

On the Music of Emergent Behaviour

What can Evolutionary Computation bring to the Musician?

Eduardo Reck Miranda
Sony Computer Science Laboratory Paris
6 rue Amyot - 75005 Paris - France
miranda@csl.sony.fr

Abstract

This paper focuss on issues concerning musical composition practices in which the emergent behaviour of computational processes is used to generate either musical material, musical form, or both. Special attention will be given to the potential of cellular automata and adaptive games for music-making. We begin the paper with an introduction to CAMUS and Chaosynth, two cellular automata-based music systems of our own design, followed by an assessment of their role in the composition of a number of successful pieces. The paper continues with a discussion on the potential and limitations of computational models for music composition followed by concluding remarks whereby the author suggests that adaptive imitation games may hold the key to foster more effective links between evolutionary computation paradigms and creative musical processes.

1 Introduction

In this paper we depart from the principle that the discussion as to whether or not computers can compose music is no longer relevant: computers *can* compose if programmed accordingly. Perhaps one the greatest achievements of Artificial Intelligence (AI) to date lies in the construction of machines that can compose music of incredibly good quality indeed; Cope's EMI system comes to mind here (Cope 1991). One must not forget, however, that these AI systems are good at mimicking well-known musical styles. They are either hard-wired to compose in a certain style or able to learn how to imitate a style by looking at patterns in a bulk of training examples. Such systems are therefore good for imitating composers of musical styles that are well-established, such as medieval, baroque or jazz (Miranda 2000a). Conversely, issues as to whether computers can create new kinds of music is much harder to study, because in such cases the computer should neither be embedded with particular models at the outset nor learn from carefully selected examples. Furthermore, it is hard to judge what the computer creates in such circumstances because the results normally sound very strange to us. We are often unable to judge these computer-generated pieces because they tend to lack those cultural references that we normally hold on to when appreciating music.

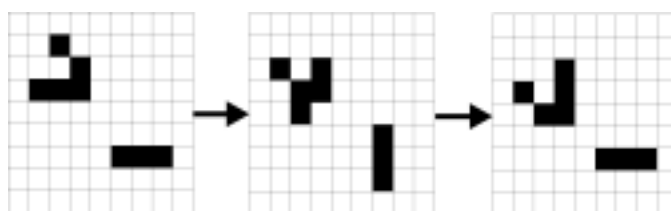
One plausible approach to these problems is to program the computer with abstract models that embody our understanding of the dynamics of some compositional processes. Since the invention of the computer, many composers tried out mathematical models which were thought to embody musical composition processes, such as combinatorial systems, stochastic models and fractals (Dodge and Jerse 1985; Xenakis 1971; Warrall 1996). Some of these trials produced interesting music and much has been learned about using mathematical formalisms and computer models in composition. The potential of evolutionary computation is therefore a natural progression for computer music research. In this paper we introduce a discussion on the potential of cellular automata and adaptive games for the composition of truly new music, but music that has potential to make sense to our ears. The paper begins with a brief introduction to CAMUS and Chaosynth, two cellular automata-based music systems of our own design, followed by an assessment of their role in the composition of a number of successful pieces.

2 Modelling the propagation of musical forms using cellular automata

CAMUS uses two simultaneous cellular automata to generate musical forms: the Game of Life and Demon Cyclic Space. Due to limitations of space, we will briefly introduce only the role of the Game of Life in its generative process. More information on CAMUS can be obtained in a recent paper that appeared in the *Computer Music Journal* (McAlpine, Miranda and Hoggar 1999).

The Game of Life is a two-dimensional CA that attempts to model a colony of simple virtual organisms. In theory, the automaton is defined on an infinite square lattice, but for practical purposes it is normally defined as consisting of a finite $m \times n$ array of cells, each of which can be in one of two possible states: alive represented by the number one, or dead represented by the number zero; on the computer screen, living cells are coloured black and dead cells are coloured white (Figure 1). The state of the cells as time progresses is determined by the state of their eight nearest neighbouring cells. There are essentially four rules that determine the fate of the cells at the next tick of the clock: a) Birth: A cell that is dead at time t becomes alive at time $t + 1$ if exactly three of its neighbours are alive at time t ; b) Death by overcrowding: A cell that is alive at time t will die at time $t + 1$ if four or more of its neighbours are alive at time t ; c) Death by exposure: A cell that is alive at time t will die at time $t + 1$ if it has one or none live neighbours at time t ; and d) Survival: A cell that is alive at time t will remain alive at time $t + 1$ only if it has either two or three live neighbours at time t . Whilst the environment, represented as E , is defined as the number of living neighbours that surround a particular live cell, a fertility coefficient, represented as F , is defined as the number of living neighbours that surround a particular dead cell. Note that both the environment and fertility vary from cell to cell and indeed from time to time as the automaton evolves. In this case, the life of a currently living cell is preserved whenever $2 \leq E \leq 3$ and a currently dead cell will be reborn whenever $3 \leq F \leq 3$. Clearly, a number of alternative rules can be set. The general form for such rules is $(E_{min}, E_{max}, F_{min}$ and $F_{max})$ where $E_{min} \leq E \leq E_{max}$ and $F_{min} \leq F \leq F_{max}$. The CAMUS implementation of the Game of Life algorithm enables the user to design rules beyond Conway's original rule. However rules other than (2, 3, 3, 3) may exist, but not all of them produce interesting emergent behaviour.

