Breeding Rhythms with Artificial Life

Joao M. Martins and Eduardo R. Miranda
Interdisciplinary Centre for Computer Music Research
University of Plymouth, UK
{joao.martins, eduardo.miranda}@plymouth.ac.uk

***Abstract* — We are interested in developing intelligent systems for music composition. In this paper we focus on our research into generative rhythms. We have adopted an Artificial Life (A-Life) approach to intelligent systems design in order to develop generative algorithms inspired by the notion of music as social phenomena that emerge from the overall behaviour of interacting autonomous software agents. Whereas most A-Life approaches to implementing computer music systems are chiefly based on algorithms inspired by biological evolution (for example, Genetic Algorithms [2]), this work is based on cultural development (for example, Imitation Games [12, 13]). We are developing a number of such "cultural" algorithms, one of which is introduced in this paper: the *popularity algorithm*. We also are developing a number of analysis methods to study the behaviour of the agents. In our experiments with the popularity algorithm we observed the emergence of coherent repertoires of rhythms across the agents in the society.**

I. Introduction

The A-Life approach to music is a promising new development for composers. It provides an innovative and natural means for generating musical ideas from a specifiable set of primitive components and processes reflecting the compositional process of generating a variety of ideas by brainstorming followed by selecting the most promising ones for further iterated refinement [8]. We are interested in implementing systems for composition using A-Life-based models of cultural transmission; for example, models of the development and maintenance of musical styles within particular cultural contexts, and their reorganization and adaptation in response to cultural exchange.

Existing A-Life-based systems for musical composition normally employ a Genetic Algorithm (GA) to produce musical melodies, rhythms, and so on. In these systems, music parameters are represented as "genotypes" and GA operators are applied on these representations to produce music according to given fitness criteria. Because of the highly symbolic nature of Western music notation, music parameters are suitable for GA-based processing and a number of musicians have used such systems to compose music.

Although we acknowledge that there have been a few rather successful stories [2], we believe that additional A-Life-based methods need to be developed [11, 12]. The work presented in this paper contributes to these developments by looking into the design of algorithms that consider music as a cultural phenomenon whereby social pressure plays an important role in the development of music. A plausible method to embed social dynamics in such algorithms is to design them within the framework of interacting autonomous software agents.

We are developing a multi-agent system for composition of rhythms where the user will be able to extract information about the behaviour of the agents and the evolving rhythms in many different ways, providing composers the means to explore the outcomes systematically. An in-depth discussion on the architecture of the whole system and how it will be used artistically to compose pieces of music falls beyond the scope of this paper. Rather, this paper will focus on one of the A-Life algorithms that we have developed for the system - the *popularity algorithm* - and the information that one can extract about its behaviour, and the analyses of the behaviours.

By way of related research, we cite the work by de Boer [3] on modelling the emergence of vowel systems by means of imitations games and Kirby's work on evolution of language [9]. Also, Miranda [13] has developed a model of the emergence of intonation systems using imitation games. Basically an imitation game consists of one agent picking a random sound from its repertoire and the other agent trying to imitate it. Then, a feedback is given about the success of the imitation. On the basis of this feedback, the agents update their memories.

II. The Agents

The agents are identical to each other and the number of agents in a group may vary. The agents move in a virtual 2D space and they normally interact in pairs. Essentially, the agents interact by playing rhythmic sequences to each other, with the objective of developing repertoires of rhythms collectively. At each round, each of the agents in a pair plays one of two different roles: the *player* and the *listener*. The agents may perform operations on the rhythms that they play to each other, depending on the iteration algorithm at hand and on the status of the emerging repertoire. The agents are provided with a memory to store the emerging rhythms and other associated information.

The fundamental characteristic of human beings is that we are able to perceive, and more importantly, to produce an isochronous pulse [6]. Moreover, humans show a preference for rhythms composed of integer ratios of the basic isochronous pulse [5]. Therefore, we represent rhythms as interonset intervals in terms of small integer ratios of an isochronous pulse (Fig. 1).

