ENGINEERING - THE ROLE OF SOCIAL PRESSURE : A NEW ARTIFICIAL LIFE APPROACH TO SOFTWARE FOR GENERATIVE MUSIC

By

JOAO M. MARTINS* EDUARDO R. MIRANDA**

ABSTRACT

Most current research into computers and music focus on the development of media technology for delivering music to consumers (e.g., MP3 format, Internet search engines, and so on). This research focuses on the development of technology for musical creativity. This paper focuses on a particular technology currently being developed, based on Artificial Life (A-Life). The Artificial Life (A-life) approach to the development of software for music is a promising new development. However, the vast majority of existing A-life-based systems for musical composition employ Genetic Algorithms (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 the given fitness criteria. It is suggested that strictly GA-based methods suffer from the fact that musical composition should not be constrained by a definite set of fitness criteria. Moreover music is largely a cultural phenomenon driven by social pressure and this is cumbersome to model with standard GA alone. An alternative approach is proposed to using strictly GA-based methods: the design of evolutionary algorithms that consider music as a cultural phenomenon whereby social pressure plays an important role in the development of musical conventions. This paper introduces three algorithms of the authors' own design: popularity, transformation and complexity algorithms, respectively. Tools are also devised for extracting information about the behaviour of the algorithms in many different ways, providing the means to study the outcomes systematically.

Keywords: Evolutionary Computer Music, Artificial Life systems for musical composition, Computer models of music.

INTRODUCTION

From the time of discovery almost three thousand years ago, the direct relationship between the pitch of a note and the length of a string or pipe, to the latest computer models of human music cognition and intelligence, musicians have always looked at science to provide new and challenging paradigms to study and create music. The field of Computer Music is as old as Computer Science. Computers have been programmed to play music as early as the early 1950's when Geoff Hill programmed the CSIR Mk1 computer, in Sydney, Australia, to play the popular musical melodies (Doornbusch, 2005). Nowadays, the computer is becoming increasingly ubiquitous in all aspects of music. Applications of computer technology to music ranges from systems for musical composition to systems for distribution of music on the Internet. The implementation of such applications often demands the skiliful combination of Software Engineering and artistic creativity. Whereas most current research into computers and music focuses on the development of media technology for delivering music to consumers (e.g., MP3 format, Internet search engines, and so on). This research focuses on the development of technology for musical creativity. That is, technology to aid musicians to create content for the media. This paper focuses on a particular technology that the authors are developing, which is based on Artificial Life (A-Life).

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 one for further iterated refinement (Kim and Cho, 2006). 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 a particular cultural contexts and their reorganization and adaptation in response to cultural exchange.

Existing A-Life-based systems for musical composition normally employ Genetic Algorithms (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 the 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, including the authors, have used such systems to compose music. However, two problems have been identified with the GA-based approach to generative music. Firstly, a musical composition should not be driven by a constant set of fitness criteria. And secondly, music is largely a cultural phenomenon driven by social pressure and this is cumbersome to model with standard GA alone.

The first problem emerges from the fact that 'music is not an exact science'. For example, it differs from Engineering, whereas the success of a piece of Engineering would normally be measured by its ability to match a number of functional requirements effectively, the success of a piece of music cannot be measured so objectively. Indeed, whereas good engineers are praised for following the rules of their métier strictly, good composers (at least in the Western music tradition) are praised for clever violations of musical conventions. Moreover, in most cases, composers do not explicitly know a priority how a new piece of music will sound like, until it is completed and indeed performed. Therefore, rather than tools to generate efficient solutions to problems automatically, composers need tools to explore a vast space of possible outcomes. Biles (1994) proposed an interesting approach to implement GA-based systems for the exploration of a space of musical possibilities, which takes into consideration the evaluation of the user; that is, the user evaluates the fitness of each generation of "solutions". This is surely a very interesting idea, but this slows down the compositional process enormously. Biles is aware of this problem, which he refers to as the "fitness bottleneck" problem.

The second problem is largely related to a problem that is endemic in the field of Computer Music, which is the tendency to design systems to generate music from algorithms that were not designed for music in the first instance. For example, in the late 1980s it became fashionable to implement systems that generated music from fractals (Mandelbrot, 1982). There was a tendency at that time to overstate the adequacy of fractals as generators of music. Nowadays, we may be witnessing a similar case of overstatement on the adequacy of GA as generators of music. Although we acknowledge that there have been a few rather successful stories (Biles, 1994), we believe that additional A-Life-based methods need to be developed in order to move the exciting field of *Evolutionary Computer Music* (Miranda and Biles, 2007; Miranda, 2004) forward.