Figure 1:
Game of Life in action.



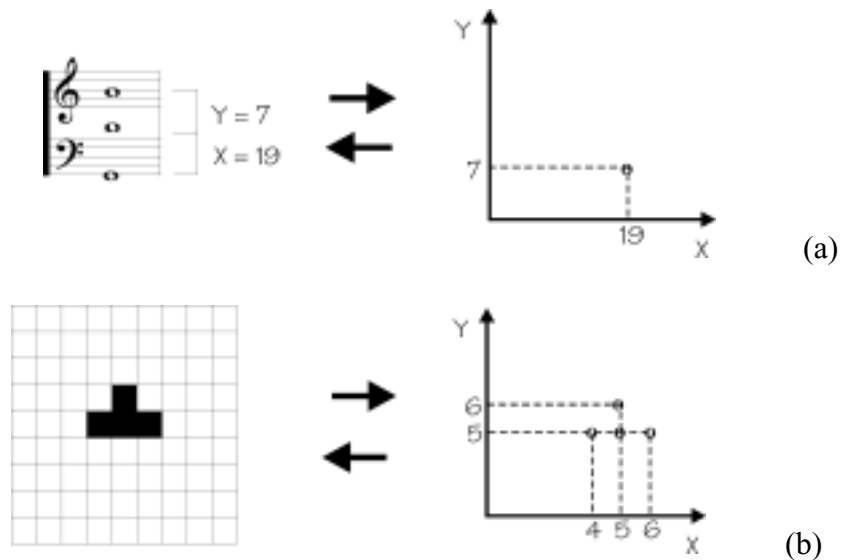
CAMUS uses a Cartesian model in order to represent a triple of notes. In this context, a triple is an ordered set of three notes, that may or may not sound simultaneously. These three notes are defined in terms of the distances between them, or intervals in music jargon. The horizontal co-ordinate of the model represents the first interval of the triple and the vertical co-ordinate represents its second interval (Figure 2a).

To begin the musical generation process, the CA are set up with an initial random configuration and are set to run. When the Game of Life automaton arrives at a live cell, its co-ordinates are taken to estimate the triple from a given lowest reference note (Figure 2b). For example, the cell at position (5, 5) in Figure 2b is alive and will thus generate a triple of notes. The co-ordinates (5, 5)

describe the intervals of the triple: a fundamental pitch is given, then the next note will be at five semitones above the fundamental and the last note ten semitones above the fundamental. Although the cell updates occur at each time step in parallel, CAMUS plays the live cells column by column, from top to bottom. Each of these musical cells has its own timing, but the notes within a cell can be of different lengths and can be triggered at different times. Once the triple of notes for each cell has been determined, the states of the neighbouring cells in the Game of Life are used to calculate a timing template, according to a set of temporal codes.

Figure 2:

- (a) CAMUS uses a Cartesian model in order to represent a triple.
 (b) Each screen of the Game of Life automaton produces a number of triples.



3 Emergent sound synthesis

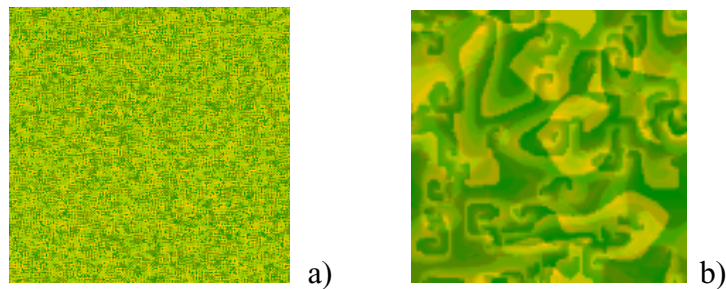
Chaosynth is essentially a granular synthesiser (Miranda 1995; 1998a). Granular synthesis works by generating a rapid succession of very short sound bursts called granules (e.g. 35 milliseconds long) that together form larger sound events. The results tend to exhibit a great sense of movement and sound flow. This synthesis technique can be metaphorically compared with the functioning of a motion picture in which an impression of continuous movement is produced by displaying a sequence of slightly different images (sound granules in our case) at a rate above the scanning capability of the eye. So far, most of these systems have used stochasticity to control the production of the granules; for example, to control the waveform and the duration of the individual granules. Chaosynth uses a different method: it uses cellular automata. The CA used in Chaosynth tends to evolve from an initial random distribution of cells in the grid, towards an oscillatory cycle of patterns (Figure 3).

The behaviour of this CA resembles the way in which most of the natural sounds produced by some acoustic instruments evolve: they tend to converge from a wide distribution of their partials (for example, noise) to oscillatory patterns; for example, a sustained tone. Chaosynth's CA can be thought of as a grid of identical electronic circuits called cells. At a given moment, cells can be in any one of the following conditions: *quiescent*, *depolarised* or *burned*. A cell interacts with its neighbours (4 or 8) through the flow of electric current between them. There are minimum (V_{min}) and maximum (V_{max}) threshold values which characterise the condition of a cell. If its internal

voltage (V_i) is under V_{min} , then the cell is quiescent (or polarised). If it is between V_{min} (inclusive) and V_{max} values, then the cell is being depolarised. Each cell has a potential divider which is aimed at maintaining V_i below V_{min} . But when it fails (that is, if V_i reaches V_{min}) the cell becomes depolarised. There is also an electric capacitor which regulates the rate of depolarisation. The tendency, however, is to become increasingly depolarised with time. When V_i reaches V_{max} , the cell fires and becomes burned. A burned cell at time t is automatically replaced by a new quiescent cell at time $t + 1$.

Figure 3:

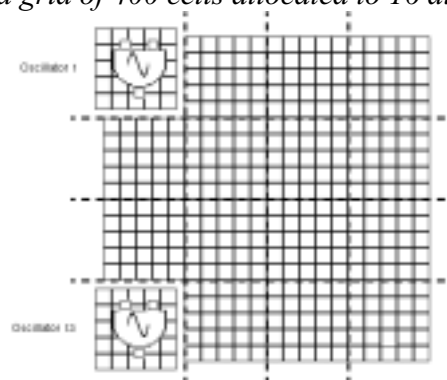
*The CA used in Chaosynth tends to evolve from
 (a) an initial random distribution of cells in the grid
 (b) towards an oscillatory cycle of patterns.*



Each sound granule produced by *Chaosynth* is composed of several spectral components. Each component is a waveform produced by a digital oscillator (i.e., a lookup sampling table containing one cycle of a waveform) which needs two parameters to function: frequency and amplitude. The CA controls the frequency and duration values of each granule (the amplitude values are set up via another procedure). The values (i.e., the colours) of the cells are associated to frequencies and oscillators are associated to a number of cells (Figure 4). The frequencies of the components of a granule at time t are established by the arithmetic mean of the frequencies associated with the values of the cells associated with the respective oscillators. Suppose, for example, that each oscillator is associated with 9 cells and that at a certain time t , 3 cells correspond to 110 Hz, 2 to 220 Hz and the other 4 correspond to 880 Hz. In this case, the mean frequency value for this oscillator at time t will be 476.66 Hz. The user can also specify the dimension of the grid, the amount of oscillators, the allocation of cells to oscillators, the allocation of frequencies to CA values, and various other CA-related parameters. An example of a grid of 400 cells allocated to 16 oscillators of 25 cells each is shown in Figure 4. The duration of a whole sound event is determined by the number of CA iterations and the duration of the particles; for example, 100 iterations of 35 millisecond particles results in a sound event of 3.5 seconds of duration.

Figure 4:

An example of a grid of 400 cells allocated to 16 digital oscillators.



4 Brief assessment

Despite the arbitrariness of the musical engine of CAMUS, we have come to conclude that CA is appropriate for generating musical material. We observed that CAMUS can produce interesting musical sequences. Indeed a number of professional pieces were composed using CAMUS-generated material, such as *Entre l Absurde et le Myst re*, for chamber orchestra, and the second movement of the string quartet *Wee Batucada Scotica* (Miranda 1998b). Chaosynth also proved to be a successful system in the sense that it is able to synthesise a large amount of unusual sounds that are normally not found in the real acoustic world, but nonetheless sound pleasing to the ear. As an example of an electroacoustic piece composed using the sounds of Chaosynth we cite *Olivine Trees*, which has recently been awarded the bronze medal at the International Luigi Russolo Electroacoustic Music Competition in Italy (the recordings of all these pieces are available by request). The results of the CA experiments are very encouraging, as they are good evidence that both musical sounds and abstract musical forms might indeed share similar organisational principles with cellular automata. We would like to make it clear, however, that none of the pieces cited above were entirely automatically generated by the computer. The programs produced the raw, but high-level material that were manually arranged for the final composition. Nevertheless, the computer-generated material was of good quality and as far as computer music is concerned, this is a great achievement. Without CAMUS and Chaosynth these pieces would probably never have been composed.

5 Discussion and conclusion

In general, we found that Chaosynth produced more interesting results than CAMUS. We think that this might be due to the very nature of the phenomena in question. The inner structures of sounds seem more susceptible to CA modelling than large musical structures. As music is primarily a cultural phenomenon, in the case of CAMUS we think that we would need to add generative models that take into account the dynamics of social formation and cultural evolution. In this case, we should find modelling paradigms where phenomena (in our case musical processes and forms) can emerge autonomously. We have strong evidence that the adaptive games paradigm might shed a clearer light onto this problem.

We have recently implemented an experiment whereby an imitation game has successfully simulated the emergence of rhythmic forms in a virtual community of agents (Miranda 2000). The agents were furnished with a speech synthesiser, an artificial ear and a memory mechanism. To begin with, the agents did not have any repertoire in their memory. The objective of the simulation was to let the agents build their own repertoire of rhythms by imitating each other. The only constraints of the game were imposed by the physiology of their speech and hearing apparatus and of course by the rules of the game. The rules of the game proceed as follows: two agents are selected for a round; one to play the role of a producer and the other to play the role of a listener. The producer plays a rhythm (composed of syllables /pa/) either selected from its repertoire or produced from scratch if there is nothing in the repertoire. The imitator in turn compares this rhythm with the ones it already knows. This imitator selects from its own repertoire the rhythm that sounds most similar to the rhythm it heard and then produces it. The producer compares the imitator's attempt with the one it had produced originally. If they are similar, then the game is a success; otherwise it is a failure. This result is communicated to the imitator by means of non-verbal feedback. Immediately after the game, both agents update their memories according to the result of the game. In short, if the game was successful, then both agents increase a success counter

for the rhythm that was used by both agents. If the game was a failure, then the imitator tries to shift the failed rhythm closer to the rhythm it heard, hoping that next time it will result in a better imitation. Sometimes rhythms that have not scored successfully for a long time are deleted from the repertoire. In other cases, rhythms that are too close to each other in the repertoire are merged by means of a quantiser mechanism. Despite the simplicity of the model, the results of the simulations are quite impressive. After a few thousand runs involving a few dozen agents, a coherent repertoire of rhythms that were shared by all agents emerged; we say that in this case the agents reached a cultural agreement. We are now scaling up this experiment in order to investigate whether such emergent phenomena also produce coherent results in situations involving pitch systems and higher level musical forms.

Should such experiments with adaptive games corroborate our hypothesis that we can improve algorithmic composition systems considerably by including mechanisms that take into account the dynamics of cultural evolution and social interaction, then we believe that a new generation of much improved intelligent composing systems will soon begin to appear over the next few years.

References

Cope, D., *Computers and Musical Style*, Oxford University Press, 1991.

Dodge, T. and Jerse, T. A., *Computer Music: Synthesis, Composition and Performance*, Schirmer Books, 1985.

McAlpine, K., Miranda, E. R. and Hoggar, S., Making Music with Algorithms: A Case-Study System, *Computer Music Journal*, Vol 23 No. 2, 1999.

Miranda, E. R., Granular Synthesis of Sounds by Means of a Cellular Automaton, *Leonardo*, Vol. 28, No. 4, 1995.

Miranda, E. R., *Computer Sound Synthesis for the Electronic Musician*, Focal Press, 1998a.

Miranda, E. R., *Wee Batucada Scotica*, musical score, Goldberg Edicoes Musicais, 1998b.

Miranda, E. R., (Ed.) *Readings in Music and Artificial Intelligence*, Gordon and Breach/Harwood Academic Publishers, 2000.

Worrall, D., Studies in metamusical methods for sound image and composition, *Organised Sound*, Vol. 1, No. 3, 1996.

Xenakis, I., *Formalized Music*, Indiana University Press, 1971.