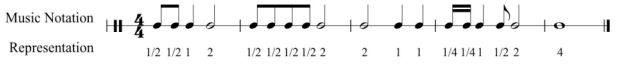

Fig. 1. Standard music notation of a rhythmic sequence and its corresponding interonset representation.

A. Transformations of Rhythms

At the core of the mechanism by which the agents develop rhythmic sequences is a set of basic transformation operations. These operations enable the agents to generate new rhythmic sequences and change the rhythmic sequences that they learn as the result of the interactions with other agents. The transformation operations are as follows:

- Divide a rhythmic figure by two (Fig. 2a)
- Merge two rhythmic figures (Fig. 3b)
- Add one element to the sequence (Fig. 2c)
- Remove one element from the sequence (Fig. 2d)

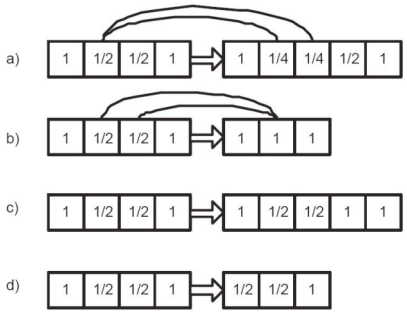

Fig. 2. Examples of rhythmic transformations.

The definition of these transformations were inspired by the dynamical systems approach to study human bimanual coordination [7] and is based on the notion that two coupled oscillators will converge to stability points at frequencies related by integer ratios [1]. We have defined other transformations that divide a figure into three, five, and other prime numbers, but the impact of these additional transformations on the system is beyond the scope of this paper. Addition and removal transformations were introduced to increase diversity in the pool of rhythms and to produce rhythms of different lengths.

B. Measuring Similarity of Rhythms

The agents are programmed with the ability to measure the degree of similarity of two rhythmic sequences. This measurement is used when they need to assess the similarity of the rhythms they play to each other. Also, this algorithm is used to measure the similarity between repertoires of rhythms from different agents.

In a previous paper [10] we introduced a method to measure the degree of similarity between two sequences of symbols by comparing various subsequences at various levels. The result is a vector, referred to as the *Similarity Coefficients Vector* (SCV), which contains the interim results of the comparisons between subsequences. For the present work, we devised a version of the SCV method to deal with rhythmic sequences.

Let us define the block distance between two sequences containing the same number of elements as follows:

$$d(\mathbf{v}, \mathbf{w}) = \sum_{i=1}^{n} |v_i - w_i|$$

where **v** and **w** are the two sequences (vectors) that are being compared, and v_i and w_i are the individual components of these vectors. After obtaining the resulting evaluation of the block distances on a given level (length of a subsequence), we can write a matrix for the *k*-level, corresponding to the comparison of all the subsequences with length *k* between the two main sequences:

$$\mathbf{D}^{(k)} = \begin{bmatrix} d(\mathbf{v}_1^{(k)}, \mathbf{w}_1^{(k)}) & \dots & d(\mathbf{v}_1^{(k)}, \mathbf{w}_{(m-k+1)}^{(k)}) \\ d(\mathbf{v}_2^{(k)}, \mathbf{w}_1^{(k)}) & \dots & d(\mathbf{v}_2^{(k)}, \mathbf{w}_{(m-k+1)}^{(k)}) \\ \vdots & \vdots & \vdots \\ d(\mathbf{v}_{(n-k+1)}^{(k)}, \mathbf{w}_1^{(k)}) & \dots & d(\mathbf{v}_{(n-k+1)}^{(k)}, \mathbf{w}_{(m-k+1)}^{(k)}) \end{bmatrix}$$

where *d* are the distances $d(\mathbf{v}, \mathbf{w})$ between all the subsequences $\mathbf{v}^{(k)}$ of **v** and all the subsequences $\mathbf{w}^{(k)}$ of **w**. Next, let us define the *k*-level *Similarity Coefficient* as follows:

$$c^{(k)}(\mathbf{v}, \mathbf{w}) = \frac{z(k)}{(n-k+1)(m-k+1)}$$

where *z(k)* is the number of zeros in the matrix $\mathbf{D}^{(k)}$. Roughly speaking, the similarity coefficient measures the sparsity of the matrix $\mathbf{D}^{(k)}$. The higher the coefficient *c(k)*, the higher is the similarity between the subsequences of level *k*. Next, we can collect all the *k*-levels coefficients in a vector referred to as *Similarity Coefficient Vector* (SCV). This is defined as follows:

$$\mathbf{C} = \left[c^{(1)}, c^{(2)}, \dots, c^{(min(m,n))} \right]$$

Fig. 3 shows an example of building a 3-level Distance Matrix and its respective SCV is *SCV* = [0.4167 0.1333 0.1250 0]. From SCV we can obtain a scalar value in order to establish a comparative analysis between larger sets of rhythms, such as the repertoires of two agents. We can take the rightmost nonzero value from the SCV, which corresponds to the higher level where two matching sequences can be found. We can either take a weighted sum of the SCV values or the average of all values, as follows:

$$SCV_{av} = \frac{1}{min(m,n)} \sum_{k=1}^{min(m,n)} SCV(k)$$

where *SCV* (*k*) are the coefficients of similarity for each of the *k* levels. The next step is to compare the repertoire of the agents in order to observe the development of

relationships amongst the agents in a group of agents; for instance, to observe if the agents form distinct sub-groupings. The similarity of the repertoire of rhythms amongst the agents in a group is computed by creating a matrix of SCV_{av} values of the repertoires of all pairs of agents. Matrices with the columns and rows corresponding to the number of rhythms in the memory of each agent reveal the similarity between their repertoires (Fig. 4).

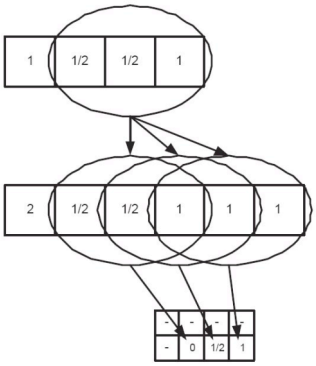

Fig. 3. Example of building a 3-level Distances Matrix.

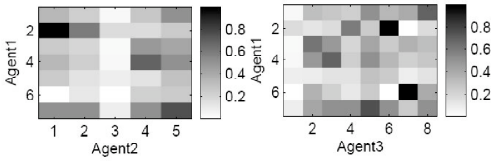

Fig. 4. Examples of similarity matrices between the repertoires of 3 agents: agent 1 vs. agent 2 and agent 1 vs. agent 3. The darker the colour the more similar the rhythms are.

By collapsing both the rows and the columns of the matrices, and taking the maximum values for each of them and an averaged sum, we obtain the scalar of similarity between repertoires, as follows:

$$SimRep_{k,l} = \frac{1}{nR_{Ak} + nR_{Al}} \left[\sum_{i=1}^{nR_{Ak}} max(SCV_{av})_{rows} + \sum_{j=1}^{nR_{Al}} max(SCV_{av})_{cols} \right]$$

where the first term corresponds to the sum of the maximum values of the SCV_{av}, for every row, and the second term is the correspondent for every column; nR_{Ak} and nR_{Al} are the number of rhythms in the repertoire of the compared agents.

Finally, the development of repertoires of rhythms of a group of agents as a whole can be observed by conducting a hierarchical cluster analysis of all distance measures between the agents (*DistRep*). This cluster analysis produces a dendrogram using a linkage method based on an unweighted average distance, also known as group average in which the distance between two clusters A and B, D_{AB}, is given by the following equation:

$$D_{AB} = \frac{1}{N_A . N_B} \sum_i d_i$$

where N_A and N_B are the number of elements in A and B, and d_i are pairwise distances between the elements of clusters A and B. The hierarchical cluster analysis produces a dendrogram of the type shown in Fig. 8. Such dendrogram is drawn through an iterative process until all the individuals or clusters are linked.

C. Measuring the Complexity of Rhythms

The complexity of a rhythmic sequence is measured as follows:

$$Complexity = \frac{nF + \sum_{i=1}^{nF} n_i}{\sum_{i=1}^{nF} T_i}$$

where nF is the number of rhythmic figures contained in the sequence, n_i is value of the numerator of a rhythmic figure, and T_i is the relative length of a rhythmic figure, considering that each rhythmic figure is a fraction of the pulse. This is a computationally cost effective method to measure the complexity of a rhythmic sequence.