One way forward is to build systems with A-Life algorithms designed or suitably modified to address musical issues. The work presented in this paper contributes to this line of thought 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 musical conventions. A plausible method to embed social dynamics in such algorithms is to design them within the framework of interacting autonomous software agents.

This paper introduces three algorithms of the authors' own design, referred to as *popularity*, *transformation* and *complexity* algorithms, respectively. These algorithms are used to implement a system for the composition of rhythms where the user can generate rhythmic sequences and also monitor the behaviour of the system. The system offers the ability to extract information about the behaviour of the agents and the evolving rhythms in many different ways, providing the composers with means to explore the outcomes systematically. This paper will focus on the algorithms themselves, the

information that one can extract about their behaviours and the analyses of the behaviours. An in-depth discussion on how the algorithms are used artistically to compose pieces of music falls beyond the scope of this paper.

By way of related research, the work by De Boer (De Boer, 1999) on modeling the emergence of vowel systems by means of imitations games is cited. Also, Miranda (Miranda, 2002) has developed a model of the emergence of intonation systems using imitation games. This research is inspired by the work developed by research into gaining a better understanding of the evolution of language with computer models (Kirby, 2002; Vogt, 2000), particularly the work of Steels (Steels, 1995) on language imitation games with software agents. 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.

1. The Agents

In this section, the agents and their "cognitive ability"are introduced; that is, the operations that they are able to perform on rhythms.

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 (Figure 1) 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 the 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 interaction algorithm being used 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 (Handel, 1989). Moreover, humans show a preference for rhythms composed of

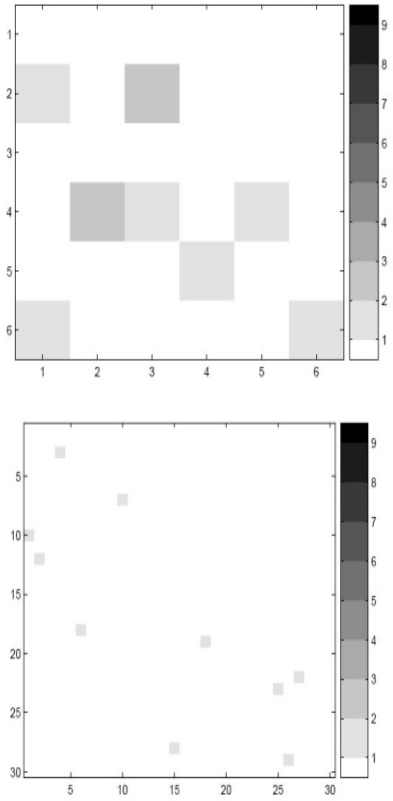

Figure 1. 2D virtual worlds with different sizes holding 10 agents. A darker colour indicates a cluster. (This will be clarified in due course)

integer ratios of the basic isochronous pulse (Drake and Bertrand, 2001). Therefore, rhythms are represented here as interonset intervals in terms of small integer ratios of an isochronous pulse (Figure 2).

1.1 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 (Figure 3a).

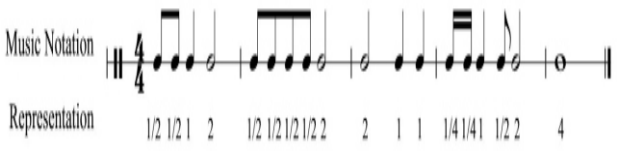

Figure 2. Standard music notation of a rhythmic sequence and its corresponding interonset representation

- Merge two rhythmic figures (Figure 3b).
- Add one element to the sequence (Figure 3c).
- Remove one element from the sequence (Figure 3d).

The definition of these transformations were inspired by the dynamical systems approach to study human bimanual coordination (Kelso, 1984) and is based on the notion that two coupled oscillators will converge to stability points at frequencies related by integer ratios (Beek et al., 2000). Furthermore, common music notation facilitates these types of transformations. Other transformations that divide a figure into three, five and other prime numbers are defined, but the impact of these additional transformations on the model is beyond the scope of this paper. Addition and removal transformations were introduced to increase the diversity in the pool of rhythms and to produce rhythms of different lengths.

1.2 Measurement of 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 that they play to each other. Also, this algorithm is used to measure the similarity between the repertoires of rhythms from different agents.

In the previous paper (Martins et al., 2005), a method to measure the degree of similarity between two sequences of symbols was introduced 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, a version of the SCV method that deals with rhythmic sequences is devised, which is introduced below.

