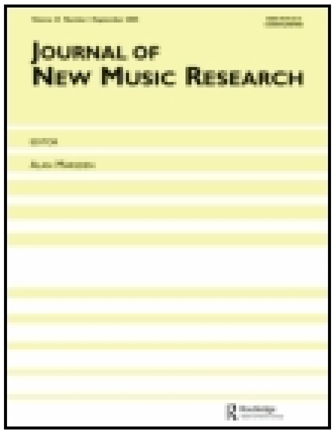


Journal of New Music Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/nnmr20

Investigating Music Pattern Formations from Heterogeneous Cellular Automata

Somnuk Phon-Amnuaisuk [a]
[a] Universiti Tunku Abdul Rahman , Malaysia
Published online: 09 Nov 2010.

To cite this article: Somnuk Phon-Amnuaisuk (2010) Investigating Music Pattern Formations from Heterogeneous Cellular Automata, Journal of New Music Research, 39:3, 253-267, DOI: 10.1080/09298215.2010.481360

To link to this article: http://dx.doi.org/10.1080/09298215.2010.481360

Journal of New Music Research
2010, Vol. 39, No. 3, pp. 253–267

Investigating Music Pattern Formations from Heterogeneous Cellular Automata

Somnuk Phon-Amnuaisuk

Universiti Tunku Abdul Rahman, Malaysia

Abstract

Patterns observed from melody lines, harmonic movements, textual appearances and formal structures in music pieces characterize individual composition. Composers exploit a large number of possible patterns and creatively compose a new piece of music by weaving various patterns together in a musically intelligent manner. To emulate high-level intelligent behaviours such as music composition skills, knowledge-intensive algorithmic composition approaches have been investigated with some successful outcomes. Nevertheless, a knowledge intensive approach has its limitations in a knowledge elicitation process. This paper discusses the applications of *heterogeneous cellular automata* (*hetCA*) in generating chorale melodies and Bach chorales harmonization. The machine learning approach is exploited in the learning of rewrite-rules for cellular automata. Rewrite-rules are learned from music examples using a *time-delay neural network* (*TDNN*). After each TDNN (hetCA model) has successfully learned musical patterns from the given examples, new patterns are generated from the hetCA model.

1. Introduction

Algorithmic composition has long been an intriguing topic for composers and computer scientists alike. Composers exploit the computing power to generate various patterns while computer scientists exploit the computing power to generalize various patterns into music grammars. Music theorists generally agree that there are patterns in tonal music compositions and these patterns are sometimes put into rule-like propositions for music education purposes. For example, *if the current tonality is a tonic in a major key (I) then it may stay in tonic or make a transition to subdominant (IV) or dominant (V)*, etc. This proposition may be coded in computing terms using various paradigms, e.g. as a disjunctive rule $I \rightarrow I$ or $I \rightarrow IV$ or $I \rightarrow V$; and as probabilistic rules $p(I|I)$, $p(IV|I)$ and $p(V|I)$. These patterns are essential ingredients for emulating intelligent behaviours using computers.

It has been demonstrated by previous works that for a system to generate pleasant tonal music, the system must possess sophisticated music knowledge (Fry, 1980; Cope, 1991, 2001; Ames & Domino, 1992; Ebcioglu, 1992; Ames, 1993). This knowledge is best captured by experts before being transformed and represented in computers. Two main challenges in building an intelligent system are (i) how is the knowledge obtained?, and (ii) how is the knowledge exploited? Although expert knowledge is generally more effective, it is difficult to elicit knowledge from experts. It is also difficult to exploit this knowledge since it always takes the shape of rules. Moreover, search space is exponentially increased as the number of applicable rules increase. Recently, progress in machine learning techniques offers alternative approaches that could respond to the challenges rising from knowledge elicitation and exploitation issues.

In our approach, the behaviour of CA was learned from given examples where the learned CA model was then used to generate a new piece of music. In this report, we organize the presentation into the following topics: (1) Introduction, (2) Related works, (3) Cellular automata, (4) Representing music using cellular automata, (5) Experiments and results, (6) Analysis and discussion, and (7) Conclusion.

Correspondence: Somnuk Phon-Amnuaisuk, Faculty of Creative Industries, Universiti Tunku Abdul Rahman, 13 Jalan 13/6, 46200 Petaling Jaya, Selangor, Malaysia. E-mail: somnuk@utar.edu.my

DOI: 10.1080/09298215.2010.481360 © 2010 Taylor & Francis

2. Related works

Formal approaches (e.g. grammar, probability laws, statistical analysis, stochastic processes, Markov chains, etc.) have been adopted by AI-music researchers as formalized composition processes (Baroni, Brunetti, Callegari, & Jacobni, 1982; Lerdahl & Jackendoff, 1983; Steedman, 1984; Xenakis, 1992; Temperley, 2007). Early computer-generated hymn tunes by Brook, Hopkins, Neumann, and Wright (1993) and Hiller's *Scherzo a Tre Voce* (1970, pp. 57–58) are such examples.

Compositions based on formalized techniques could normally be expressed in a concise and elegant manner. However, stylistic touches are quite hard to express with rigid formalism. This results from a lack of *flexible control* in the *composition process* generally associated with automated algorithms; since structures in the composition process are rigidly pre-defined in algorithms. Improvement in flexibility of control and expressiveness of control are crucial aspects of composition systems. The issue of flexibility and expressiveness of control was addressed by Ebcioglu (1992) using intelligent backtracking mechanisms and by Phon-Amnuaisuk, Smaill, and Wiggins (2006) using explicit control. Both works produced impressive chorale output as more knowledge was effectively explored and exploited by their systems. Note that Ebcioglu (1992) and Phon-Amnuaisuk et al. (2006) had their knowledge-bases explicitly handcrafted.

Other examples of handcrafted knowledge-based system are Cope's EMI systems (Cope, 1991, 2001). Cope dealt with control issues by providing a template to guide a newly generated composition. It was similar to the idea of a template approach used in our experiment. This tactic provided an effective way to control long-range dependency without actually having the knowledge learned by the system (it is difficult to learn this knowledge).

As a rule of thumb, the better the exploitation of knowledge, the better the output is. However, to explicitly handcraft domain knowledge is very hard. Many researchers, therefore, have chosen to extract domain knowledge via machine learning techniques. Systems from Hild, Feulner, and Menzel (1991) and Allan and Williams (2004) were such attempts. In their works, progression at the harmonic level were learned by *Neural networks* in Hild et al. (1991) and by *Hidden Markov Models* in Allan and Williams (2004). However, researchers could not rely solely on music knowledge learned from machine learning techniques to produce convincing output. Still, Hild et al. (1991) and Allan and Williams (2004) handcrafted knowledge of *functional harmony* in their melody-chord realization parts and in the harmonization processes (i.e. the division of the whole process into melody-functional harmony, voices and ornamentations sub-processes); Hild et al. (1991) also had explicitly handcrafted knowledge in the ornamentation part of the system.

In recent works from Assayag and Dubnov (2004), and Dubnov and Assayag (2005), an improvisation model was constructed using *Factor Oracle (FO)*. The factor oracle is an automaton constructed by learning from a sequence of symbols. The FO could compactly represent all *factors* in a sequence. The FO also provided *pointers* known as *forward links* and *suffix links* which could be used to recall seen sequences and generate new sequences. In Law and Phon-Amnuaisuk (2008), a hierarchical self-organizing map (HSOM) was explored for the generation of new folk melody from examples. These works were looking for a way to generate new sequences from learned examples by relying on machine learning techniques (without explicitly handcrafted domain knowledge).

Despite many attempts to learn music structure using various machine learning techniques, learning deep structural information was still a big challenge. In this paper, we investigated the *cellular automata* (CA) music composition system where it was used to represent music knowledge and processes for predicting a new pitch in a given context. In the CA paradigm, CA cells could be associated with any musical process. However, here, CA cells are associated with pitches where the presence and absence of pitches across time form a piano roll pattern. Instead of viewing music in traditional structures of form, melody, harmony and texture (West, Howell, & Cross, 1991; Smaill, Wiggins, & Miranda, 1993), music is seen as a time series pattern of a non-empty pitch set in which the behaviour of pitches are governed by both local and global dependencies among pitches. In this view, traditional music structures are implicitly captured in CA formations.

3. Cellular automata

Cellular automata (CA) are dynamic systems with discrete space and time (Wolfram, 2002). Space is represented as an array of cells and time is perceived as a discrete time step in each computing iteration. Each CA cell takes two or more discrete states and the state of each cell is updated based on the states of the cell and its neighbours' states. The update functions are referred to as *rewrite rules*. A cellular automaton is a simple yet powerful computing paradigm. Most CA share the following generic features:

I. CA consist of discrete lattices of cells.
II. Each cell takes two or more discrete states.
III. Each cell interacts with its neighbours and updates its current state according to transition rules.

