and ending in some other state. Suffix links connects repeated patterns of A. In general, given a sequence, the constructed FO returns two *deterministic* functions: a transition function $F_{trn} : S \times A \rightarrow \{S \cup \emptyset\}$ and suffix links $F_{sfx} : S \rightarrow \{S \cup \emptyset\}$.

For this experiment, we use 6 different attributes taken out of musical events which consist of `pitch`, `duration` in quantized beats and `harmonic interval(s)` relative to a leading pitch and their first derivatives. Note that while the derivatives of pitch and harmonic interval are subtractive, the derivative of duration would be multiplicative. Upon the arrival of a sequence, these attributes are extracted and would update corresponding FOs. Table 1 shows 6 attributes computed for the score shown in Figure 2 with four sample learned FOs in Figure 3.

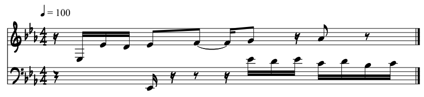
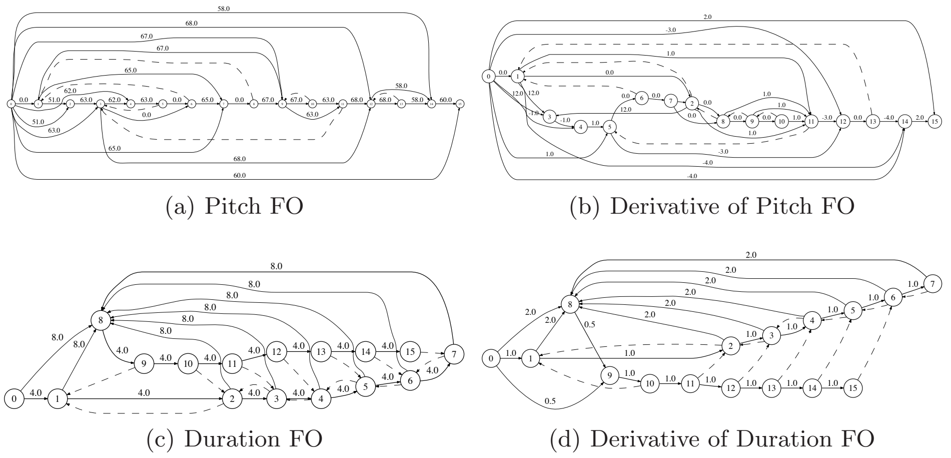


Fig. 2. First bar of J.S.Bach's *two-part Invention* No.5

Event Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pitch (MIDI)	0	51	63	62	63	0	65	0	67	67	63	68	68	58	60
Harmonic Int.	0	0	0	0	24	0	0	0	4	5	0	8	6	0	0
Duration	4	4	4	4	4	4	4	8	4	4	4	4	4	4	4
Pitch Deriv.	0	0	12	-1	1	0	0	0	0	0	1	-3	0	-4	2
Harm. Deriv.	0	0	0	0	0	0	0	0	0	1	0	0	-2	0	0
Dur. Deriv.	1	1	1	1	1	1	1	2	0.5	1	1	1	1	1	1

Table 1. Attributes for parsed events in Figure 2



(a) Pitch FO (b) Derivative of Pitch FO

(c) Duration FO (d) Derivative of Duration FO

Fig. 3. Factor Oracles learned over sequences in Table 1. Each node represents a state, each solid line a *transition* and dashed line a *suffix link*.

5 Interactions and Rewards

When a new music sequence $O^t = o_1 o_2 \cdots o_N$ is received from the environment, an ideal reward signal should reinforce the part of memory which most likely evoke the sequence received to be able to generate recombinations or musically meaningful sequences thereafter. In the RL framework, this means that we want to assign numerical rewards to *transition states* and *suffix states* of an existing Factor Oracle with internal states s_i. Reward computation occurs *before* integrating the new sequence into the model.

After different attributes of O^t is extracted as separate sequence each in form $\{x_1 \dots x_N\}$ (as in Table 3), we use a *probability assignment function* P from $S^* \rightarrow [0,1]$ (where S^* is the set of all tuple of states available to FO) to assign rewards to states in the model as follows:

$$P(s_{1^*} s_{2^*} \dots s_{N^*} | S^t) = \left[\sum_{i=1}^{N} p(x_i | s_{i^*}) \right] / N \tag{2}$$

where

$$p(x_i|s_i) = \begin{cases} 1 \text{ if } F_{trn}(s_{i-1}, x_i) = s_i \\ 0 \text{ if } F_{trn}(s_{i-1}, x_i) = \emptyset \end{cases}, \; s_0 \equiv \{\forall s_t \in S^{span} : F_{trn}(s_t, x_1) \neq \emptyset\} \tag{3}$$

with s_0 as an initialization set for the search space, and S^{span} corresponding to a finite recent history of states emphasizing the *finite memory process* property of a finite state machine model such as FO.

To interpret Equation 2 by words, it reinforces the states in the memory that can most regenerate the new sequence. In other words, it will *guide* the learning described later to the places in the memory that should be mostly regarded during learning and generation. For example, for an attribute sequence $\{e_1, \ldots, e_N\}$, a sequence of state transitions for $\{s_{1^*}, \ldots, s_{N^*}\}$ in a FO structure would get a reward of 1 if they can regenerate all the original, and so on.

To assign rewards to suffix links, we recall that they refer to previous states with largest common suffix. Using this knowledge, a natural *reward* for a suffix link would be proportional to the *length* of the common suffix that the link is referring to. Fortunately, using a factor oracle structure, this measure can be easily calculated online and has been introduced in [24].

6 Interactive Learning

In our architecture, each musical event is defined by its corresponding state numbers in different FOs. For example, an event defined by a tuple `<pitch, duration, harmony>` gives indices to states in FOs for pitch, duration, harmony or their derivatives (see section 4). This way, the RL framework learns policies (Q matrices) over FOs' transition and suffix links. Correspondingly, this policy matrix will be used during music generation to activate the appropriate states in the model to generate certain music sequences or variants of them.

The main structure of the learning algorithm is based on a *Dyna* architecture [25] as a model-based learning with FOs used for representation and model learning. The main cycle of the algorithm consist of internal environment model (FOs) update and *Q-learning*. Planning in a musical system should be memory based, collaborative, competitive and anticipatory as discussed in section 2.2. In our application, each RL episode can be viewed as an *improvisation* simulation with the terminal state corresponding to the length of the simulation. In other words, during each learning episode, the system is *practicing* what it has learned based on updated rewards to learn new policies.

We present a method for combining multiple agents on the same domain and in parallel which uses the same experience knowledge and learns collaboratively and competitively inspired by [26]. As discussed in section 2.2, different mental representations of music work in a collaborative and competitive manner based on their predictive power to make decisions. This can be seen as kind of a *model selection* where learning uses all the models available and chooses the best one for each episode. This winning model would then become the *behavior policy* with its policy followed during that episode and other agents being influenced by the actions and environmental reactions from and to that agent.

At the beginning of each episode, the agent selects the policy of one module following the probability in Equation 4, with parameter β_{sel} being positive and the inverse temperature. Low β_{sel} causes equi-probable selection of all modules and vice versa. This way, a behavior policy π^{beh} is selected *competitively* at the beginning of each episode based on the *value* of the initial state s_0 among all policies π^i as demonstrated in Equation 4.

$$Pr(i|s_0) = \frac{e^{\beta_{sel} \sum_{a'} Q^i(s_0,a')}}{\sum_{j=1}^{N} e^{\beta_{sel} \sum_{a'} Q^j(s_0,a')}} \quad , \quad \pi^{beh} = \operatorname*{argmax}_i Pr(i|s_0) \tag{4}$$

During each learning episode, the agent would be following the behavior policy. For update of π^{beh} itself, we can use a simple Q-learning algorithm but in order to learn other policies π^i, we should find a way to compensate the mismatch between the target policy π^i and the behavior policy π^{beh}. Uchibe and Doya [26] use an *importance sampling* method for this compensation and demonstrate the implementation over several RL algorithms. Adopting their approach, during each update of π^i when following π^{Beh} we use a compensation factor $IS = \frac{Q^i(s_m,a_m)}{Q^*(s,a)}$ during Q-learning as depicted in Equation 5, where (s_m, a_m) are *mapped* state-action pairs of (s, a) in behavior policy to attribute i. This would define the *collaborative* aspect of our learning, with α as learning rate and $R(s_m)$ as in equation 1. Due to different natures of Q^is, we enforce an upper and a lower bound on IS.

$$Q^i(s_m, a_m) = Q^i(s_m, a_m) + \alpha \left[R(s_m) + \gamma \cdot IS \cdot \max_{a'}(Q^i(s_m, a')) - Q^i(s_m, a_m) \right] \quad (5)$$

In a regular Dyna agent, simulated transitions are started in state-action pairs selected uniformly at random from all previously experienced pairs. But a uniform selection is not the best and planning can be much more efficient if simulated transitions and backups are focused on *useful* state-action pairs. In general, we want to go back in the memory from any state whose value has changed. Equally as the frontier of useful backups propagates, not all of them will be equally useful. The value of some states may have changed a lot while others rest intact, suggesting that the predecessor pairs of those who have changed a lot are more likely to change a lot as well. So it is natural to prioritize the backups according to measures of their urgency and perform them in order of priority. This is the idea behind *prioritized sweeping* or *memory-based learning* [27] embedded in our learning with the priority measure as in Equation 6 for a current state s and next state s', leading to a priority queue of state-action pairs (chosen by a threshold θ) to be traced backwards during updates.

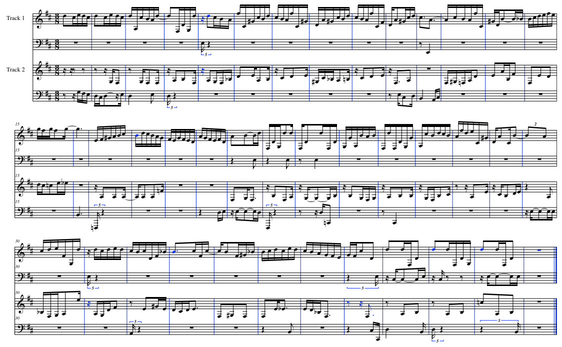
$$p \leftarrow |R(s) + \gamma \max_{a'}(Q^{Beh}(s', a')) - Q^{Beh}(s, a)| \quad (6)$$

7 Generation

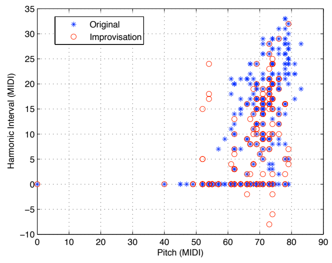
There are many ways to generate or improvise once the policies for each attribute are available. We represent just one simple solution using the proposed architecture. At this stage, the system would be in the *self listening mode* (Figure 1(b)). The agent would generate *phrases* of fixed length following a behavior policy (learned from the previous interaction). When following the behavior attribute, the system needs to *map* the behavior state-action pairs to other agents in order to produce a complete music event. For this, we first check and see whether there are any common transitions between original attributes and if not, we would follow the policy for their derivative behavior. Once a phrase is generated, its (negative) reinforcement signal is calculated and policies are updated as in section 6 but without updating the current models (FOs).

8 Results and Discussions

As a sample result for this paper, we include a sample run of the system on a single polyphonic piece, *two-part Invention* No.3, by J.S. Bach, for style imitation. For this example, the learning phase was run in *interaction mode* with a sliding window of 50 events with no overlaps over the original MIDI score. After the learning phase, the system entered *self listening* mode where it generates sequences of 20 events and reinforces itself until termination. Parameters used for this session were $\alpha = 0.1$, $\gamma = 0.8$, $\theta = 2$, and $\epsilon = 0.1$ for the *epsilon*-greedy of state-action selection. Number of episodes simulated during each RL phase was 100. The generated score is shown in Figure 4(a) for 240 sequential events (as described in section 4) where the original score has 348. For this generation, the *pitch* behavior has *won* all generation episodes and direct mappings of *duration* and *harmonic* agents have been achieved 76% and 83% in total respectively



(a) Style imitation sample result



(b) Improvisation vs. Original

Fig. 4. Machine Improvisation sample run results

While both voices follow a polyphonic structure, there are some formal musicological structures that can be observed in the generated score. Globally, there are *phrase* boundaries in measures 11 and 29 which clearly segment the score into three formal sections. Measures 1 to 11 demonstrate some sort of exposition of musical materials which are expanded in measures 12 to 29 with a weak cadence in measure 16. There are several thematic elements which are reused and expanded. For example, the repeated `A` note appearing in measures 3 and 4 appear several times in the score notably in measure 22 as high *D* with a shift in register and harmony. More importantly, these elements or their variants can be found in the original score of Bach.

Figure 4(b) shows the pitch-harmony space of both the original MIDI and the generated score. As is seen, due to the collaborative and competitive multiagent architecture of the system, there are new combinations of attributes which do not exist in the trained score.

9 Conclusion

We proposed an anticipatory framework for machine improvisation and style imitation of music. Our anticipatory framework is a result of fundamentals of music cognition and incorporates multiple collaborative and competitive agents each handling part of the musical attribute and through memory-based learning. This anticipatory scheme serves to *motivate*, *prepare* and *represent* music data in interaction with its environment. Besides the results demonstrated earlier, the proposed architecture addresses two main challenges of music information processing systems reviewed earlier: that of complex structure of music signals as they inherit a conjunction of different features and that of formal planning through interactive learning with an environment. In contrary to similar systems, the proposed architecture achieves certain formal planning and with only small data available through unsupervised learning and no a priori information of formal structure for any style of music. We demonstrated results using 6 different musical attributes but it should be noted that the architecture is independent of the number and nature of attributes used.

Despite its achievement, this anticipatory scheme is rather simple and does not address the wide role of musical expectation in listening and generating music. Future work should consider more elaborate models of musical memory or representations than the one presented. The musical representation used here, despite its linear and expressive formalisms, does not reveal all the structures necessary that occur during early processing in our auditory systems. Also, in a music generation system, it is desirable to combine local rule-learning schemes with long-term formal planning described before. More anticipatory schemes such as *sensory anticipation* should also be studied to achieve more interesting interactive music systems.

References

1. Leonard B. Meyer. *Emotion and Meaning in Music*. University of Chicago Press, 1956.
2. Robert Rosen. *Anticipatory Systems*, volume 1 of *IFSR International Series on Systems Science and Engineering*. Pergamon Press, Oxford, 1985. Editor-in-Chief: George J. Klir.
3. Lejaren A. Hiller and L. M. Isaacson. *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill Book Company, New York, 1959.
4. I. Xenakis. *Formalized Music*. University of Indiana Press, 1971.
5. Darrell Conklin. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35, 2003.
6. Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.
7. M. Feder, N. Merhav, and M. Gutman. Universal prediction of individual sequences. *IEEE Trans. Inform. Theory*, 38(4):1258–1270, Jul 1992.
8. Shlomo Dubnov, R. El-Yaniv, and Gérard Assayag. Universal classification applied to musical sequences. In *Proceedings of International Computer Music Conference*, pages 322–340, Michigan, 1998.

9. Shlomo Dubnov, Gerard Assayag, Olivier Lartillot, and Gil Bejerano. Using machine-learning methods for musical style modeling. *IEEE Computer Society*, 36(10):73–80, October 2003.
10. François Pachet. The continuator: Musical interaction with style. In *Proc. of International Computer Music Conference*, Gotheborg, Sweden, September 2002. ICMA.
11. Dana Ron, Yoram Singer, and Naftali Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.
12. Gérard Assayag and Shlomo Dubnov. Using factor oracles for machine improvisation. *Soft Computing*, 8-9:604–610, 2004.
13. Cyril Allauzen, Maxime Crochemore, and Mathieu Raffinot. Factor oracle: A new structure for pattern matching. In *Proc. of Conference on Current Trends in Theory and Practice of Informatics*, pages 295–310, London, 1999. Springer-Verlag.
14. John A. Biles. Genjam in perspective: A tentative taxonomy for genetic algorithm music and art systems. In Colin G. Johnson and Juan Jesus Romero Cardalda, editors, *Genetic Algorithms in Visual Art and Music*, pages 133–135, Las Vegas, Nevada, USA, 8 July 2000.
15. Judy A. Franklin. Predicting reinforcement of pitch sequences via lstm and td. In *Proc. of International Computer Music Conference*, Miami, Florida., September 2004. ICMA.
16. Darrell Conklin and I. Witten. Multiple viewpoint systems for music prediction. In *Journal of New Music Research*, volume 24, pages 51–73, 1995.
17. David Huron. *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press, 2006.
18. G. Edelman. *Neural Darwinism: The Theory of Neuronal Group Selection*. Basic Books, 1987.
19. Bob Snyder. *Music and Memory: An Introduction*. MIT Press, New York, 2000.
20. Martin Butz, Olivier Sigaud, and Pierre Gérard, editors. *Anticipatory Behavior in Adaptive Learning Systems, Foundations, Theories, and Systems*, volume 2684 of *Lecture Notes in Computer Science*. Springer, 2003.
21. Martin Butz, Olivier Sigaud, and Pierre Gérard. Internal models and anticipations in adaptive learning systems. In *Anticipatory Behavior in Adaptive Learning Systems*, pages 86–109, 2003.
22. Paul Davidsson. A framework for preventive state anticipation. In *Anticipatory Behavior in Adaptive Learning Systems*, pages 151–166, 2003.
23. Shlomo Dubnov. Spectral anticipations. *Computer Music Journal*, 2006.
24. A. Lefebvre and T. Lecroq. Computing repeated factors with a factor oracle. In L. Brankovic and J. Ryan, editors, *Proceedings of the11th Australasian Workshop On Combinatorial Algorithms*, pages 145–158, Hunter Valley, Australia, 2000.
25. Richard S. Sutton. DYNA, an Integrated Architecture for Learning, Planning and Reacting. In *Working Notes of the AAAI Spring Symposium on Integrated Intelligent Architectures*, 1991.
26. E. Uchibe and K. Doya. Competitive-cooperative-concurrent reinforcement learning with importance sampling. In *Proc. of International Conference on Simulation of Adaptive Behavior: From Animals and Animats*, pages 287–296, 2004.
27. Andrew Moore and Chris Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.