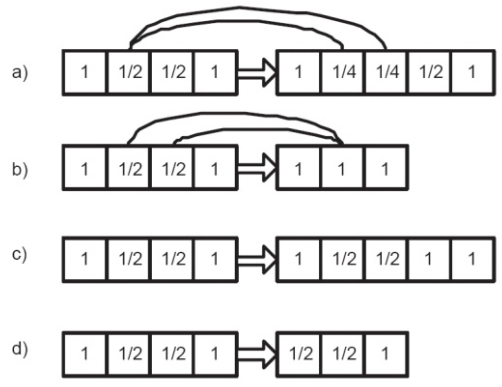

Figure 3(a - d). Examples of rhythmic transformations

Let us define the block distance between two sequences containing the same number of elements as follows (1):

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

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 (2):

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

.....(2)

Where *d* is the distance defined by Equation 1 between all the subsequences $v^{(k)}$ of v and all the subsequences $w^{(k)}$ of w. Next, let us define the *k*-level *Similarity Coefficient* as follows (3):

$$c^{(k)}(\mathbf{v}, \mathbf{w}) = \frac{z(k)}{(n-k+1)(m-k+1)} \qquad \text{.....(3)}$$

Where, *z(k)* is the number of zeros in the matrix $D^{(k)}$. Roughly speaking, the similarity coefficient measures the sparsity of the matrix $D^{(k)}$. The higher the coefficient *c(k)*, the higher is the similarity between the subsequences of level *k*.

Next, we can collect all the *k*-level coefficients in a vector referred to as *Similarity Coefficient Vector* (SCV). This is defined as follows (4):

$$\mathbf{C} = \left[c^{(1)}, c^{(2)}, \dots, c^{(min(m,n))} \right] \qquad \text{.....(4)}$$

Figure 4 shows an example of building a 3-level Distance Matrix and its respective SCV is *SCV* = [0.4167 0.1333 0.1250 0].

From this vector 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 (5):

$$SCV_{av} = \frac{1}{min(m,n)} \sum_{k=1}^{min(m,n)} SCV(k) \quad \text{.....(5)}$$

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 reveals the similarity between their repertoires (Figure 5).

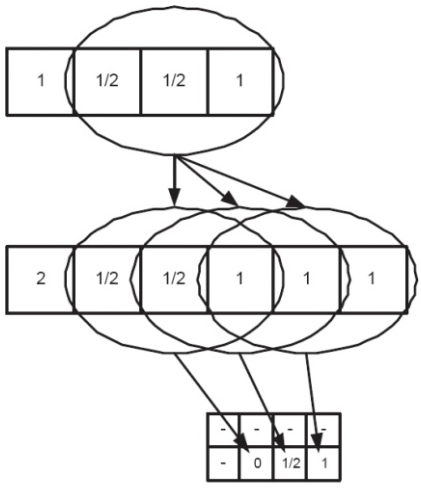

Figure 4. Example of building a 3-level Distance Matrix

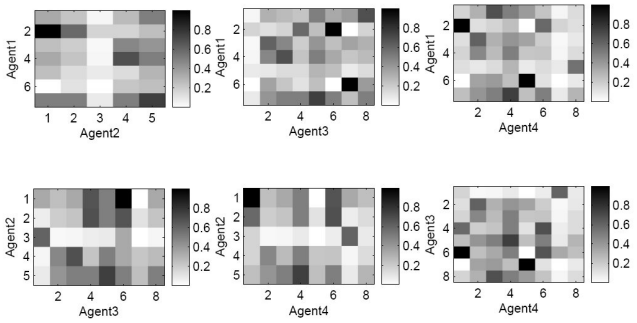

Figure 5. Similarity matrices between the repertoires of 4 agents. The darker the colour, the more similar the rhythms

By collapsing both the rows and the columns of the matrices, and taking the maximum values for each of them and an averaged sum,the scalar of similarity between repertoires is obtained, as follows (6):

$$SimRep_{k,l} = \frac{1}{nR_{Ak} + nR_{Al}} \left[\sum_{i=1}^{nR_{Ak}} max(SCV_{av})_{rows} + \sum_{j=1}^{nR_{Al}} max(SCV_{av})_{cols} \right] \quad \text{.....(6)}$$

Where the first term corresponds to the sum of the maximum values of the SCV_{av}, defined in Equation 5, 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.