Formally, we define cellular automata as follows:

Definition 1

A cellular automaton can be defined as a quintuple $CA = \langle A, S, A_0, N_r, f \rangle$ *where A is a discrete lattice of cells;*

S is a finite set of cell states; A_0 is a non-empty set of cells at the initialization $A_0 \subset A$; N_r: $a \rightarrow$ seq of a is a neighbourhood function that maps a given cell a to a sequence of cells where its members are determined by the neighbourhood structure of the cell a , the subscript $r \in Z^+$ denotes the radius of the neighbourhood; and f is a rewrite rule that maps a cell's state (conditioned by its neighbouring cell-states) to a new possible state $s \in S$. f: $S^{2r+1} \rightarrow S$; where S^{2r+1} is a sequence of states obtained from a neighbourhood structure of a, i.e. $N_r(a)$. Cellular automata (CA) are called homogeneous CA if all the cells in the CA share the same rewrite-rules (f). In heterogeneous CA, different cells have different rewrite rules.

For example, a one-dimensional binary CA with a neighbourhood size equal to three has eight possible state-combinations, i.e. {{1, 1, 1}, {1, 1, 0}, {1, 0, 1}, {1, 0, 0}, {0, 1, 1}, {0, 1, 0}, {0, 0, 1}, {0, 0, 0}}, hence, a 2^8 possible set of rewrite rules. Given a rewrite rule 18, an initial condition $A_0 = \{a_1, a_2\}$ where $a_1(t)$ and $a_2(t)$ take the state of 1 and 0 respectively. Assuming that the neighbourhood structure forming on $a_1(t)$ is {0, 1, 0} and on $a_2(t)$ is {0, 0, 1} then the new state of $a_1(t+1)$ would be 0 and the new state of $a_2(t+1)$ would be 1, respectively.

CA have been explored by many researchers, such as von Neumann (1966), in the demonstrations of universal computation; Conway, in the study of mathematical games (game of life); and Beyls (1989) and Miranda (1993), in experimental music. Figure 1 shows an example of a 1D cellular automaton initialized at time t_0 with a single black tile as a seed. A transfer function used in this example is known as *Rule 18*. In this example, the 1D CA is evolved from time t_0 to t_n. It is interesting to see that complex behaviours could be obtained from simple rewrite rules.

Observing CA patterns over time may reveal interesting temporal behaviours. The progression of the whole CA from $CA(t) \rightarrow CA(t+1)$, $t = 0, 1, 2, \ldots n$ reveals its temporal behaviours. Temporal information in CA could be accessed by representing a one-dimension CA as a two-dimension array where each column represents a one-dimension CA at each time-step (see Figure 1). In this sense, a *one-dimension binary CA* unfolding in time could be seen as a piano roll representation unfolding in time.

In this study, a piano roll at time *t* is represented by a column of CA. Let us denote each cellular as *a*, hence, an $i \times j$ matrix, $\mathbf{P}_{ij}$ represents a piano roll with *i* pitches over *j* time steps. Each $a_{i,j}$ cell takes a possible binary state value {0, 1}. The neighbourhood structure N_r for each $a_{i,j}$ cell could take many shapes, such as

$$N_r = \{a_{i-r,j}, a_{i-r+1,j}, \ldots, a_{i,j}, \ldots, a_{i+r-1,j}, a_{i+r,j}\}$$

if the neighbourhood cells are taken within the same time step *t*, or

$$N_r = \{a_{i,j-2r}, a_{i,j-2r+1}, \ldots, a_{i,j-1}, a_{i,j}\}$$

if the neighbourhood cells are taken horizontally across many time steps. The neighbourhood structure could be from the adjacent cells as in the examples above or from other neighbourhood structures.

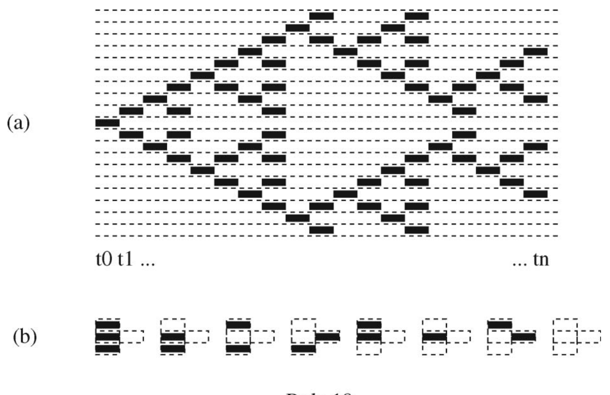


Fig. 1. (a) The CA formation formed by Rule 18 on a single black cell seed and (b) visualization of Rule 18 (i.e. {1, 1, 1}→ 0, {1, 1, 0}→ 0, {1, 0, 1}→ 0, {1, 0, 0}→ 1, {0, 1, 1}→ 0, {0, 1, 0}→ 0, {0, 0, 1}→ 1, {0, 0, 0}→ 0). Take note that a binary 00010010_2 is 18_{10} in decimal number.

Boundary conditions of the cellular automata

Since the CA is not infinitely extended up or down in the pitch height direction, we must deal with the boundary conditions of the CA. Three common techniques are: (i) fixed boundary, where a predefined extended boundary is used; (ii) reflective boundary, where the extended boundary is the mirror image of the cells at the boundary; and (iii) periodic boundary, where the extended boundary is the wrapped boundary on the opposite end. In this implementation, the wrapped boundary was used.

3.1 Music pattern formations from cellular automata

Although algorithmic music composition with cellular automata has been investigated before by previous works such as Beyls (1989), Miranda (1993), McAlpine, Miranda, and Hoggar (1999), Millen (2004), Brown (2005), Ariza (2007) and Wolframtones (2009)—see more reviews in Burraston and Edmons (2005), previous works in algorithmic music composition looked at CA compositions from an experimental music perspective (i.e. focus on interestingness of generated music pieces using the CA paradigm). The music was commonly generated by mapping rules that map CA patterns to note patterns. This work, however, looks at CA from a different perspective. We are more interested in *the issue of whether CA would be able to learn music examples, recall the examples and generate new compositions in the style of what it has learned.* Here, we investigate the use of heterogeneous CA to learn and generate chorales.

CA can be used to represent a piano roll, an $i \times j$ matrix $\mathbf{P}_{ij}$. Let $a_{i,j}$ denote a cell a with pitch i at time j where $i \in \{C2, C\#2, \ldots, B6\}$. To simplify the notation, let us think of i as an integer from 1 to 60 where integer 1 denotes C2, 2 denotes C#2 and so on. A sequence $a_{i-r,j}, a_{i-r+1,j}, \ldots, a_{i,j}, \ldots, a_{i+r,j}$ is a single dimension CA in which the prediction of $a_{i,j+1}$ is governed by $a_{i,j}$ and its neighbouring cells.

$$a_{i,j+1} = f(a_{i-r,j}, \ldots, a_{i,j}, \ldots, a_{i+r,j}). \tag{1}$$

If $a_{i,j}$ is a binary state cell where the two states represent a note-on and a note-off state, then a sequence $a_{i-r,j}, \ldots, a_{i+r,j}$ could represent 2^{2r+1} patterns.

In our representation, the note duration is the amount of time. Here, we fix a discrete time step t at a duration equivalent to a quaver note. In this paper, we employ a neural network to learn the rewrite-rules from given examples. The learned model is later used to generate (i.e. predict) a new note. The prediction of a new note in this manner is deterministic (see Equation (1)) and local in nature. Also, it does not exploit any historical information. We shall discuss how to include historical information (e.g. memory) and long-range dependency into the model next.

3.1.1 Exploiting memory in predicting a new note

Predicting a new note using Equation (1) exploits the local memory from the neighbourhood structure of CA. The power of associative memory in this style could be enhanced by increasing the neighbourhood size. Historical contexts could also be included in the system by including past events. In this fashion, the prediction of the next note at $a_{i,j+1}$ could be conditioned by the context of input from $a_{i,j}, \ldots, a_{i,j-n}$.

$$\begin{aligned} a_{i,j+1} = f(&a_{i-r,j}, \ldots, a_{i+r,j}, a_{i,j-1}, \ldots, a_{i+r,j-1}, \ldots, \\ &a_{i,j-n}, \ldots, a_{i+r,j-n}). \end{aligned} \tag{2}$$

4. Representing music using cellular automata

In our approach, we explored variations of representation from a CA perspective. Instead of representing music as a score where the melody line would be naturally represented as a sequence of note events, our input–output pair was a sequence of $\langle$ *set of input notes, set of output notes* $\rangle$. Figure 2 illustrates the transition of CA with three input notes and one output note. In this example, an input of three CA cells is applied to a piano roll of size seven. This requires seven successive steps from top to bottom. The output from each successive step determines the new piano roll at time $t+1$. Notice an extra one unit wrapped up in the cell at the top and bottom boundary (three input cells need one extra cell at the boundary, five input cells would need two extra cells and so on).

There could be many variations from the representation scheme discussed above. Figures 3(a) and (b) illustrate two main representation styles of the given *Ode to Joy* theme. The example in Figures 3(a) illustrates a neighbourhood structure across time and the example in Figures 3(b) illustrates a neighbourhood structure within the same time step. Readers may already see that there could be many variations to this representation scheme and this offers various ways to capture different contexts (e.g. the neighbourhood structure may not be adjacent cells). Loosely, representation in the style of Figures 3(a) captures the local melodic context while representation in the style of Figures 3(b) captures the local harmonic context. The terms local melodic contexts and local harmonic contexts are used here since the prediction of a new note is dependent on the information of $\mathbf{P}_{ij}$, i.e. $p(note|\mathbf{P}_{ij})$. Increase in neighbourhood size in the i direction would capture the harmonic contexts while increase in neighbourhood size in the j direction would capture the melodic contexts.