It is important to bear in mind that our implementation ensures that there are no reducible fractions included in the sequence, meaning that there is always a single numerical representation for a given rhythm. Fig. 5 shows an example of a graph plotting the complexity of a sequence of relative interonset intervals [1, 1] as it is transformed thirty times recurrently, using the transformation operations mentioned earlier.

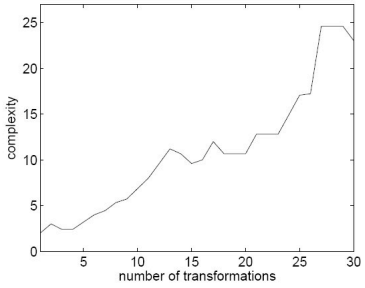

Fig. 5. Example where complexity increases with the number of transformations.

III. The Popularity Algorithm and Experiments

Popularity is a numerical parameter that each agent attributes to a rhythm in its repertoire. This parameter is modified both by the (agent-)listener and by the (agent-)player during the interactions. If the listener recognises a rhythm (that is, if it holds the "perceived" rhythm in its repertoire), then it will increase the popularity index of this rhythm and will give a positive feedback to the player. A positive feedback is an acknowledgment signal, which will prompt the player to increase the popularity index of the rhythm in question in its repertoire. Conversely, if the listener does not recognize the rhythm, then it will add it to its repertoire and will give a negative feedback to the player. This negative feedback will cause the player to decrease the popularity index of this rhythm.

Furthermore, there is a memory loss mechanism whereby after each interaction all the rhythms have their popularity index decreased by 0.05. This accounts for a natural drop in the popularity index due to ageing. The core of the popularity algorithm works as follows:

Agent Player:

P1. Plays a rhythm and increase the counter for the number of times that this rhythm has been used.

Agent Listener:

L1. Search for the heard rhythm in its repertoire

L2. If the rhythm is found, then give a positive feedback to the agent player and increase the counter for the popularity of the rhythm in its repertoire

L3. If the rhythm is not found, then add this rhythm to the repertoire and give as negative feedback to the agent player

Agent Player:

P2. Receive the feedback from agent listener

P3. If feedback is positive, then increase the counter for the popularity of the rhythm in its repertoire

P4. If feedback is negative, then decrease the counter for the popularity algorithm in its repertoire

P5. If the minimum popularity threshold for this rhythm has been reached, then remove this rhythm from its repertoire

P6. If the transformation threshold for this rhythm has been reached, then transform this rhythm

As for the analyses, firstly we analyse the development of the size and the complexity of the repertoire of individual agents. Then, we analyse the values of the corresponding individual measures from the agents, as well as similarity between agents and how they are clustered in terms of the rhythms they share. Finally, we measure the lifetime of the rhythms, the amount of rhythmic sequences that the society develops and the degree to which the agents share similar rhythms. We trace the lifetime of a rhythm by counting the number of agents that hold the sequence in their memories during the interactions. Fig. 6 shows 3 examples of analyses.

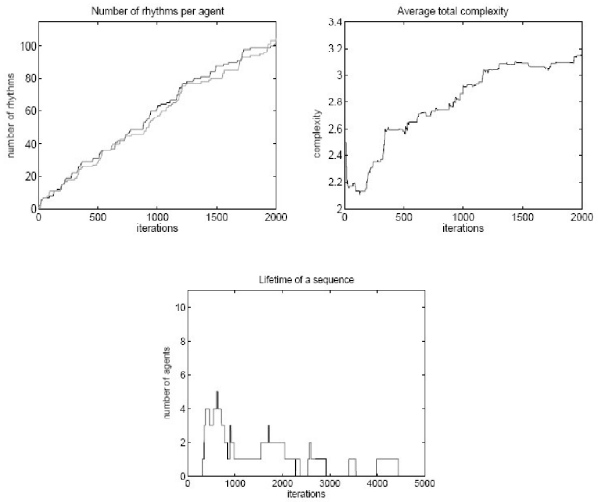

Fig. 6. Examples of amalyses: development of the size of the repertoire for different agents (top left), complexity of the rhythms of the society (top right) and number of agents sharing a particular rhythm (bottom).

The experiments were run for 5000 iterations each for a number of times, with the objective of observing the behaviour of the agents, the society and the evolving rhythms, under different conditions. We have run experiments with societies of 3, 10 and 50 agents. For some of the experiments we have limited the lifetime of the agents to 1000 iterations; when an agent dies, another is born. Sometimes the algorithms take into account the movement of the agents in the 2D space, which may or may not influence the nature of the iterations. Fig. 7 shows the results after 5000 iterations of the popularity algorithm with 10 agents (without population renewal).

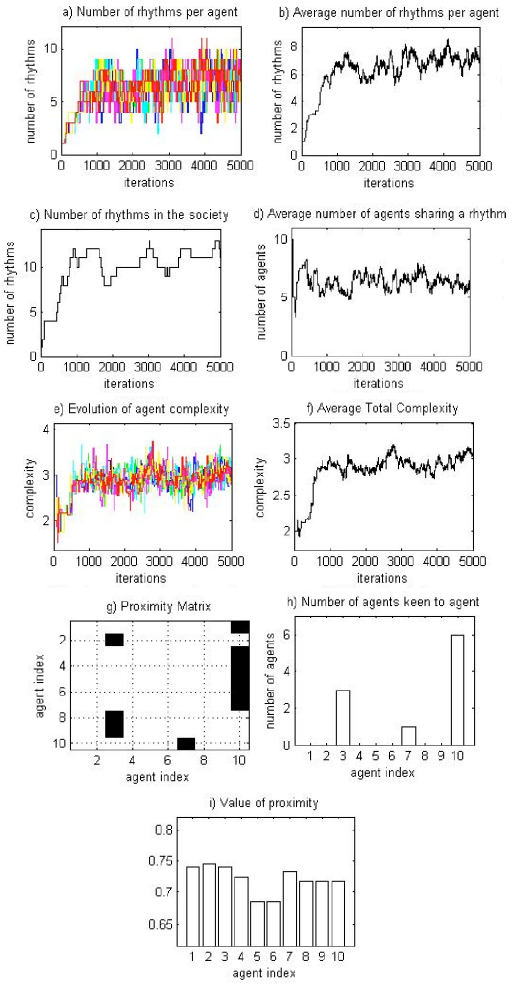

Fig. 7. Results from a typical run of the popularity algorithm with 10 agents.

Fig. 7a displays the development of the repertoire of individual agents and Fig. 7b displays the corresponding average across all agents. Here the repertoires of the agents grow steadily up to approximately 1000 iterations and subsequently oscillates around a stable point. Fig. 7c displays the development of the repertoire of the whole society being a direct consequence of the lifetime of each rhythm. The average number agents sharing a rhythm (Fig. 7d) is calculated by summing the instant number of agents sharing a rhythm for all rhythms, and dividing the result by the number of rhythms currently present in the society (Fig. 7c). This graph (Fig. 7d) provides the means to assess the global behaviour of the society; for instance, if it develops coherently in terms of popularity of existing

rhythms. Fig. 7e represents the development of complexity of the individual agents and Fig. 7f gives the corresponding average. Initially, the size and complexity of the repertoire of individual agents are very close to the average, but this trend is replaced quickly by repertoires of different sizes amongst the agents.

The last three graphs show the degree of similarity between the repertoires of the agents according to the similarity measure defined earlier. Fig. 7g displays information about the identity of the agent with whom each agent relates most; i.e., has the highest similarity value. The graph in Fig. 7h shows the agents that are regarded by others as being most similar to them. In this case, it shows that agent 3 has three agents with similar repertoires, and agent 10 is the one that concentrates the highest number of keen agents, having six agents considering its repertoire to be more similar to theirs.

Hierarchical cluster analysis is conducted in order to observe groupings of agents according to the similarity of their repertoires. Fig. 8 shows the dendrogram containing elements of three societies of 10 agents each: Society 1 comprises agents 1 to 10, Society 2 comprises agents 11 to 20 and Society 3 the remaining 21 to 30. By comparing the three societies we can observe 3 clearly independent clusters, which were developed separately in three separate runs with the same set of parameters. In addition to the previous observations, this suggests that the repertoires that emerged from the popularity algorithm display diversity, are stable in terms of size, and are coherent within their respective societies. We can also observe differences in the clusters within a given society.

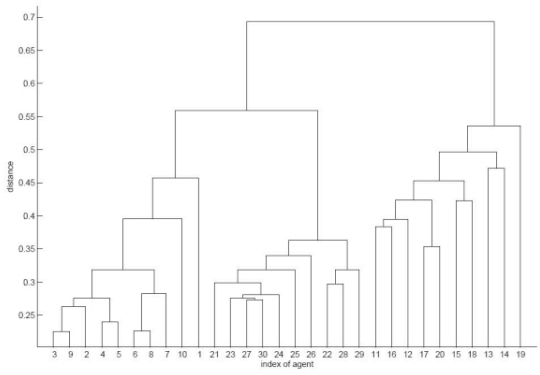

Fig. 8. Dendrogram resulting from the hierarchical cluster analysis conducted in the context of the popularity algorithm containing three independent societies with 10 agents each.

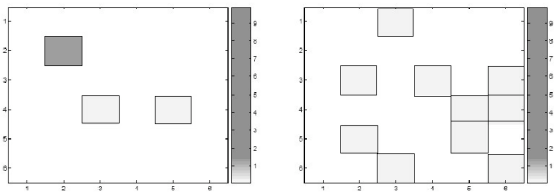

Fig. 9. World visualisation of two steps of the iterative process. Clustering takes place (figure on the left) followed by scattering at a later stage (figure on the right). A cluster is indicated by a darker colour.

By letting the agents move in their environment, we also investigated whether the interaction rules could influence the movement of the agents and whether this process would influence the development of their repertoires. In this case, if a listening agent "recognises" the rhythm played by the other agent, then it will follow the player agent in the space in the next iteration. Fig. 9 shows periodic clustering of one or more groups of agents that move together and keep interacting until the cluster is scattered due to an unsuccessful interaction.

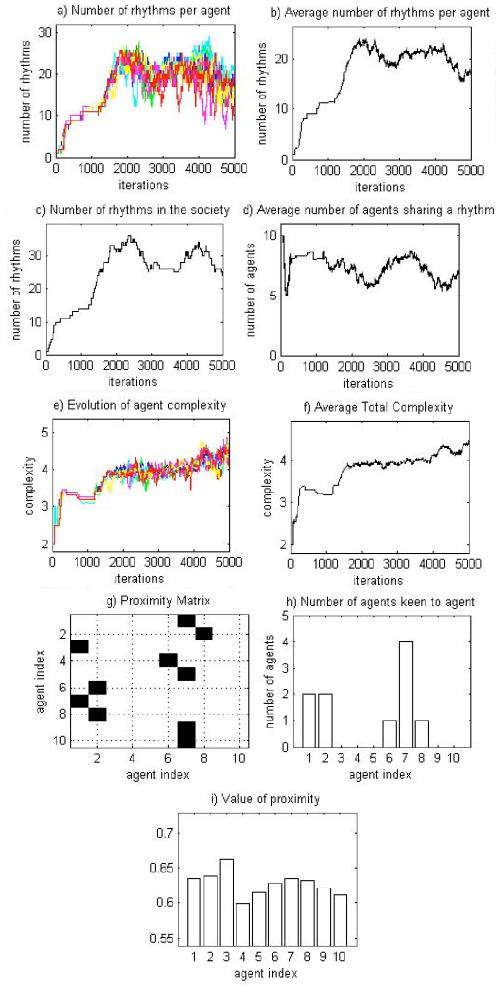

Fig. 10. Results from a typical run of the popularity algorithm taking into account the movement of the agents as an influencing factor in the evolution of the repertoire.

In Fig. 10, we can observe two behaviours that are typical of the popularity algorithm with movement taken into account. The first being that there are many more rhythms affecting the interactions than in the case without movement; this is due to the fact that every time a positive feedback occurs, two or more agents will form a group. This increases the number of interactions and consequently the number of rhythms in their repertoires. The second being that there is an initial overshoot of the size of the repertoire before reaching a level of stability. This is possibly caused by the initial clustering of agents when individual repertoires grow consistently among very closely related agents.

IV. Concluding Discusssion

We are developing novel A-Life-based generative music algorithms with a view on producing an intelligent system for the composition of rhythms. Most current approaches to using A-Life in software for generating music entail the application of a GA. We propose that a strictly GA-based approach to generate music is questionable because they were not designed to address musical problems in the first place, but to solve engineering and searching problems. The act of composing music seldom involves an automated selective procedure towards an ideal outcome based on a set of definite fitness criteria.

As a way forward, we suggest that A-Life-based systems for generating music should employ algorithms that consider music as a cultural phenomenon whereby social pressure plays an important role in the development of musical conventions. To this end, we are developing a number of algorithms, one of which was introduced in this paper: the *popularity algorithm*. In addition, we developed a number of methods to monitor the behaviour of the algorithms.

In all runs of the popularity algorithm we observed the emergence of coherent repertoires across the agents in the society. Clustering of agents according to their repertoires could also be observed on various occasions.

Whereas the size of the repertoire is controlled by a popularity parameter in the algorithm, it tends to grow constantly in the other algorithms that we have implemented. We also observed that a small subset of agents tend to concentrate the preference of most of the population. This trend tended to appear in many runs with different settings.

Acknowledgment

This project was developed thanks to a grant from The Leverhulme Trust.

References

[1] Beek, P. J., Peper, C. E., and Daffertshofer, A. (2000). "Timekeepers versus nonlinear oscillators: how the approaches differ". In P. Desain and L. Windsor, Eds., *Rhythm Perception and Production*, pp. 9-34. London: Swets and Zeitlinger (now Taylor & Francis).

[2] Biles, J. A. (1994). "Genjam: A genetic algorithm for generating jazz solos". *Proceedings of the International Computer Music Conference*, Aarhus(Denmark). San Francisco, USA: International Computer Music Association.

[3] de Boer, B. (1999). *Self-Organisation in Vowel Systems*. PhD thesis, Vrije Universiteit Brussel.

[4] Doornbusch, P. (2005). *The Music of the CSIRAC: Australia's First Computer Music*. Victoria, Australia: Common Ground.

[5] Drake, C. and Bertrand, D. (2001). "The Quest for Universals in Temporal Processing in Music". *Annals of the New York Academy of Sciences*, 930(1):17-27.

[6] Handel, S. (1989). *Listening: An Introduction to the Perception of Auditory Events*. Cambridge, USA: The MIT Press.

[7] Kelso, J. A. (1984). "Phase transitions and critical behavior in human bimanual coordination". *American Journal of Physiology – Regulatory, Integrative and Comparative Physiolpgy*, 246(6):1000-1004.

[8] Kim, K.-J. and Cho, S.-B. (2006). "A comprehensive overview of the applications of artificial life". *Artificial Life*, 12(1):153-182.

[9] Kirby, S. (2002). "Natural language from artificial life". *Artificial Life*, 8(2):185-215.

[10] Martins, J. M., Gimenes, M., Manzolli, J., and Maia Jr., A. (2005). "Similarity measures for rhythmic sequences". *Proceedings of the 10th Brazilian Symposium on Computer Music (SBCM)*, Belo Horizonte (Brazil).

[11] Miranda, E. and Biles, A., Eds. (2007). *Evolutionary Computer Music*. London, UK: Springer-Verlag.

[12] Miranda, E. R. (2004). "At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestras and the Origins of Melody", *Evolutionary Computation* 12(2):137-158.

[13] Miranda, E. R. (2002). "Mimetic development of intonation", *Proceedings of the 2nd International Conference on Music and Artificial Intelligence (ICMAI 2002)*. Springer Verlag - Lecture Notes on Artificial Intelligence.