In the case shown in Figure 6, it would be $SimRep_{k,l} = 0.7$ or, conversely, the distance between the repertoires of both the agents *k* and *l* would be $DistRep_{k,l} = 1 - SimRep_{k,l} = 0.3$. The values of 0.65 and 0.8 in Figure 6 corresponds to the similarity of the repertoires from the point of view of each agent, which is used to

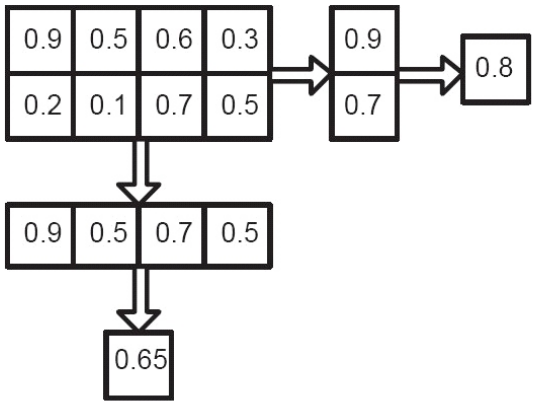

Figure 6. Scalar value of similarity between repertoires

generate proximity matrices and graphs to monitor the behaviour of the evolving repertoire of rhythms and the agents.

Finally, the development of repertoires of rhythms for 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 (7)

$$D_{AB} = \frac{1}{N_A . N_B} \sum_i d_i \qquad \text{.....(7)}$$

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 Figure 11. This dendrogram is drawn through an iterative process until all the individuals or clusters are linked.

1.3 Measurement of Complexity of Rhythms

The complexity of a rhythmic sequence is measured as follow

$$Complexity = \frac{nF + \sum_{i=1}^{nF} n_i}{\sum_{i=1}^{nF} T_i} \qquad \text{.....(8)}$$

Where *nF* is the number of rhythmic figures contained in the sequence, n_i is the 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. Figure 7 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.

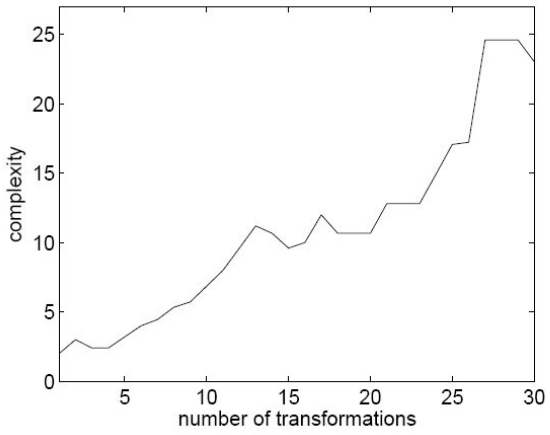

Figure 7. The complexity of a given rhythmic sequence increases in function of the number of transformations applied to it

2. The Algorithms

This section introduces the three proposed algorithms and the respective analysis methods that are implemented. Each algorithm is introduced in the context of illustrative experiments aimed at studying the development of repertoires of rhythmic sequences from three different perspectives:

- From the perspective of an individual agent.
- From the perspective of a group of agents, referred to as *the society*.
- From the perspective of the developed rhythms.

From the perspective of an individual agent, the development of the size and the complexity of the repertoire of individual agents is analyzed. From the perspective of the society, the values of the corresponding individual measures from the agents, as well as the similarity between the agents and how they are clustered in terms of the rhythms they share are analyzed. Finally, from the perspective of the developed rhythms, their lifetime, the amount of rhythmic sequences that the society develops and the degree to which the agents share similar rhythms are measured. The lifetime of a rhythmic sequence is traced by counting the number of agents that hold the sequence in their memories during the interactions. Figure 8 shows graphs illustrating these various types of analyses.

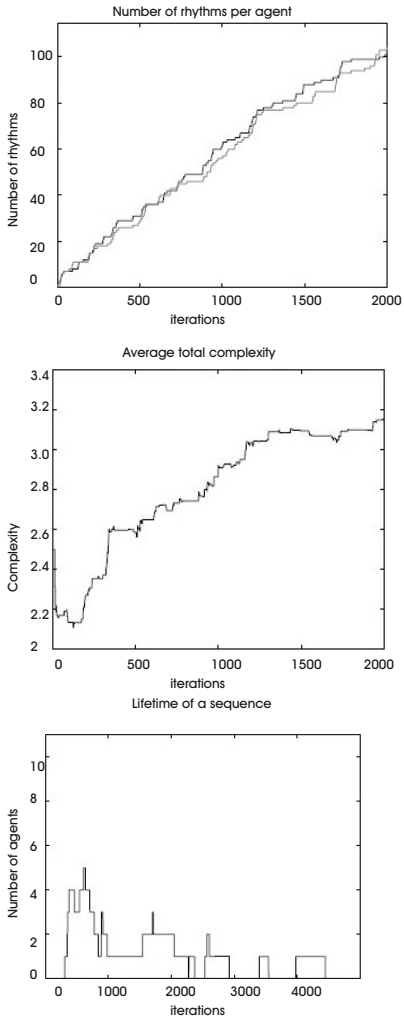

Figure 8. a) Development of the size of the repertoire for different agents; b) Complexity of the rhythms of the society; c)Number of agents sharing a particular rhythm

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. The experiments were run with societies of 3, 10 and 50 agents. For some of the experiments the lifetime of the agents were limited 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.

2.1 The Popularity Algorithm

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 when they interact with each other. 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 iteration all the rhythms have their popularity index decreased by a small value of 0.05. This accounts for a natural drop in the popularity index due to ageing. A diagram summarising the popularity algorithm is displayed in Figure 9.

Figure 10 shows the results after 5000 iterations of the popularity algorithm without population renewal.

Figure 10a displays the development of the repertoire of individual agents and Figure 10b displays the corresponding average across all the agents. Here the repertoires of the agents grow steadily upto approximately 1000 iterations and subsequently oscillates around a stable point. Figure 10c displays the

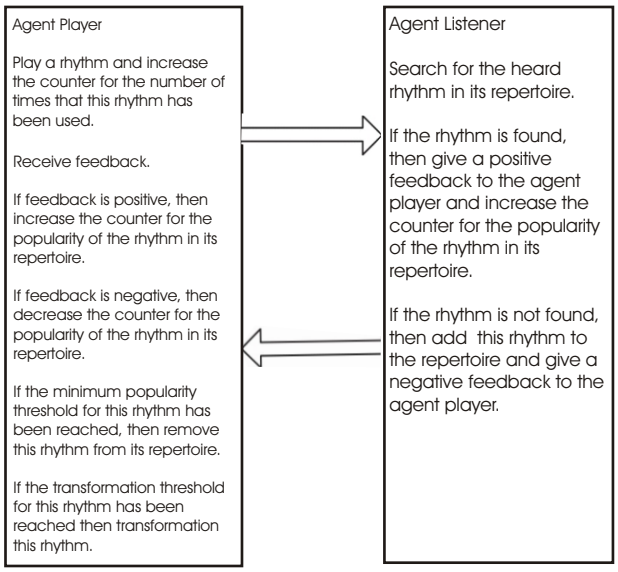

Figure 9. The popularity algorithm

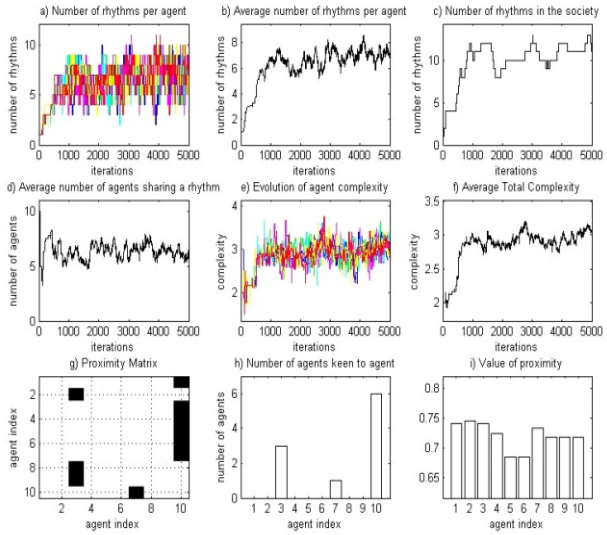

Figure 10(a-i). Results from a typical run of the popularity algorithm with 10 agents

development of the repertoire of the whole society being a direct consequence of the lifetime of each rhythm. The average number of agents sharing a rhythm (Figure 10d) 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 (Figure 10c). This graph (Figure 10d) provides the means to assess the global behaviour of the society; for instance, if it develops coherently in terms of the popularity of the existing rhythms. Figure 10e represents the development of complexity of the individual agents and Figure 10f gives the corresponding average. Initially, the size and complexity of the repertoire of individual agents are very close to average, but this trend is replaced quickly by the 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. Figure 10g displays the information about the identity of the agent with whom each agent relates most; i.e., one which has the highest similarity value. The graph in Figure 10h 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 the groupings of agents according to the similarity of their repertoires. Figure 11 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 the three 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. The differences in the clusters within a given society can also be observed.

By allowing the agents to move in their environment, it is also been investigated whether the popularity algorithm (an others) 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 of the algorithm.

Figure 12 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.