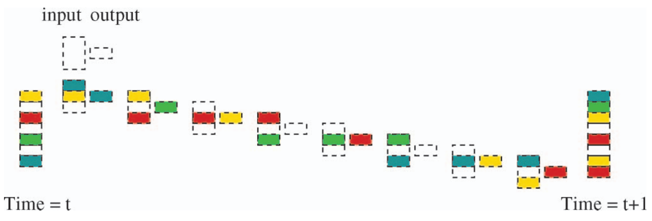


Fig. 2. Successive building steps between discrete time step t and $t+1$. Assume that the transition is carried out from top to bottom.

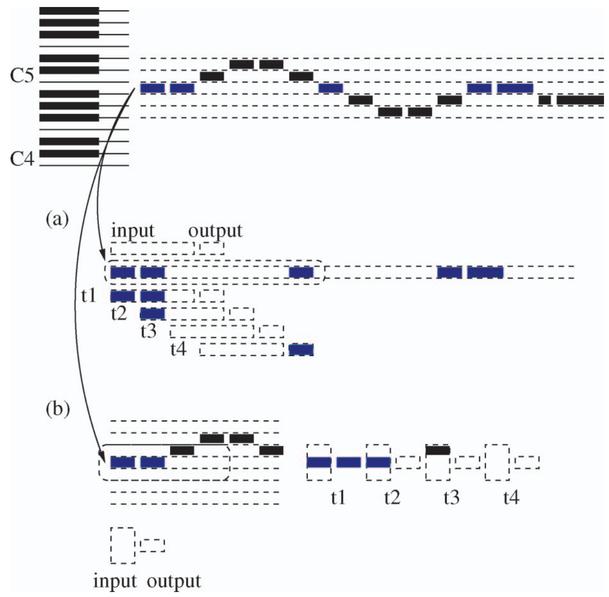


Fig. 3. The first 15 notes from the *Ode to Joy* theme and examples of two encoding styles that emphasize (a) melodic context and (b) harmonic context.

4.1 Learning CA rewrite rules using neural networks

Representing music as a piano roll is intuitive. Midi note-on and note-off events are displayed from top to bottom (according to their pitches) and from left to right as they are unfolded in time. Patterns from this piano roll could be learned using various machine learning techniques. We hypothesize that an artificial neural network could be used to learn CA's rewrite rules as shown in Figure 3(b). To prove the feasibility of this idea, multilayer perceptrons (MLPs) were constructed to learn CA patterns generated from known rules.[1] MLPs were structured in layers of input layer, hidden layer and output layer. The number of layers and nodes in each layer was problem dependent. In brief, the MLP was a trainable differential function where the training was through back-propagating errors from mismatch between the actual output and the expected output to adjust the weight connection among the nodes.

$$E = \sum_n (d_n - y_n)^2, \tag{3}$$

where d_n was the expected output from the nth-training sample and y_n was the actual output from the nth-training sample. The error is backpropagated to adjust the weight connection between each pair of neurons i and j in a small step α (known as learning rate):

$$w_{ij} = w_{ij} + \alpha \frac{\partial E}{\partial w_{ij}}. \tag{4}$$

The derivative of error between each pair of neurons i and j at the output layer was

$$\frac{\partial E}{\partial w^o_{ij}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial w^o_{ij}}, \tag{5}$$

where y_i was the output of the output node i. The derivative of error for between each pair of neurons i and j in a hidden layer was

$$\frac{\partial E}{\partial w^h_{ij}} = \sum_k \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial h_i} \frac{\partial h_i}{\partial w^h_{ij}}, \tag{6}$$

where y_k was the output of the output node k and h_i was the output of a hidden node i.

Figures 4 illustrates that MLPs could successfully learn rules used to generate CA patterns provided that the number of input nodes to an MLP was equal or more than the input size of the rewrite rule. The size of the input to MLP and the size of the rewrite rule were equal to three (3) in this example. CA patterns were generated from arbitrary chosen rules. Different MLPs were trained with each CA pattern. It was found that MLPs could successfully learn the hidden rules from the training patterns.

Theoretically speaking, the number of pitches in the input set (i.e. three binary input in the example in Figures 3 and 4) determines the number of plausible patterns, i.e. there are 2^n patterns for the binary input of length n. Unfortunately, due to the complexity of the music domain, ambiguity always exists even when the whole piano roll (88 notes) was used as an input. In other words, 2^{88} patterns of the whole input piano roll did not have enough prediction power. To include more context into the model, we employed two tactics (i) employing heterogeneous CA and (ii) exploiting temporal context by using time-delay neural networks. Heterogeneous CA allowed different pitches to have different sets of rules (promoting local context in the model) while time-delay introduced historical context into the model.

4.1.1 Employing heterogeneous CA

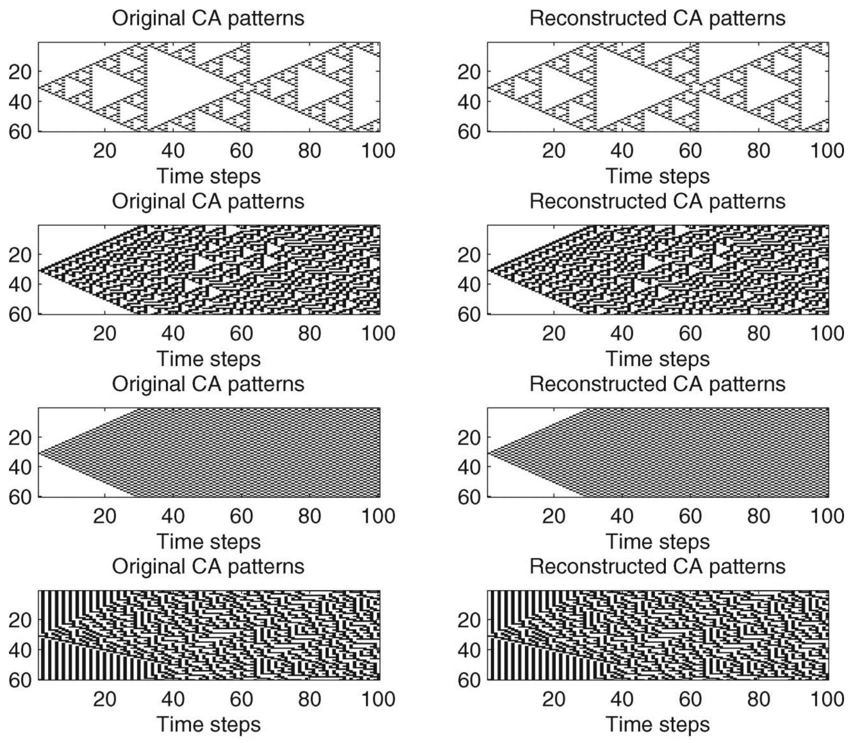
In this experiment, we decided to represent the input/output of the neural network in the style of Figures 3(b). Pitches between C2 to B6 (five octaves or 60 pitches) were allowed in this experiment. The duration of each input note was fixed at a quaver. For each pitch in the piano roll (e.g. C2), the rewrite rules were learned independently (i.e. each pitch had its own model). Hence, this setup realized a heterogeneous CA.

4.1.2 Employing temporal contexts

MLPs have one major drawback in that they cannot handle temporal dependency. Many researchers dealt with the time dependency using time delay input, the feedback from the output or the combination of delay input and output feedback. For example, the CONCERT system (Mozer, 1999) exploited historical events $n_{t-n}, \ldots, n_t$ to predict the next note n_{t+1}. Similar concepts were also being explored earlier by Conklin and Witten (1995), Todd (1989), Chen and Miikkulainen (2001) and recently by Oliwa and Wagner (2008). Adding temporal contexts into the model via delayed input or recurrent input helps improve the performance of the model. Here we report our experimental results from the *time-delay neural network* (*TDNN*) which was a delayed input neural network (Lang & Hinton, 1988).

Figure 5 shows the topology of a multilayer feedforward network and a time-delay network. Let $\mathbf{u}(t)$ be a vector where its element $u_i \in \{0, 1\}$ and $y(t) \in R$ be the input and output at time t, the relationship between

[1] MLPs and TDNNs are closely related, a TDNN can be seen as a buffered input MLP.



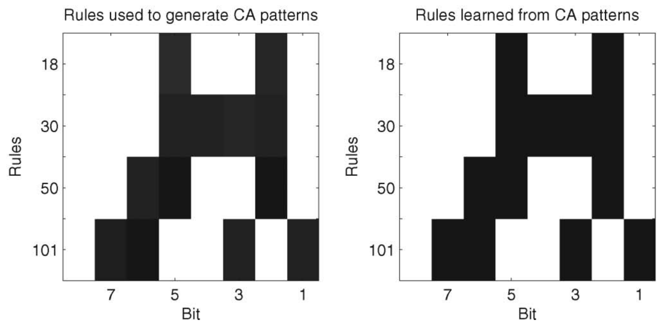


Fig. 4. Top four rows: original CA generated by different rules: 18, 30, 50, and 101 (left) and the reconstructed CA generated by learned MLP models (right); bottom: original rules used for generating the patterns (left) and learned rules (right), black cells represent 1 and white cells represent 0. From top down, 00010010, 00011110, 00110010, and 01100101. Note that 00010010 was named rule 18 (please refer to Figure 1 again).

output y and input $\mathbf{u}$ of the MLP and the TDNN can be expressed using Equations (7) and (8):

$$y(t) = f(u(t)), \tag{7}$$

$$y(t) = f(u(t), u(t-1), u(t-2), \ldots, u(t-m)). \tag{8}$$

4.2 Overview of the system

Combining both the neighbourhood size and time-delay enabled the model to exploit both the harmonic and melodic contexts (please refer to Figure 3 again). Therefore, each TDNN model could learn CA rewrite-rules that captured the local melodic, harmonic and textual contexts of a given piece. Figure 6 summarizes the

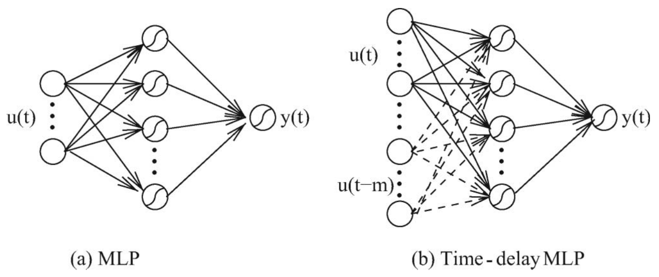


Fig. 5. (a) Network architecture of an MLP and (b) a time-delay MLP. Here, the input signal $\mathbf{u}(t-m), \ldots, \mathbf{u}(t-1), \mathbf{u}(t)$ was buffered before being fed to the TDDN. The time-delay MLP exploited the delayed input while the standard MLP was static and did not exploit any temporal context.

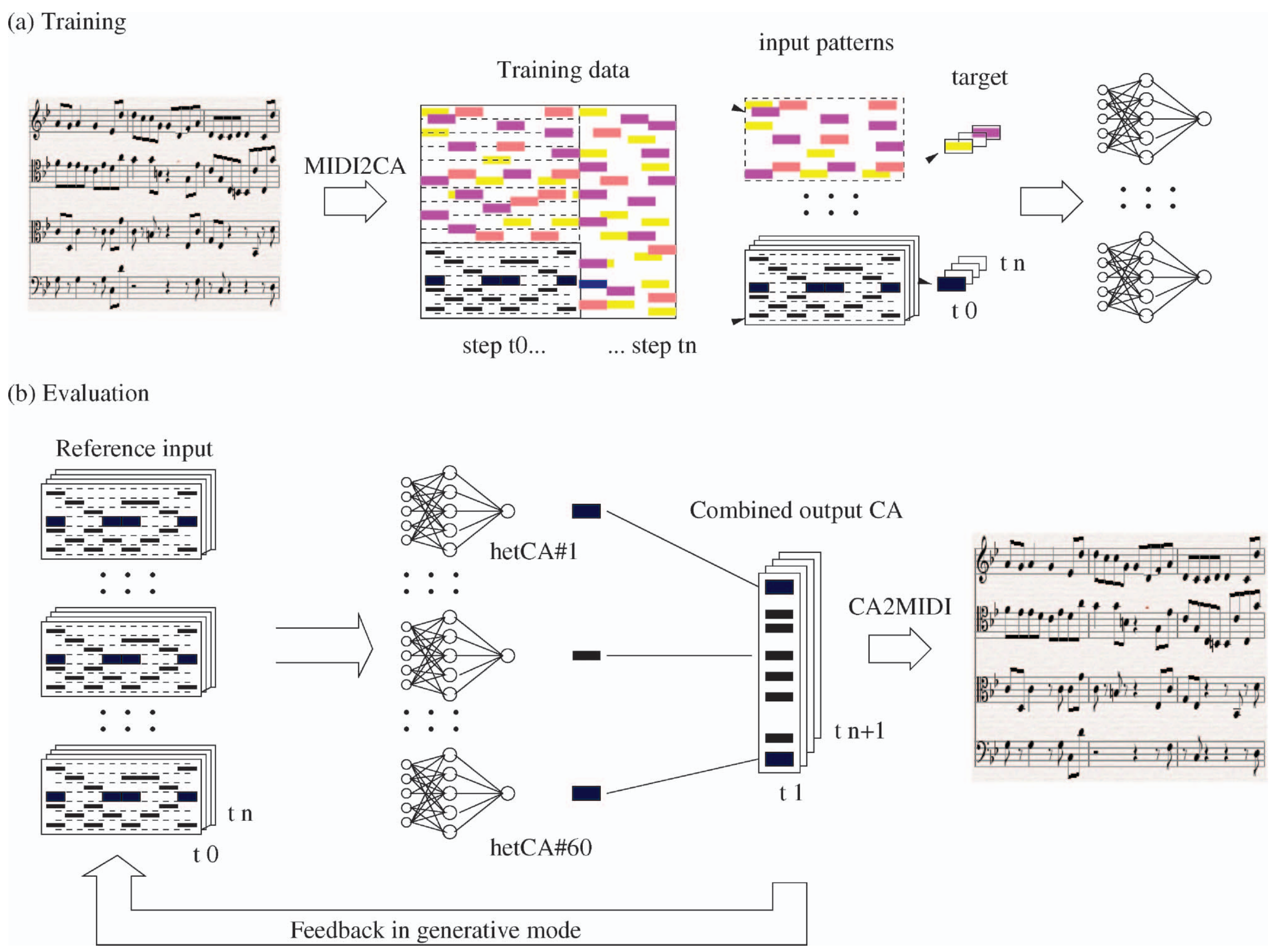


Fig. 6. An overview of the system, (a) a MIDI file was converted to CA. The data were segmented into input-target output pairs and were used to train the TDNNs; (b) the trained TDDNs (hetCAs) were employed to (i) evaluate the recall and precision rates of the model, and (ii) evaluate the generative power of the model.

whole process of the system. There were two phases, training and evaluation. In the training phase, the input MIDI file was converted into 2D-cellular automata.[2] The 2D-cellular automata were then segmented into an input–output sequence to train 60 TDNNs. After successful training, 60 heterogeneous CA would be available for evaluation.

There were two main evaluation criteria: (i) evaluation of the recall and precision rates of the models, and (ii) evaluation of the generative power of the models. The recall and precision rates were objectively evaluated using standard *recall* and *precision* measurements. The original input was fed to the learned models while all predicted outputs were collected and assembled to an output CA. The measurements *TP*, *TN*, *FP* and *FN* were calculated by comparing the generated notes with the original notes. TP denoted the count of generated correct notes, FP denoted the count of extra generated notes and FN was the count of rests where there should have been notes. All notes were counted at the resolution of a quaver note. This meant one wrong crotchet would be counted as two wrong units. The reason behind this was that our time step had the duration of a quaver note.

In evaluating the generative power of the model, one bar of chorale was used as a seed. Then generated output at time t were fed back to form a new input at time $t+1$. Figure 7 details the generative process. To generate a new piece, two or more input pieces were given to the system. One or more input pieces (CA_A) could be used to train the hetCA models. The model learned musical signatures from the training examples. After successful learning, the model was used to generate a new piece using some structural information extracted from a guided CA (CA_B).

[2] We write a small routine to facilitate the tranformation between MIDI to cellular automata (*MIDI2CA*) and from cellular automata back to MIDI (*CA2MIDI*). The *MIDI2CA* routine takes an input file in MIDI format and returns a matrix $\mathbf{P}_{ij}$ while the *CA2MIDI* routine takes the matrix $\mathbf{P}_{ij}$ and returns a MIDI file.

```
function $generate\ (CA_A, CA_B)$ return $CA_O$
    // $CA_A$ is an input CA used as a training example
    // $CA_B$ is an input CA used to guide the generation process
    // $CA_O$ is a generated CA
    $hetCA \leftarrow$ buildmodel($CA_A$)
    [$positions$ ] $\leftarrow$ calStructure($CA_B$)
    for each step $\in$ [$positions$]
        if it is the first bar or it is the cadence point
            then use patterns derived from $CA_B$ to generate new notes,   $n \leftarrow hetCA\ (CA_B)$
            else use patterns derived from $CA_O$ to generate new notes,   $n \leftarrow hetCA\ (CA_O)$
        endif
        $CA_O \leftarrow$ update($CA_O$,$n$)
    endeach
end
```

Fig. 7. A function used to generate a new musical formation pattern.

At the current stage, the musical context at the start and the cadence of each phrase were employed to guide the generative process. This was obtained by feeding the input from the guided piece to the model at the structural points (instead of using the feedback from the output).

5. Experiments and results

We selected seven pieces from *Riemenschneider 371 harmonized chorales and 69 chorale melodies* (Riemenschneider, 1941) for evaluation. The fact that music is a subjective topic and its evaluation involves both subjective and objective dimensions, has led us to devise two experiments to verify these two dimensions. For the first dimension, the precision and recall rate of the hetCA models were studied. The first experiment aimed to verify that the model could successfully learn the signature of a given piece. For the second dimension, the generative properties of the CA models were studied to investigate whether the models could be used to generate new pieces.

The performance of the first dimension could be objectively evaluated. However, the performance evaluation of the second objective could only be done objectively up to the syntax level (i.e. whether the generated composition had violated the normal part writing practice, but the quality of the composition was a subjective matter and would not be addressed in the present study).

5.1 Experimental designs

5.1.1 Preparing training input for TDNNs

All the seven chorales used in our experiment were downloaded from http://www.jsbchorales.net. They were originally in MIDI format. Each chorale was transformed into a piano roll representation where each time step represented a duration equivalent to a quaver note (sixteenth note). This piano roll could be seen as a one-dimension CA unfolding in time.

A training data set was prepared by constructing an input–output pair as described in Equation (1). Each chorale was partitioned into $60 \times T$ input–output sequences where T was the number of quaver time steps in the chorale. For example, sixteen bars of chorale music in 44 time would have 128 time steps ($T = 128$). All the experiments reported here were carried out with the following parameters: neighbourhood size of CA was 7 (e.g. $r = 3$); time-delay was 7 steps, hence, the TDNN network had 56 input nodes (7×8). The size of the hidden nodes was empirically determined and we had decided to use 64 hidden nodes in all the experiments; the transfer functions of the hidden nodes were tan-sigmoid transfer functions and the transfer function of the output node was a log-sigmoid transfer function (see Figure 6). From our choice of network architecture, the input signal (either from original piano roll or from feedback output in the generative mode) to the TDNN must be normalized to $[-1\ 1]$.

5.1.2 Converting the output from CA to music scores

The conversion of the output from CA to music scores must deal with two issues: extra voices and the combination of quavers having the same pitch. Looking at the CA formation output, one would observe that the generated patterns must be translated into four voices (soprano, alto, tenor and bass). However, sometimes, the voices could be more than or less than four voices. Also, if two adjacent quavers have the same pitch, should they be combined?

In this implementation, we used the following heuristics to resolve these issues: (i) voices were always read from soprano downward; and (ii) in each bar, if two or more quaver notes could form a crotchet on the beat then they were combined. The heuristics were simple and had worked well in our case, although they were not perfect since (i) the real bass could be missed out if CA patterns showed more than four voices; and (ii) a single minim might be interpreted as two crotchets.

Table 1. Evaluate precision and recall performances.

	Chorale	TP	FP	FN	Recall	Precision	f
	Melody (Soprano)						
R100:	*Durch Adams Fall ist ganz verderbt*	125	2	11	91.91	98.43	0.95
R182:	*Wär Gott nicht mit uns diese Zeit*	111	2	1	99.11	98.23	0.98
R266:	*Herr Jesu Christ, du höchstes Gut*	115	3	5	95.83	97.46	0.96
R321:	*Wir Christenleut'*	85	1	3	96.59	98.84	0.97
	Harmony (SATB)						
R26:	*O Ewigkeit, du Donnerwort*	492	2	20	96.09	99.60	0.98
R100:	*Durch Adams Fall ist ganz verderbt*	497	2	47	91.36	99.60	0.95
R182:	*Wär Gott nicht mit uns diese Zeit*	436	0	12	97.32	100.00	0.99
R266:	*Herr Jesu Christ, du höchstes Gut*	461	2	19	96.04	99.57	0.98
R279:	*Ach Gott und Herr*	314	1	6	98.12	99.68	0.99
R321:	*Wir Christenleut'*	339	0	13	96.31	100.00	0.98
R355:	*Nun ruhen alle Walder*	361	0	23	94.01	100.00	0.97

5.1.3 Measurement metrics

The performance of hetCA learned by TDNN was objectively evaluated using *recall* and *precision* measurements as defined below:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative},$$
$$Precision = \frac{TruePositive}{TruePositive + FalseNegative},$$

$$f = \frac{2 \times recall \times precision}{recall + precision}.$$

5.2 Precision and recall rate of hetCA

Seven Bach Chorales (Riemenschneider ID R26, R100, R182, R266, R279, R321 and R355) were chosen for this experiment. Eleven hetCA models were built, four models for four soprano melodies and the other seven models for chorale harmonization. The top block of Table 1 shows the precision and recall rates of the soprano melody alone while the bottom block shows the precision and recall rates of the chorale harmonizations (i.e. soprano, alto, tenor and bass-SATB). The recall and precision were quite good for both melody and harmony recall tasks. To the best of our knowledge, this approach was novel and we could not directly compare the recall and precision rates of this work to other related works (interested readers could carry out the same experiment by obtaining the same MIDI files from http://www.jsbchorales.net).

Figure 8 illustrates the behaviours of the hetCA models in a melody recall task. The first row was a piano roll of the original soprano line. The second row was the recalled soprano line (i.e. the system predicted notes at time t +1 from given input at time t). The third and the fourth rows were *FP* and *FN*. Each white dash unit was counted as one FP (see 3rd row) and each black dash unit was counted as one FN (see 4th row).[3] From the figures, *FP* and *FN* were calculated from the differences between the original chorales and the reconstructed chorales: $FP = max(Original - Reconstructed,\ 0)$ and $FN = max(Reconstructed - Original,\ 0)$ where $max(a,\ b)$ returned the a if $a \geq b$, otherwise returned b .

Figures 9 and 10 illustrate the behaviours of hetCA models in a recall task. The first row was a piano roll of the original chorale. The second row was the recalled chorale. The third and the fourth rows were *FP* and *FN* respectively. The results show that the models had successfully learned a Bach chorale harmonization pattern. The recalled harmonization was also transcribed into a standard score format for our readers' convenience (see Figures 11 to 13).

5.3 Generating music formations using CA

It has been demonstrated in the previous section that hetCAs could successfully learn and recall the trained musical pattern formations. In this section, it is of our interest to see whether hetCA models can be used in a generative mode.

Unlike in the recall mode where the original patterns were given to the model and the responses of the model were monitored, in the generative mode, a seed (i.e. a few CA time steps) was given to initialize the composition process. The generated output then became the input of the next time step (see Figure 5) and the composition was terminated when the desired time step was reached. In this experiment, the chorale *Herr Jesu Christ, du höchstes Gut* was chosen as a template where the first bar of the chorale was taken as a seed and two crotchet notes at

[3] Please take note that the white dash represented a note-on and a black dash represented a note-off in these figures. The colors were reversed to enhance clarity of these images.

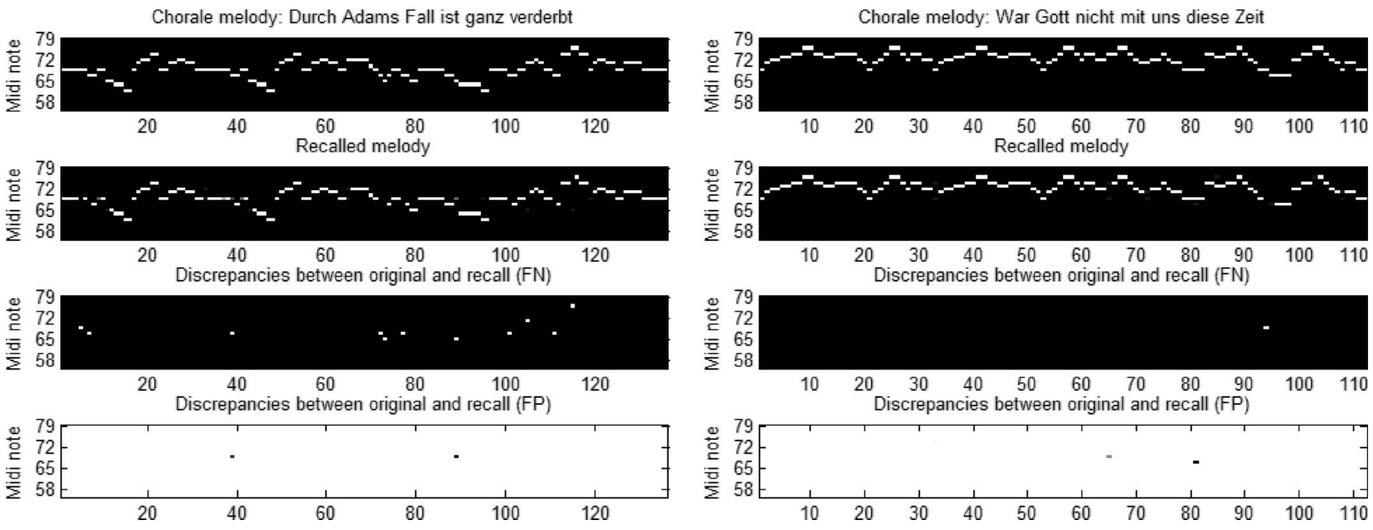


Fig. 8. From top down, a piano roll of the original chorale melody, recalled, TP and FP. On the left *Durch Adams Fall ist ganz verderbt* and on the right *Wär Gott nicht mit uns diese Zeit*.

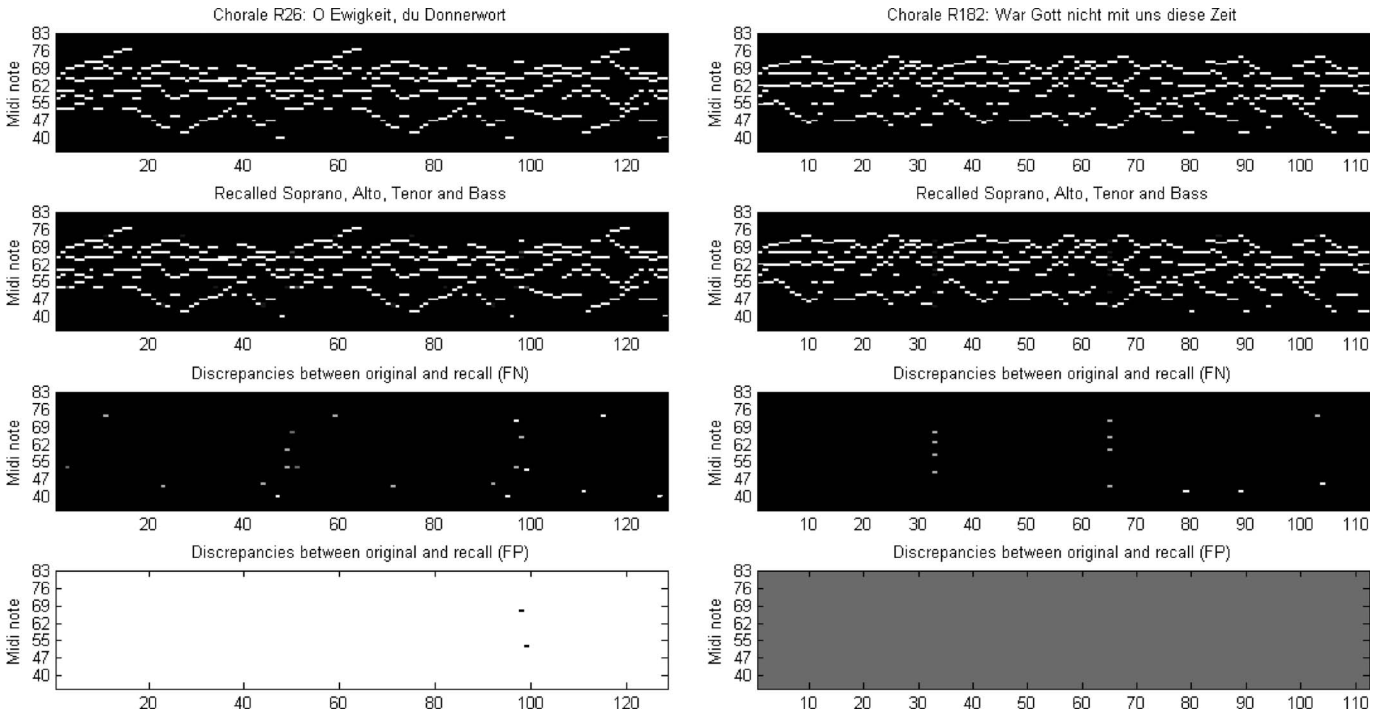


Fig. 9. From top down, a piano roll of the original chorale, recalled, TP and FP. On the left *O Ewigkeit, du Donnerwort* and on the right *Wär Gott nicht mit uns diese Zeit*.

each cadence point were also fed to the model as input patterns at cadences. The following experiments were carried out:

1. Generative model 1: the hetCA model was built from chorale A and the generation was guided by chorale A.
2. Generative model 2: the hetCA model was built from a set of chorale S and the generation was guided by chorale $A \in S$.
3. Generative model 3: the hetCA model was built from chorale A and the generation was guided by chorale B where $A <> B$.

5.3.1 Generative models

In the first generative model, the hetCA was trained with the input training samples prepared from the chorale *Herr Jesu Christ, du höchstes Gut*. The model was used to generate a new piece using the same piece as a template. The template provided a control over the generation

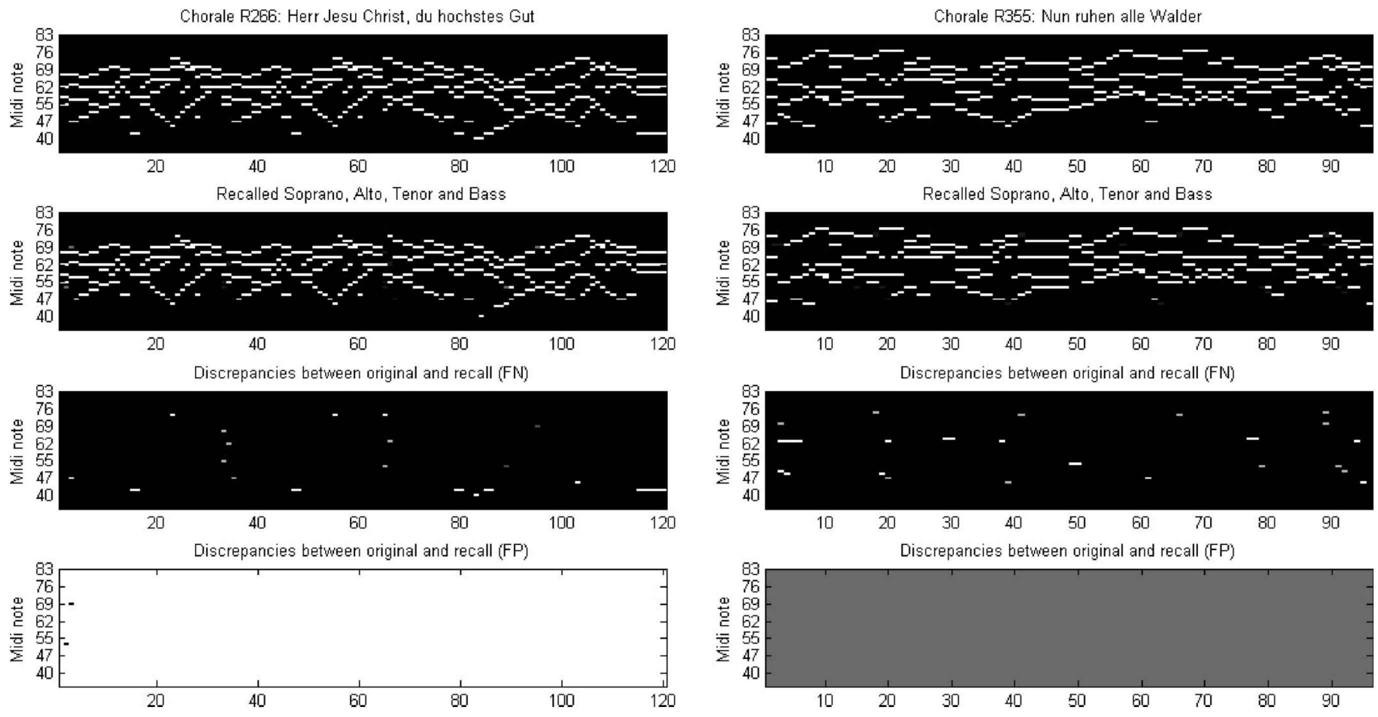


Fig. 10. From top down, a piano roll of the original chorale, recalled, TP and FP. On the left *Herr Jesu Christ, du höchstes Gut* and on the right *Nun ruhen alle Walder*.

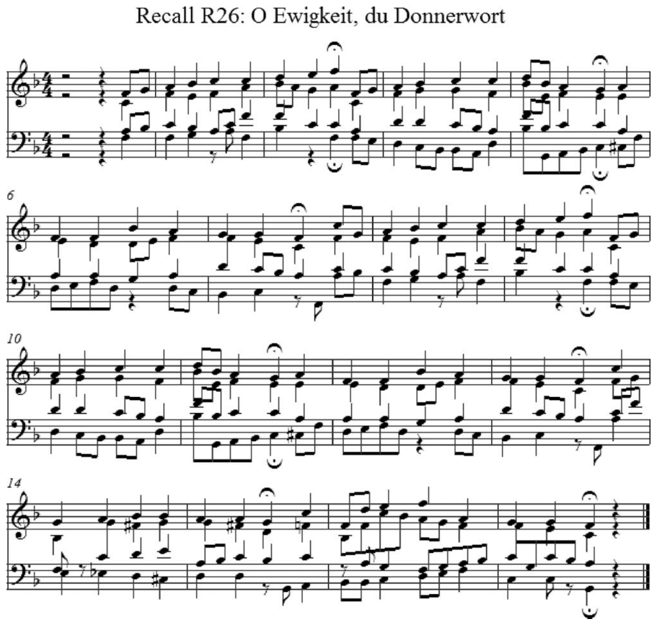


Fig. 11. Recall of a chorale *O Ewigkeit, du Donnerwort*.

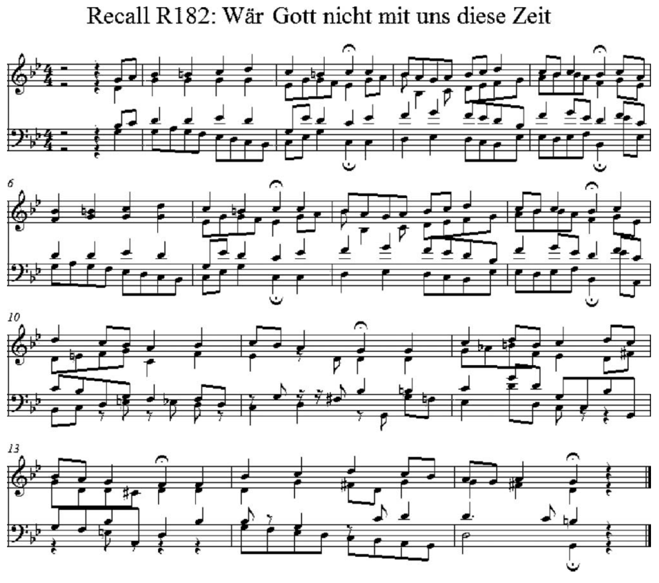


Fig. 12. Recall of a chorale *Wär Gott nicht mit uns diese Zeit*.

process by providing a predefined input (i.e. first bar and all cadences) to the hetCA from the template; otherwise, the input to hetCA at step *n* was the output from hetCA at step $n-1$. The output from this generative model is presented in a standard score in Figure 14.

In the second generative model, the hetCA was trained with the input training samples prepared from the chorale *Wär Gott nicht mit uns diese Zeit* and the chorale *Herr Jesu Christ, du höchstes Gut*. After training, the model was used to generate a new piece using the chorale *Herr Jesu Christ, du höchstes Gut* as a template. The output from this generative model is presented in a standard score in Figure 15.

In the third generative model, the hetCA was trained with the input training samples prepared from the chorale *Wär Gott nicht mit uns diese Zeit*. After training, the model was used to generate a new piece using the

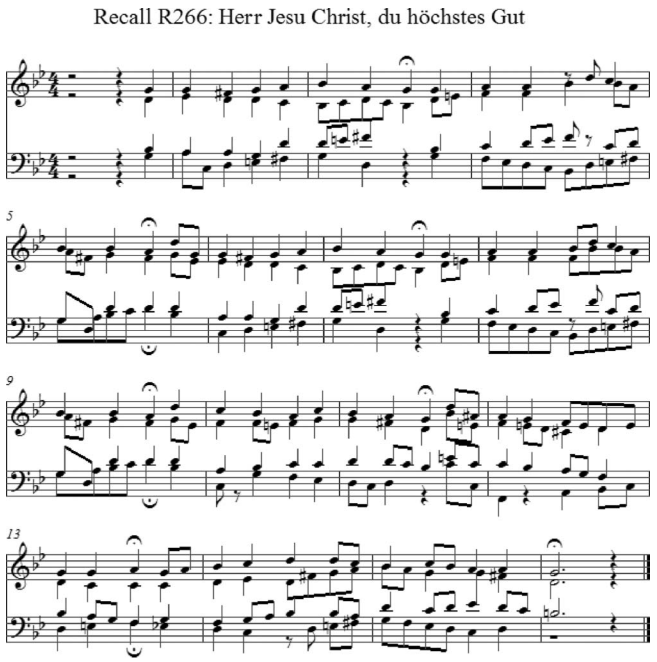


Fig. 13. Recall of a chorale *Herr Jesu Christ, du höchstes Gut*.

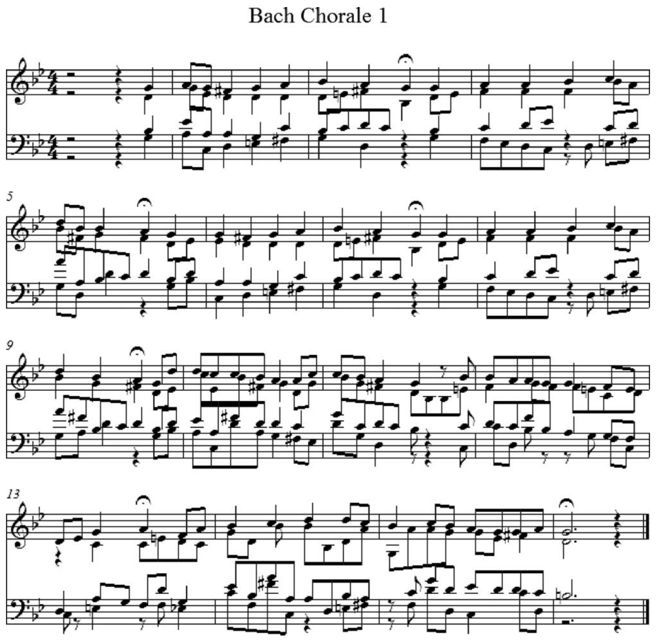


Fig. 14. Generating a chorale from a model learned from R266, using R266 as a template.

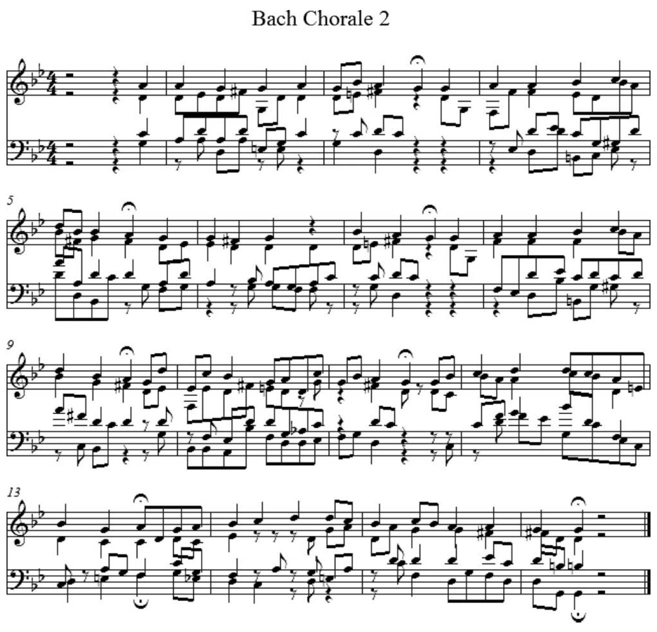


Fig. 15. Generating a chorale from a model learned from R182 and R266, using R266 as a template.

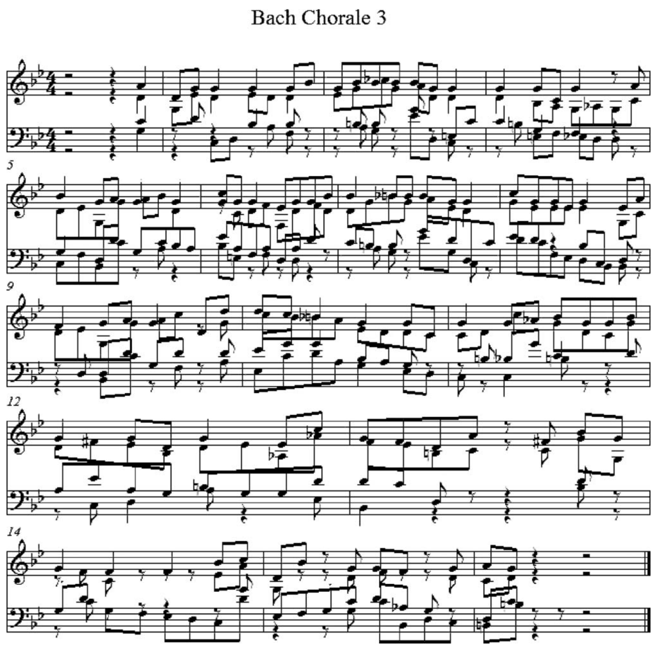


Fig. 16. Generating a chorale from a model learned from R182, using R266 as a template.

chorale *Herr Jesu Christ, du höchstes Gut* as a template. The output from this generative model is presented in a standard score in Figure 16.

6. Analysis and discussion

It is safe to assume from the experiment that the hetCAs were capable of learning patterns presented to the system and reproducing what it had learned effectively (i.e. evidence from high recall and precision rates). Figure 11 to 13 show the recalled output from the three models learned from R26, R182 and R266 respectively. The outputs were very good indeed. They recalled all the notes almost perfectly. Only some notes in the bass part were left out.

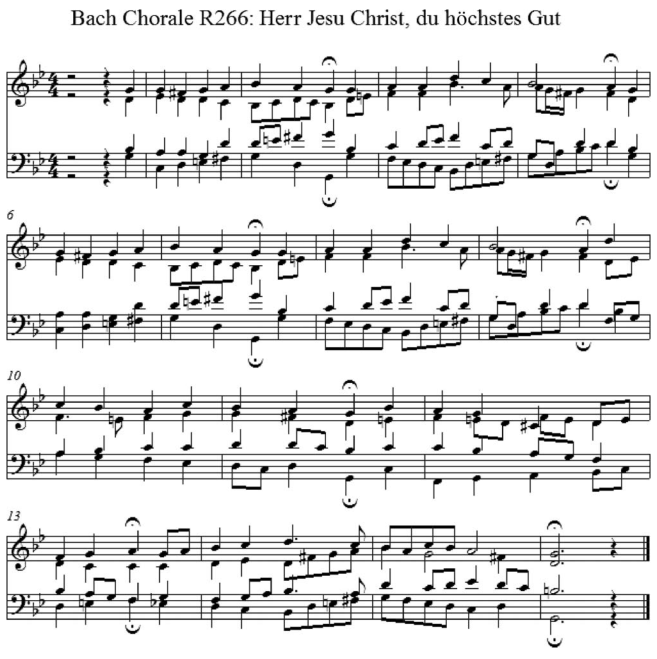


Fig. 17. Reproduction of the original chorale *Herr Jesu Christ, du höchstes Gut*.

The hetCA models could generate reasonable output with some guidance in the form of a template. In this report, we decided to choose the chorale *Herr Jesu Christ, du höchstes Gut* as a template. The original chorale (edited by Riemenschneider (1941)) is shown in Figure 17. A brief description of the structure of chorale R266 is given here. R266 had 16 bars, in the key of G-minor and with seven cadence points:

$$g : V_7 - i; i - v; V_7 - i; i - v; V_7 - i; F : V_7 - I; g : V_7 - I.$$

The first two cadences were in a perfect cadence, then an imperfect cadence, and they were repeated before moving to a brief modulation to the key of F major (observed in bar 13). The chorale came to a perfect cadence in the home key at the end. Three generated chorales based on the structure above are presented in Figure 14 to 16.

The three outputs from the generative models 1, 2 and 3 displayed similar characteristics to chorale R266. However, it was observed that the similarity was strongest in the output from generative model 1 and weakest in generative model 3. This could be explained in terms of the effects from the *memory capacity* of the model. The generative model could actually be seen as a lookup table. Hence, it was natural that the reproduction of patterns learned earlier should be more effective than the reproduction of patterns not in the lookup table. In other words, generative model 3 learned the patterns from R182, so we should not expect the model to generate a convincing chorale using the template from R266.

If the lookup table was built from the whole 371 chorales from Riemenschneider (1941), then would it be possible to generate the chorale number 372 in the style of Bach? Unfortunately, *the curse of context dependency* was still not solved in the proposed paradigm. This meant the model would be able to look up the answer for each query pattern. However, the solution would likely make sense in the local context and less sensible in the global context. This drawback is universal in most algorithmic composition techniques.

6.1 What was learned by TDNN?

From the results, it is evident that the patterns were successfully learned by the system since it performed well in associating the input–output patterns. However, the chorale output in the generative mode was less successful. So what was learned by TDNN?

Training TDNN to associate between the input and output was a memory task. All that the TDNN had learned were the associations between input–output patterns. Hence, the characteristics of music were implicitly learned in the model. That is, the model would not have explicit knowledge regarding harmonic vocabulary, voice leading, stylistic textures, etc. Generating new materials, therefore, should be less successful since the generation mechanism could only blindly exploit the associations between the input and output patterns.

Advocates of a rule-based system would point out that this is the major drawback of the approach since theoretical knowledge could not be exploited here. However, from another perspective, this is the strong point of soft computing techniques. Explicit capture of knowledge contents and explicit exploitation of them are in many cases less preferred (e.g. applications in speech recognition and face recognition are more successful with soft computing methods), since the domains are too complex to be modelled solely by rules.

How can we improve the situation? In our opinion, long-range dependency is the main culprit and we need to be able to learn patterns that span across time such as the patterns of harmonic progression as well as the patterns that are local such as voice leading. Logically, the longer the input patterns are, the longer the time span for the model to learn. However, this method has a limit in its scalability. Theoretically, plausible associations in the current implementation are $2^{56} = 7.2 \times 10^{16}$. This is huge and a dimension reduction technique would be needed to reduce the input size. Apart from expanding the input size, emphasis could be given to the input patterns. The emphasis may be from the non-nearest neighbourhood structure where influences from the nearest neighbourhood could be lessened. This could be used, for example, to emphasize the influence from the strong beat or from the opening phrase, etc.

7. Conclusion

Here, our goal was to let the system learn music domain knowledge from CA pattern formations. Information related to pitch, melody, and harmony were captured in CA rewrite rules (i.e. a sequence of seven cell input captured harmonic intervals up to seven semitones while seven delayed steps input captured melodic progression up to four crotchets). Increasing the time-delay and the length of the neighbouring cells will enrich the context but will also incur a lot of computing expenses.

The current work successfully shows that (i) it is possible to perform knowledge elicitation through machine learning techniques; (ii) hetCAs could be employed as generative models and (iii) with some extra knowledge in the form of a template, the current hetCA models could reproduce convincing chorale harmonizations as compared to previous works in this area (Hild et al., 1991; Ebcioglu, 1992; Allan & Williams, 2004; Phon-Amnuaisuk et al., 2006).

It is a real challenge to have a neural network learn the right music contexts from training patterns. A critic might believe that this could never be achieved with a weak system[4] such as CA. However, it is our belief that there are many interesting components in this study. One of them is the dependencies among cells in cellular automata. The standard CA exploit only local dependencies, that is, the context is only from the neighbourhood cells (see Equation (1)). In the experiments presented above, the historical context was also being exploited (see Equation (2)). This historical context captured the dependencies in different time steps. It is possible to have a longer and more complex context on computational expenses, though, this might not always be a fruitful approach.

We hope readers could agree with us that, theoretically speaking, it is possible to create a lookup table for all songs that have ever been composed so far. If such a lookup table exists, it could be imaginable that we would be able to predict the next note of any given sequence with absolute 100% recall rate, provided that we can access all of its contexts (i.e. no ambiguity). Can we turn this humongous table into a compositional system? We think it is possible provided that we could access the knowledge that describes the deeper structure of compositions in particular genres. Therefore the aim of model learning should be to capture important deeper melodic, harmonic, textual and formal contexts of training examples.

[4] The term *weak* is used here to signify the non-knowledge intensive approach.

In this report, we present a composition system that relied on cellular automata. Rewrite-rules of each hetCA was learned using a time-delay neural network. Examples of music pattern formations generated from heterogeneous CA were displayed. Evaluations of the proposed approach were carried out and the results show that CA could successfully learn complex patterns. In further work, we shall investigate the effectiveness of the approach in learning deep structural information (Marsden, 2005); we will also explore dynamic control of CA operations e.g. through dynamic rewrite rules, etc. Hybrid systems between other machine learning approaches and CA would be another interesting area to explore.

Acknowledgement

We wish to thank the anonymous reviewers for their useful comments and suggestions.

References

Allan, M., & Williams, C.K. (2004). Harmonising chorales by probabilistic inference. In L.K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in Neural Information Processing Systems 17* (pp. 25–40). Cambridge, MA: MIT Press.

Ames, C. (1993). Protocol: Motivation, design, and production of a composition for solo piano. In S.M. Schwanauer & D.A. Levitt (Eds.), *Machine Models of Music* (Chap. 16, pp. 357–382). Cambridge, MA: MIT Press.

Ames, C., & Domino, M. (1992). Cybernetic composer: An overview. In M. Balaban, K. Ebcioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (Chap. 8, pp. 186–205). Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press.

Ariza, C. (2007). Automata bending: Applications of dynamic mutation and dynamic rules in modular one-dimension cellular automata. *Computer Music Journal*, *31*(1), 29–49.

Assayag, G., & Dubnov, S. (2004). Using factor oracles for machine improvisation. *Soft Computing*, *8*(9), 604–610.

Baroni, M., Brunetti, R., Callegari, L., & Jacobni, C. (1982). A grammar for melody: Relationship between melody and harmony. In M. Baroni & L. Callegari (Eds.), *Musical Grammar and Computer Analysis* (pp. 201–218). Florence: Olschki.

Beyls, P. (1989). The musical universe of cellular automata. In T. Wells & D. Butler (Eds.), *Proceedings of the International Computer Music Conference (ICMA 1989)* (pp. 33–41). San Francisco: International Computer Music Association.

Brook Jr., F.P., Hopkins Jr., A.L., Neumann, P.G., & Wright, W.V. (1993). An experiment in musical composition. In S.M. Schwanauer & D.A. Levitt (Eds.), *Machine Models of Music* (Chap. 2, pp. 23–40). Cambridge, MA: MIT Press.

Brown, A.R. (2005). Exploring rhythmic automata. In *Proceedings of EvoWorkshops: Applications of Evolutionary Computing* (LNCS 3449, Lecture Notes in Computer Science, pp. 551–556). Berlin: Springer.

Burraston, D., & Edmons, E. (2005). Cellular automata in generative electronic music and sonic art: Historical and technical review. *Digital Creativity*, *16*(3), 165–185.

Chen, C.C.J., & Miikkulainen, R. (2001). Creating melodies with evolving recurrent neural networks. In *Proceedings of the 2001 International Joint Conference on Neural Network, IJCNN-01* (pp. 2241–2246). Washington DC: IEEE.

Conklin, D., & Witten, I. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, *24*, 51–73.

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

Cope, D. (2001). *Virtual Music: Computer Synthesis of Musical Style*. Oxford: Oxford University Press, 2001.

Dubnov, S., & