EMERGENT CONSTRUCTION OF MELODIC PITCH AND HIERARCHY THROUGH AGENTS COMMUNICATING EMOTION WITHOUT MELODIC INTELLIGENCE

Alexis Kirke
Interdisciplinary Centre for Computer Music Research,
University of Plymouth, UK

Eduardo R. Miranda
Interdisciplinary Centre for Computer Music Research,
University of Plymouth, UK

ABSTRACT

A multi-agent system is presented which generates melody pitch sequences with a hierarchical structure. The system has no explicit melodic intelligence and generates the pitches as a result of emotional influence and communication between agents, and the hierarchical structure is a result of the emerging agent social structure. Another key element is that the system is not a mapping from multi-agent interaction onto musical features, but actually utilizes music for the agents to communicate emotions. Each agent in the society learns its own growing tune during the interaction process.

1. INTRODUCTION

The generation of novelty is at the heart of many computer-aided composition (CAC) systems. Without some way of generating new material, a CAC will churn out the same material time after time. To avoid this, many systems utilize random numbers. A more recent alternative is the use of complexity which is ordered but unpredictable. Popular types of systems that generate such complexity are found in the field of Artificial Life or A-Life [1]. A-Life investigates systems related to life, their processes, and evolution; it does this most often through computer simulations and models – for example Cellular Automata. Many A-life systems have two elements in common with have made them attractive to composers for use in CAC: they generate complexity with order and structure, and they inspire composers by their variety of patterns. So although A-Life systems can generate unexpected complexity, there is an inherent order – they are not solely random. This is often called "Emergent" behaviour.

One field which has a large intersection with Artificial Life is Multi-agent Systems (MAS), which (along with computer-aided composition and computer expressive performance) is one of the 3 key areas utilized in this paper. Each agent in an MAS is a digital entity which can interact with other agents to solve problems as a group, though not necessarily in an explicitly co-ordinated way. What often separates agent-based approaches from normal object-oriented or modular systems is their emergent behaviour [1]. The solution of the problem tackled by the agents is often generated in an unexpected way due to their complex interactional dynamics, though individual agents may not be that complex. As with the application of other A Life systems in CAC, these social dynamics can be both artistically functional – for example each agent in an ensemble can contribute a motif or play an artificial instrument in a piece of music; or artistically motivational, inspiring an algorithmic composer to produce the "music of artificial societies".
In this paper we present a new MAS approach for melody generation which, unlike much A-Life CAC work, is not a mapping from multi-agent interaction onto musical features, but actually utilizes music for the agents to communicate emotions. Each agent in the society learns its own growing monophonic MIDI tune during the interaction process.

2. MULTI-AGENT SYSTEMS FOR MUSIC

Table 1 shows the majority of multi-agent systems for creating music from past research. It is designed to give quick familiarity with a number of key issues found in musical multi-agent systems. The fields are explained below:

• "Complexity" – What is the level of processing in individual agents, how complex are they?

• "Homog / Het" – are the agents in the MAS homogeneous or heterogeneous (i.e. do agents all start out the same, or are some different)?

• "Comm" – do the agents communicate, and if so do they do it synchronously or asynchronously (i.e. do they take it in turns to communicate and process, or do they do it concurrently)?

• "Initial Hierarchy" – is there a hierarchy of planning/control for the agents; are some agents dependent on others? Can some agents control others?

• "Tune" – Does the system generate multiple composition alternatives when it completes processing, or a single composition?

• "Real-time" – when the agents are activated, is the music generated in real-time?

• "Size" – what is the number, or average number, of agents in the system?

System	Complexity	Homog / Het	Comm	Tune	Initial Hierarchy	Real time	Size	Model / Func
Swarm Music	Low	Het	No	1	Flat	Y	21	F
Ant Colony Music	Low	Homog	No	1	Flat	Y		F
Swarm Orchestra	Low	Homog	No	1	Flat	Y		F
Society of Music Agents	Low	Homog	Sync	1	Flat	N		F
MMAS	Higher	Het	ASync	1	Flat	Y	8	F
Musical Agents	Higher	Het	Async	1	Flat	Y		F
Andante	Higher	Het	Async	1	Flat	Y		F
VirtuaLatin	Higher	Het	Sync	1	Hierarchy	N	1	F
MAMA	Higher	Het	ASync	1	Hierarchy	Y		F
Kinetic Engine	Higher	Het	ASync	1	Hierarchy	Y		F
CinBalada	Higher	Het	ASync	1	Flat	N		F
AALIVE	Higher	Het	ASync	1	Hierarchy	Y		F
NetNeg	Higher	Het	ASync	1	Hierarchy	N	3	F
Inmamusys	Higher	Het	Sync	1	Hierarchy	N	9	F
CAC Multi-Agent	Medium	Het	ASync	1	Flat	N	6	F
Critic Culture	Medium	Hom	Sync	Multi	Flat	N		M
Miranda Culture	Medium	Hom	Sync	Multi	Flat	N	5	M
Artificial Musical Society	Medium	Het	Sync	1	Flat	Y		F
System in this Paper	Medium	Hom	ASync	Multi	Flat	N	10	F

Table 1. Musical Multi-Agent Systems

• "Model / Func" – is the system designed solely to model some element of music, or as a computer-aided composition system?

The above choice of properties is also because they are also some of the key defining features of any MAS. The system in this paper is a non-realtime system which works with a small to medium number of agents (i.e. not hundreds of agents), it generates multiple tunes in parallel, and it is focused on computer-aided composition not on modelling the composition process or musical culture.

4. RELATED WORK

The four closest systems to the one in this paper are now examined a little more closely. The Dahlstedt and McBurney system [1] seems to be more a proposal that has never been implemented. The approach is to use agents which have different explicit goals that represent different parts of the process of music composition. An example is given of an agent whose goal is to reduce sound object density if the population of the system's "sound landscape" becomes too "cluttered"; another is given of an agent who does the opposite. Both agents would take into account the musical context while doing this. The researchers explicitly intend to utilise emergence to generate interesting music. This is a similarity with the system in this paper, though key differences are: the agents here act on a single music composition together, whereas agents in this paper each have their own repertoires which can develop in parallel, and do not have explicit and distinct goals.

Miranda's system [2] generates musical motifs in a way designed to study the evolution of culture. In this case the agents use a two-way imitation procedure to bond socially. Agents can store a repertoire of tunes and have a basic biological model of an adaptive voice box and auditory system. Agents pick other agents to interact with randomly.

When two agents A and B interact the following process occurs: if agent A has tunes in its repertoire it picks one randomly and sings it, if not then it sings a random tune. These tunes are three notes long and do not grow in length. Agent B compares the tune from A to its own repertoire and if it finds one similar enough, plays it back to agent B as an attempted imitation. Then agent B makes a judgement about how good the imitation is. If it is satisfied with the imitation it makes a "re-assuring" noise back to agent A, otherwise it does not. Based on the success of the imitation Agents A and B update their repertoires and their voice box settings to try and improve their chances of socially bonding in later interactions – e.g. by deleting or re-enforcing tunes, or making random deviations to their voice box parameters. The aim of the system is to see how the repertoire is generated and affected under such social pressures. As a result of the social bonding interactions a community repertoire begins to emerge.

Gong et al [3] produced a simple music composing system with a similar purpose to Miranda [2] - investigating the emergence of musical culture. The agents start with a set of random motifs, together with different agents being equipped with distinct but very simple aesthetic evaluation functions (for rhythm, pitch, etc.). An agent plays its tune to another agent and if the second agent finds the tune unpleasant, it modifies it (based on its musical evaluation), and plays it back to the first agent. If the first agent thinks the modified tune is better than its original, it deletes its original and stores the modified version. As agents interact this leads to "more pleasant" motifs emerging. Also, using an interaction-history measure, the social link between first and second agent is strengthened so that they are more likely to interact in the future. However if the first agent does not prefer the modified

tune to its own version, it discards it and the link between the two agents is not strengthened. It was found that in the emergent social network the agents tended to cluster according to their aesthetic preference function. This system has a couple of similarities to the one in this paper: it utilizes MAS social network/trust techniques to decide who interacts with whom, and in each interaction agents vary their repertoire based on their opinion of the other agent's repertoire. The key differences between this system and the one in this paper is that agents in this paper have no explicit evaluative melodic intelligence, and they can extend the number of notes in their repertoire; and finally the social network in this paper is used to generate music structure within an agent's repertoire not to experiment with the clustering of agents according to their repertoires.

The A-Rhythm [4] system sets out to examine the application of multi-agent systems to algorithmic composition, but has not yet fulfilled that goal. Current papers focus on, like Miranda and Gong et al., investigating the emergence of social clusters, and are solely based on rhythmic repertoire. A-Rhythm has some similarities to the system in this paper: the agents communicate and process one at a time serially (rather than in parallel) and their musical content grows longer. However A-Rhythm focuses on rhythm, i.e. is non-pitched. Also the similarity measures are more directly based on the music, rather than affective content of the music. Finally A-Rhythm uses measures for the "popularity" of rhythms in an agent's repertoire, but not for the "popularity" of agents. Agents in the system can transform their repertoires based on interaction – using certain rhythmic transformation rules (rather than the affective-based transformations used in this paper). A number of experiments are done based on different interaction approaches, and the resulting population and repertoire dynamics are examined, showing the potential for the emergence of structured rhythmic repertoires.

5. SYSTEM OVERVIEW

Agents in this system are initialized with a tune contain a single note, and over the interaction period each agent builds longer tunes through interaction. Figure 1 shows a static representation of a collection of agents.

A summary of the system is given below based on some of its key features:

1. Size – usually consists of a small-medium size (2 to 16) collection of agents, but can be more.

2. Music - Each agent can perform monophonic MIDI tunes and learn monophonic tunes from other agents.

3. Affective Performance - An agent has an affective state (an "artificial emotional state") which affects how it performs the music to other agents; e.g. a "happy" agent will perform their music more "happily.

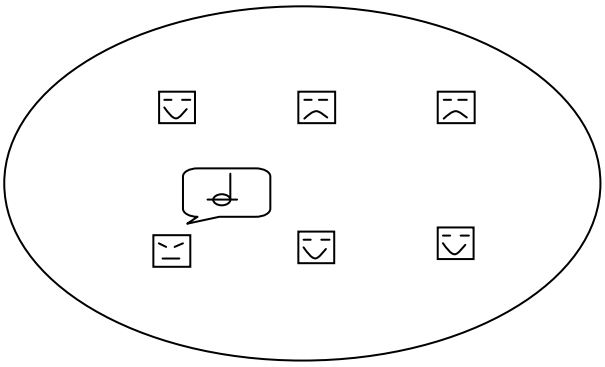

Figure 1. Six agents in a variety of affective states with one agent performing.

4. Affective Influence - An agent's affective state is in turn affected by the affective content of the music performed to it; e.g. if "sad" music is performed to a "happy" agent, the agent will become a little more "sad".

5. Tune Learning - Agents will only learn tunes performed to them if the affective content of the tune is similar enough to their current affective state; learned tunes are added to the end of their current tune.

6. Agent Interaction Coefficient - Agents develop "opinions" of other agents that perform to them, depending on how much the other agents can help their tunes grow. These opinions affect who they interact with in the future.

Each agent contains three data structures. The first is an Agent Tune, a monophonic tune in MIDI format. The second is an Agent Affective State – a number pair [valence, arousal] representing the artificial affective state of the agent based on the valence/arousal model of affectivity. The most common dimensional affective representation in computer music is the valence / arousal 2D emotional representation [5]. In this model, the first dimension being how positive or negative the emotion is (Valence), and the second dimension being how strong the emotion is (Arousal). "Valence" refers to the positivity or negativity of an emotion – e.g. a high valence emotion is joy or contentment, a low valence one is sadness or anger. Arousal refers to the arousal level of the emotion – for example joy has a higher arousal than happiness (though both have high valence), and anger a higher arousal than sadness (though both have low valence).

The final agent data structure is an Interaction Coefficient List, which is a list of interaction coefficients of all the other agents in the collection –

these are non-negative floating point numbers which measure how "popular" the agent finds each of the other agents. The concept of Interaction Coefficient is used here to attempt to create emergent compositional hierarchies (as will be demonstrated). However another way of thinking of Interaction Coefficient at this point is to consider an imagined "motivation" for an agent. The aim of this MAS is – starting with each agent having a single note - to build actual melodies. So an agent should "want" notes. An agent A's Interaction Coefficient measure of another, say Agent B, is based on the note count and number of performances it has added from B to its own tune.

An agent also has a number of internal processing functions. The Performance Output Choice involves choosing which agent to perform to, based on its Interaction Coefficient list of agents. It will only perform to agents it finds "popular enough". The Performance Output Transform involves the agent playing its single stored tune as a performance to another agent, with musical performance features based on its own current affective state. The Performance Input Estimate allows the agent to estimate the affective content of a tune performed to it by another agent, and adjust its own internal affective state based on the affective content. An agent's Performance Input Choice involves it deciding whether to store a performance from another agent, and is based on: (a) the affective content of that performance, and (b) how long the listening agent's current tune is (an agent have a finite tune length memory which can fill up). The Performance Input Interaction Coefficient system lets the agent update its Interaction Coefficient measure of another agent based on that agent's performance. Finally the Performance Input Add function lets the agent store a performance by concatenating it to the end of its current tune. An example interaction cycle is shown below. This cycle is repeated until the desired compositional result is reached. It starts by selecting the next performing agent, say Agent A:

1. If Agent A's Interaction Coefficient measure for Agent B is below Agent A's average Interaction Coefficient for other agents, then ignore Agent B and select the next listener agent, repeating this test.
2. Agent A performs its tune T_A, adjusting the tune based on its own current affective state to give performance P_A.
3. Agent B estimates the affective content of Agent A's performance P_A.
4. If B's estimated affective content of P_A is close to its own current affective state, Agent B concatenates P_A to the end of its own tune TB. Or to put it another way: $T_B = T_B + P_A$.
5. Agent B adjusts its own affective state towards its estimate of the affective content of performance P_A.
6. Agent B updates its Interaction Coefficient measure of Agent A proportional to the number of notes provided by Agent A in performance P_A.
7. Agent A turns its attention iteratively to the next agent, and returns to Step 1. Note: once all agents have been considered as candidates for performance by Agent A, a new performer agent is iteratively selected to perform , say agent B, and to listen, say agent C.

6. MUSIC TRANSFORMS

Before performing its tune to another agent, an agent will transform its tune in a "compositional" way, and then transform it using expressive performance transformations. The compositional trasnsforms are defined as transformations on the agent's MIDI tune that are not of the size and type found in Computer Systems for Expressive Performance (CSEMPS) [5]. CSEMPs create patterns in music involve micro-changes in timing, loudness and pitch. In systems such as the Computational Music Emotion Rule System [6] which perform both compositional and expressive performance transformations, the compositional transformations involve larger changes in pitch (i.e. a semitone or more) and timing. The compositional transformations used in the system in this paper are based on two sources: CMERS and Juslin's [7] paper on musical features and emotional expression. Two types of compositional transformations are applied - Linear Feature Transforms and a Key Mode Transform. The linear transformations - from [6] and [7] act on: (a) timing onset and (b) duration are inversely proportional to arousal, (c) loudness is proportional to valence and arousal, and (d) pitch is proportional to valence and arousal (but with valence having twice as much influence). These are implemented using linear equations described in equations (1) to (4).

$$IOI_i(A)' = IOI_i(A)(1 - \theta_{IOI}\, arousal_A) \quad (1)$$

$$dur_i(A)' = dur_i(A)(1 - \theta_{IOI}\, arousal_A) \quad (2)$$

$$loud_i(A)' = loud_i(A)\left(1 + \frac{\theta_{loud}}{2}(valence_A + arousal_A)\right) \quad (3)$$

$$pitch_i(A)' = pitch_i(A)\left(1 + \frac{\theta_{pitch}}{2}(2valence_A + arousal_A)\right) \quad (4)$$

When an agent A is about to perform and has a particular level of valence ($valence_A$) and arousal ($arousal_A$), it will first compositionally transform its stored tune based on the effects of equations (1) to (4).

The primed values on the left hand side of the equations are the defining features of the compositionally transformed music, and are used to unambiguously generate a transformed MIDI file. The pre-transformation values $IOI_i(A)$, $dur_i(A)$, $loud_i(A)$, and $pitch_i(A)$ are: the inter-onset interval between note i and the next note i+1, the note duration in seconds, the MIDI loudness, and MIDI pitch of the i-th musical note of Agent A's stored tune. The theta values – (θ_{ioi}, θ_{loud}, and θ_{pitch}) – define the affective sensitivity of the transformation – i.e. how much affect a change in Agent A's valence or arousal will have on the transformation.

They are the maximum variation percentage bars around the current feature value. For example if theta is 0.25, then by Equation (1) the onset will vary from 25% below its current value to 25% above its current value when arousal varies from -1 to 1. If a transformation goes above the maximum MIDI value (127) then it is set to 127. Similarly if it goes below 1 it is set to 1. Note θ_{ioi} is used both for onsets and duration so that as gaps between notes are increased or decreased, the duration of the same notes is increased and decreased by the same amount.

For positive emotion a major key is utilized and for negative emotion with negative arousal (e.g. "sadness") a minor key is utilized. For negative valence and positive arousal (e.g. "Anger" or "Fear") each note is transformed to C minor then moved alternately up or down a semitone; this is designed to inject an "atonal" element to the music. The transform is algorithmic and deterministic – it searches either side of the current notes for a note in the new mode which does not violate a MIDI boundary (i.e. not out of the MIDI 128 parameter range). So suppose an agent A has stored a tune from a "happy" agent which is a major key. If agent A then performs its tune while "sad" it will convert all of its tune, include the major part it received from another agent, into the minor mode. The current version in this paper has no ability for actual key composition functionality, hence the reason for using only C major and C minor.

The Expressive Performance transformations are not detailed in this paper. These are the micro transformations which one often hears a human making a fixed score – speeding up and slowing down, and getting louder or quieter, without any explicit structure in the score. There is not space in this paper to detail all elements of the expressive performance algorithms used, and their effectiveness is not fully tested at this stage. The method used is based on Director Musices rules for affective expressive performance [8]. Specifically the rules Phrase Arch, High Loud, Duration Contrast, Punctuation, and Duration Contrast Articulation. These are also similar to the rules used in [6] to augment the compositional emotional transforms. In this paper a similar method is also used to implement the combining of the compositional and performative elements. The key difference to [6] is that Livingstone requires the transformed tunes to first have a structural analysis. Whereas the system in this paper utilizes the key advantage of applying expressive performance during composition: that a structural analysis is no longer needed [5].

7. TUNE AFFECTIVE ESTIMATION

A linear equation is used to model the listening agent's (say agent B) affective estimate of a performance by agent A – this is shown in equations (5) and (6).

$$valenceEst_B = x_p mean(pitch_A) + x_l mean(loud_A) + x_k mean(keyMode_A) + x_{IOI} mean(IOI_A) + x_0 \quad (5)$$

$$arousalEst_B = y_p mean(pitch_A) + y_l mean(loud_A) + y_{IOI} mean(IOI_A) + y_0 \quad (6)$$

In these equations $pitch_A$ and $loud_A$ refer to the average MIDI pitch and MIDI loudness of an agent A's performance (heard by B). $keyIndex_A$ is defined as having value 2 for a minor key, and 1 for a major key; and the key mode of A's tune is estimated using a key profile-based algorithm [9]. The x and y coefficients in the Equations are constants estimated by linear regression.

These are estimated in a one-off process as follows. A set of 1920 random MIDI files was generated, of random lengths between 1 and 128 notes. Each MIDI file was transformed for 10 known and equally spaced valence and arousal values between -1 and 1 using transformation equations (1) to (4), and key mode transformations. Then a linear regression was run on the resulting transformed MIDI files against the known arousal and valence values – based on equations (5) and (6). The resulting coefficients were tested and the average percentage errors – when tested on a separate 1920 transformed random files - were 10% for valence and 9% for arousal. These are considered to be sufficiently accurate given that actual human musical emotion recognition error rates can be as high as 23% [5].

It will be noted that only the compositional, and not the expressive performance, transformations are used in the regression above. This was because it was desired to keep the model flexible for use with and without expressive performance, since a composer may wish to compose a tune without expressive performance. It was found that using the Linear Estimator with expressive performance transformations overlaid on agent B's performance did not cause excessive errors in affective estimate by agent A.

The Linear Estimator is used in two aspects of the agents – firstly for an agent to decide whether or not to add a performance to its own tune, and secondly for an agent to be influenced by the affective content of a performance it has heard. Equations (7) and (8) below are used to update the valence and arousal of agent B

after a performance from agent A. The γ (Gamma) constant - between 0 and 1 - defines how sensitive an agent is to affective state change – i.e. the amount of change to valence and arousal. If it is set to 1 then the new valence and arousal values will be totally changed to the estimated values of the performance the agent has just heard. A value of 0 will lead to no change. Values between 0 and 1 will cause the estimate to have a proportionally greater effect.

$$valence'_B = (1 - \gamma_v)valence_B + \gamma_v valenceEst_A \quad (7)$$

$$arousal'_B = (1 - \gamma_a)arousal_B + \gamma_a arousalEst_A \quad (8)$$

Once the agent B has decided whether or not to append the performance from A (and if so, has done so), it will update its valence and arousal based on Equations (7) and (8). In future, when it next performs a tune, it will transform it based on its new valence and arousal state. It is designed so that through this series of updating affective states and the agent tune communication and system, new musical structures will emerge.

8. INTERACTION COEFFICIENT

Before an Agent A performs to an Agent B it compares its Interaction Coefficient measure of Agent B to the average of its Interaction Coefficient for other agents:

$$IC(A,B) > mean[IC(A, all\ agents)] \quad (9)$$

If it is not, then it does not perform to Agent B and moves on to the next agent. The increase in Interaction Coefficient is proportional to the length of tune it has added. So the more notes in Agent A's performance, the greater its Interaction Coefficient will be viewed by Agent B. The parameter *d* is a constant called the Interaction Coefficient Update Rate.

$$IC(B,A) = IC(B,A) + d.N \quad (10)$$

This can be visualised as an Agent's basic resources being tunes - so the more notes in an Agent's tune, the greater its potential Interaction Coefficient to other agents. However the actual reason for including Interaction Coefficient functionality, and making Interaction Coefficient proportional to the number of notes in a performing agent's tune is primarily to generate a "social" hierarchy amongst the agents which influences the hierarchy of the composed music. Bearing in mind that an agent will only perform to other agents with a high enough Interaction Coefficient, it can be seen that:

- Agents which perform more than listen will tend to have lower interaction coefficients
- Agents which mostly listen and store will have longer tunes and higher interaction coefficients
- Agents with higher interaction coefficients will tend to be selected as listeners more often

So the system is designed to turn the agent population into a set of agents who tend to perform and have shorter tunes, and a set of agents who tend to listen and store. The aim is for lower Interaction Coefficient agents to be focused on providing lower elements of the musical hierarchy.

9. AN EXAMPLE CYCLE

An example cycle will now be shown. In this example we examine three agents: (a) Agent 1 is the performer and starts by considering performing to Agent 2; (b) Agent 1's measure of Agent 2's Interaction Coefficient is very low in this example; (c) Agent 1's measure of Agent 3's Interaction Coefficient is very high; (d) Agent 1's affective state is high valence and high arousal – i.e. "happy"; Agent 3's affective state is low valence and low arousal – i.e. "sad".

1. Because Agent 1's Interaction Coefficient of Agent 2 is very low, Agent 1 does not even perform to Agent 2. It selects the next Agent iteratively – Agent 3.
2. Agent 1's view of Agent 3's Interaction Coefficient is very high – so Agent 1 performs its tune T1, adjusting it to make it "Happier" because of its high valence and arousal state, giving a performance P1.
3. Agent 3 estimates the affective content of Agent 1's performance P1 and gets a result of high valence and arousal – i.e. it estimates it is a "happy" performance.
4. Because Agent 3's affective estimate of Agent 1's tune is high valence and arousal but Agent 3's state is low valence and arousal – i.e. very different to "happy" - Agent 3 discards Agent 1's tune.
5. However Agent 3 still adjusts it owns affective state towards its estimate of the affective content of performance P1 i.e. it becomes a little more "happy".
6. Neither Agent makes any adjustment to their Interaction Coefficient measures since no performances were stored.
7. Agent 1 remains the performer, and the next agent is iteratively chosen to listen – i.e. Agent 4.

10. EXAMPLE RESULTS

(Note: In this system there are a number of parameters which need to be set; it is beyond the scope of this paper to describe all of them. Those not mentioned explicitly here were set to default values.) The best way to indicate that this system can produce non-trivial melodies, in spite of its lack of melodic intelligence is to explore the space of what sort of pitches can be produced by the system. This is partially done in Figures 2 to 4. This is 8 agents run for 10 cycles. In each cycle an agent performs to all other 7 agents, depending on their interaction coefficient. Then in the next cycle the second agent is selected to perform, and so on moving through the agents. At the start agents were initialized with certain affective states and these were allowed to evolve through the interaction cycles. Four initializing states were used with different levels for [Valence, Arousal] pairs. These were: "Happy" = [0.5, 0.5]; "Sad" = [-0.5, -0.5]; "Angry" = [-0.5, 0.5]; Tender = [0.5; -0.5].

The shown tunes and a number of others were played to 10 listeners in tests. It was found that when at least 6 out of the 8 agents had the same initial valence and arousal, then the listeners had a 71% chance of detecting that same valence in the final tune, and an 82% chance of detecting that same arousal in the tune.

11. HIERARCHICAL STRUCTURE RESULTS

To display the effects of Interaction Coefficient an 8 agent system was used with equally spread agent initial affective states – i.e. 2 sad, 2 happy, 2 angry, 2 tender, and run for 32 cycles. The Interaction Coefficient update rate was set to 0.2; this is the rate at which an agent's Interaction Coefficient Measure of other agents is updated. At the end of the runs the number of notes the agents 1 to 8 have is respectively: 291, 102, 102, 102, 102, 102, 18, and 5. It was found that this relates to the Interaction Coefficient – the higher an agent's final Interaction Coefficient the higher its note count. Agent 1 ended up with the highest, followed by Agents 2 to 6 and agents 7 and 8 had the lowest. The pattern of interaction can be seen from another perspective in Table 2. It can be seen that smaller Interaction Coefficient agents tend to give out tunes, while the larger Interaction Coefficient agents tend to receive tunes. The lower numbered agents have higher Interaction Coefficient because of the ordering of agent interaction in each cycle. The lower agents will be performers first, and have a chance to build up their IC. Then when they become listeners (receivers) these lower numbered agents will receive larger tunes back, and their Interaction Coefficient will increase as a result. To see how this creates the hierarchical structure, consider that by

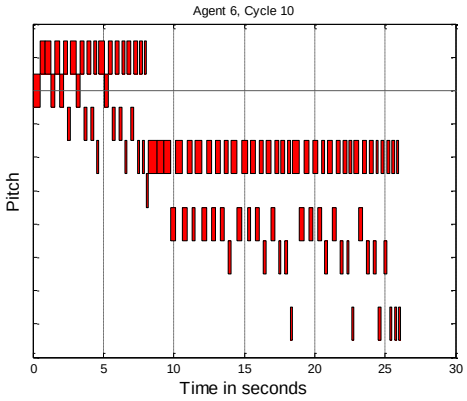

Figure 2. 6 Angry, 2 Sad initialized

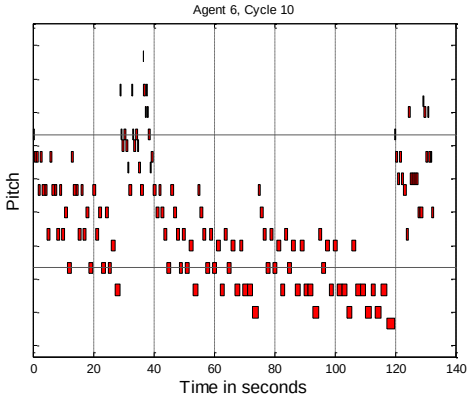

Figure 3. 6 Sad, 2 Happy initialized

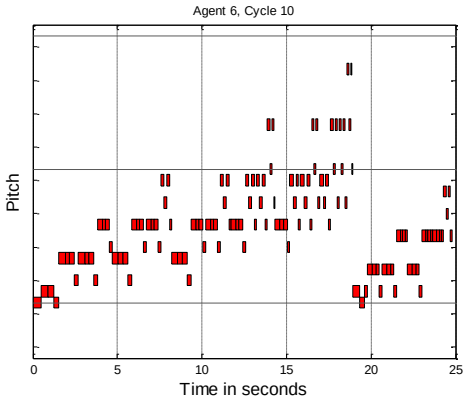

Figure 4. 6 Happy, 2 Angry initialized

Table 2 Agent 1's final tune could be written as a concatenation of subtunes:

$$1_0 2_1 3_2 4_3 5_4 6_5 7_6 2_9 3_{10} 4_{11} 5_{12} 6_{13} 2_{17} 3_{18}$$

where each number indicates the agent who performed, and the subscripts are the cycle numbers (an agent's tune varies over different cycles – e.g. 3_{18} is not the same 3_2.)

Because the MAS is a closed system, all tunes in this structure are the result of a transformation on another agent's tune. So for example:

Cycles / Agent	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	24	32
1		2	3	4	5	6				2	3	4	5	6				2	3		
2	1						7	8									1				
3	1						7	8									1				
4	1						7	8									1				
5	1						7	8									1				
6	1						7	8									1				
7								8									8			8	8
8							7								7						

Table 2. Pattern of Interaction

$2_1 = 2_0 1_0$'
$3_2 = 3_0 1_0$''
…

$7_6 = 7_0 1_0$'''

Where the primes (') represent transformations on Agent 1's tune due to Agent 1's affective state at the time. In the next round of tunes being given to Agent 1 this gives:

$$2_9 = 2_1 7_7 '18_8 ' = (2_0 1_0 ') 7_7 '18_8 '$$

This expansion can be continued until there is a full description of Agent 1's tunes based on the way in which the tune grows. This description will show the building structure of the tune. (It will not necessarily show the *perceptual* structure of the tune – this is not claimed, but it will show how the tune was built from the motifs and phrases etc of other agents). This structure is clearly a function of the agent interaction hierarchy, and as has been seen this hierarchy is strongly influenced by the Interaction Coefficient functionality. Hence this supports the idea that the Interaction Coefficient provides a non-affective-based method for generating coherent hierarchical structure in the tunes, based on the emerging interaction structure in the multi-agent system.

12. CONCLUSIONS

A multi-agent system has been presented which generates melody pitch sequences with a hierarchical structure. The system has no explicit melodic intelligence and generates the pitches as a result of emotional influence and communication between agents, and the hierarchical structure is a result of the emerging agent social structure. Another key element was that the system is not a mapping from multi-agent interaction onto musical features, but actually utilizes music for the agents to communicate emotions. Each agent in the society learns its own growing tune during the interaction process.

13. REFERENCES

[1] Dahlstedt, P., McBurney, P., "Musical agents: Toward Computer-Aided Music Composition Using Autonomous Software Agents", *Leonardo* 39, 469-470, 2006

[2] Miranda, E.R., "Emergent Sound Repertoires in Virtual Societies", *Computer Music Journal* 26, 77-90, 2002

[3] Gong, T., Zhang, Q., Wu, H., "Music evolution in a complex system of interacting agents", *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, UK, 2005

[4] Martins, J.M., Miranda, E.R., "Emergent Rhythmic Phrases in an A-Life Environment", *Proceedings of ECAL 2007 Workshop on Music and Artificial Life (MusicAL 2007)*, Portugal, 2007

[5] Kirke, A.J., Miranda, E.R.," A Survey of Computer Systems for Expressive Performance of Music", *ACM Computing Surveys* 42, 3:1, 2009

[6] Livingstone, S., Muhlberger, R., Brown, A., Thompson, W., "Changing Musical Emotion: A Computational Rule System for Modifying Score and Performance", *Computer Music Journal* 34, p41, 2010

[7] Juslin, P. "From Mimesis to Catharsis: expression, perception and induction of emotion in music", In *Music Communication*, D. Miell, R. Macdonald, And D.J. Hargreaves, Eds. Oxford University Press, 85-116, 2005

[8] Friberg, A., Bresin, R., Sundberg, J., "Overview of the KTH rule system for musical performance", *Advances in Cognitive Psychology* 2, 145-161, 2006

[9] Krumhansl, C., Kessler, E., "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys", *Psychological Rev* 89, 334-368, 1982