In Figure 13, 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

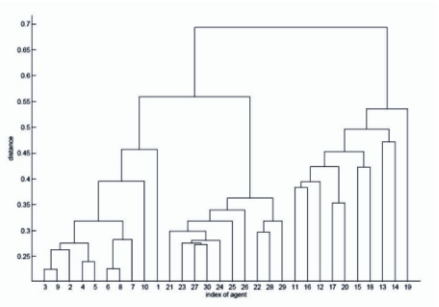

Figure 11. Dendrogram resulting from the hierarchical cluster analysis conducted in the context of the popularity algorithm containing three independent societies with 10 agents each

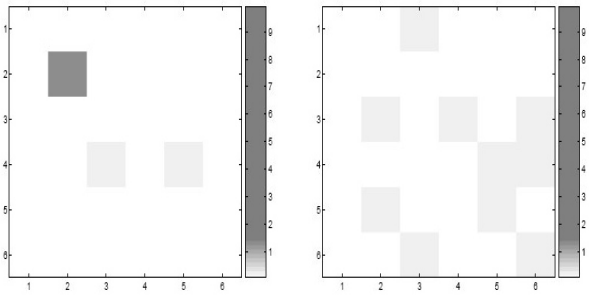

Figure 12. Clustering can take place (figure on the left) followed by scattering at a later stage (figure on the right). A cluster is indicated by a darker colour

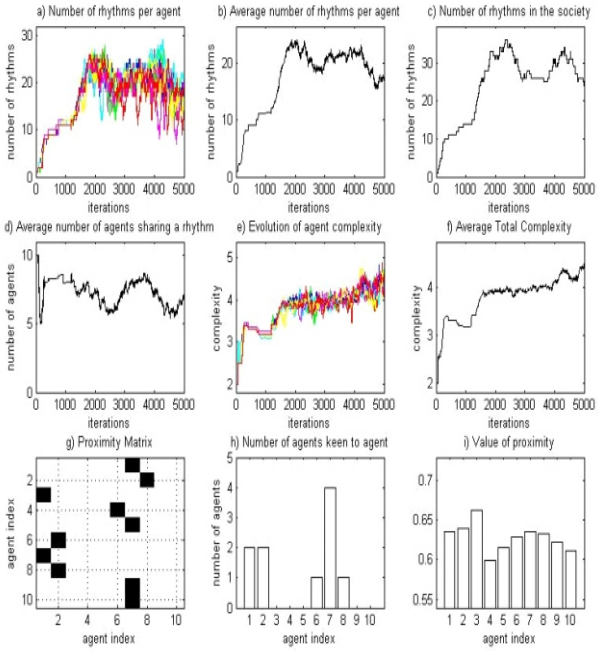

Figure 13. Results from a typical run of the popularity algorithm taking into account the movement of the agents as an influencing factor in the evolution of the repertoire

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 between them 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.

Figure 14 shows the lifetime of sequences that emerged during the typical runs of the popularity algorithm.

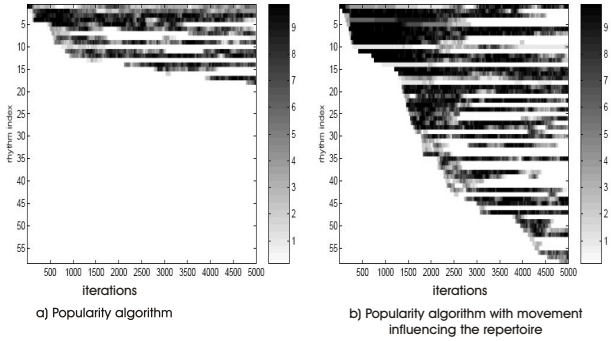

Figure 14. The lifetime of all rhythms that emerged using the popularity algorithm in cases where: a) movement does not influence the developments, and b) movement influences the developments. The number of agents that share a particular rhythm is represented by tones of grey; that is, the darker the colour the higher the number of agents sharing a particular rhythm

2.2 The Transformation Algorithm

A diagram summarising the transformation algorithm is given in Figure 15. As its name suggests, the transformation algorithm applies transformations on a rhythm whenever it is communicated between agents. The motivation behind this algorithm is to foster novelty in the repertoires of the agents. The transformation algorithm allowed for experiments aimed at assessing the degree to which the transformations occurring during the interactions have an impact on the organisation of the emerging repertoire as time progresses.

It is possible to observe in Figure 16 that due to the rise of the amount of transformations, the repertoires are much larger than in the popularity algorithm. For instance, compare Figure 10b with Figure 16b. Figure 16f shows the development of the average complexity of the society, where we can observe two clearly differentiated growing rates before and after 200 iterations. When this algorithm is run with 50 agents we can also observe similar growing rates, although the initial rate is not as steep as it is with 10 agents, and the transition is smoothed (Figure 17).

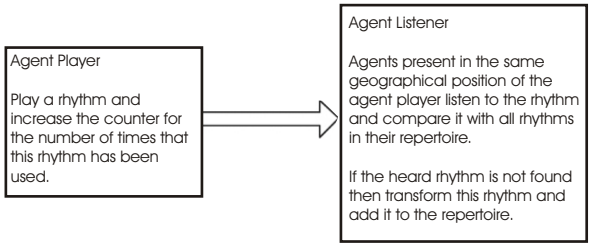

Figure 15. The transformation algorithm

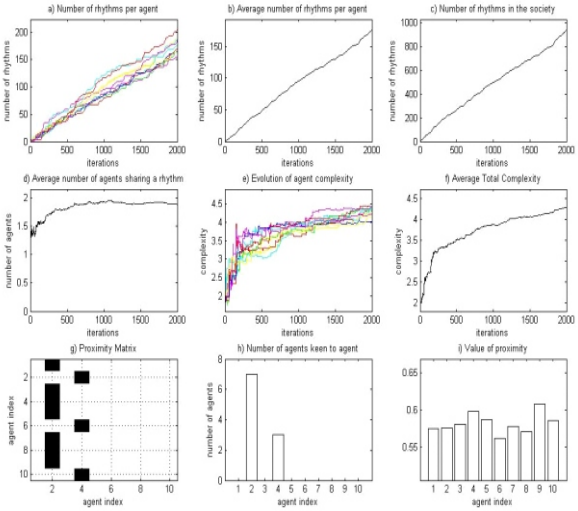

Figure 16. Results from a typical run of the transformation algorithm with 10 agents

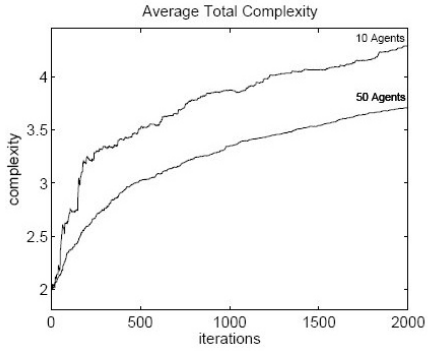

Figure 17. Average complexity evolution curves resulting from the transformation algorithm with 10 and 50 agents

2.3 The Complexity Algorithm

A diagram summarising the complexity algorithm is given in Figure 18. With the complexity algorithm one can study the effect of preference for particular types of rhythms. For instance, we can observe whether the agents would show preference for rhythms with identical complexity.

With the complexity algorithm, the agents include in their repertoire only those listened rhythms that fall within a window of complexity centred in the average complexity of the rhythms of the listening agent. That is, all listened rhythms that fall within the interval of [*AvComplexity* - 1, *AvComplexity*+1] will be included in the repertoire of the agent.

Figure 19 displays the results from a run of the complexity algorithm with the same parameters as for the run of the

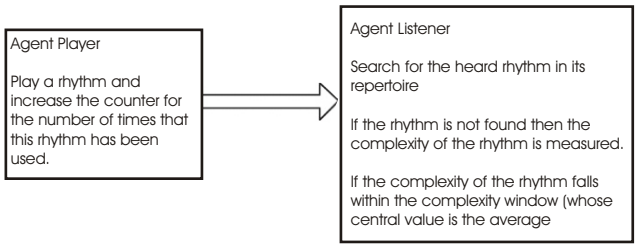

Figure 18. The complexity algorithm

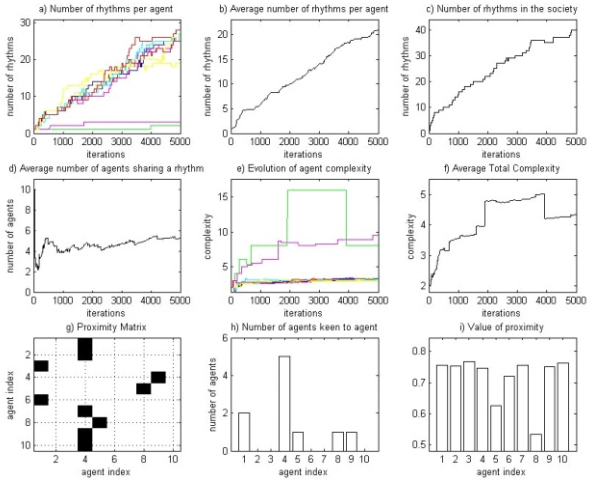

Figure 19. Results from a typical run of the complexity algorithm with 10 agents

popularity algorithm shown in Figure 10. The most interesting emergent behaviour that can be observed from these graphs is the distinct repertoires developed by the agents 5 and 8; they are distinct in terms of the complexity (represented by two distinct plots in Figure 19e) and the number of developed rhythms (represented by distinct plots at the bottom on the graph in Figure 19a). Although they are considered to have the smaller values of proximity in relation to the closer agent (Figure 19i), their development seems to be tightly connected. It is seen here that initial small changes in complexity due to transformations can actually result in completely different developments between the agents.

The cluster tree for the results shown in Figure 19 is given in Figure 20. Two main clusters appear, separated by a value of *DistRep* = 0.8. Furthermore, the two agents that at an early stage of the simulation were able to perform transformations leading to sequences of higher complexity remain more apart than the agents of the other cluster.

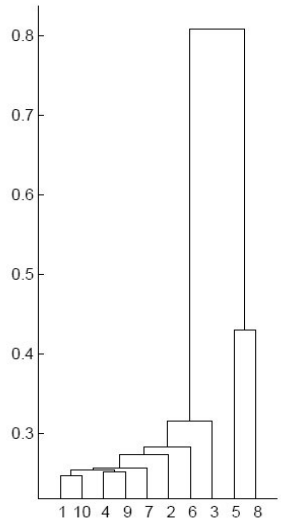

Figure 20. Dendrogram resulting from the hierarchical cluster analysis conducted in the context of the complexity algorithm with 10 agents

Conclusion

Most current approaches using A-Life in software for generating music entail the application of a GA. It was suggested 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 Engineering 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, the authors suggested 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, three algorithms inspired by A-Life were proposed: the popularity, transformation and complexity algorithms, respectively. In addition, a number of methods were developed to monitor the behaviour of the algorithms.

In all runs of the three algorithms, the emergence of coherent repertoires were observed 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 first algorithm, it tends to grow constantly in the other two algorithms. It is understood that this behaviour would change if the lifetime of the agents is limited, which would imply some form of population renewal. This might increase the role of the memory loss mechanism and therefore constrain the growth of the repertoire. Also a small subset of agents that tend to concentrate the preference of most of the population was observed. This trend tended to appear in many runs with different settings, in all the three algorithms. In the third algorithm we observed that large clusters of agents tended to appear, grouped according to the complexity and average number of rhythms.

As mentioned earlier, a system for the composition of rhythms was implemented using the three algorithms introduced in this paper. The analysis methods shown above offer the ability 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. However, an in-depth discussion on how the system is used artistically to compose pieces of music falls beyond the scope of this paper, and shall be reported in the future.

Acknowledgement

This project was developed thanks to a grant from The Leverhulme Trust. The authors are most grateful for this support.

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]. Mandelbrot, B. B. (1982). *The fractal geometry of nature*. New York, USA: W.H. Freeman and Co.

[11]. 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).

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

[13]. 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.

[14]. 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.

[15]. Steels, L. (1995). "A self-organizing spatial vocabulary". *Artificial Life*, 2(3):319-332.

[16]. Vogt, P. (2000). *Lexicon Grounding on Mobile Robots*. PhD thesis, Vrije Universiteit Brussel.

ABOUT THE AUTHORS

** Ph.D Research Student, Interdisciplinary Centre for Computer Music Research (ICCMR), University of Plymouth, Plymouth, UK.*
*** Professor in Computer Music, Interdisciplinary Centre for Computer Music Research (ICCMR), University of Plymouth, Plymouth UK.*

Joao M. Martins received his BEng in Electrical Engineering from the University of Coimbra, Portugal. He is currently pursuing his Ph.D on the topic of Computer Music at the Interdisciplinary Centre for Computer Music Research (ICCMR) under the guidance of Prof. Miranda. His research is concerned with Artificial Life modeling of music evolution. He was one of the organizers of the Music-AL workshop on music and Artificial Life, as part of the European Conference on Artificial Life 2007.

Prof. Eduardo R Miranda received his M.Sc in Music Technology from the University of York, UK and Ph.D in Music from the University of Edinburgh, UK. He served as a Research Scientist at SONY Computer Science Laboratory in Paris, France, before moving to the University of Plymouth in 2003 to establish the Interdisciplinary Centre for Computer Music Research (ICCMR) in the Faculty of Technology. He is currently the chair of the School of Computing, Communications and Electronics Research Committee and is the head of ICCMR.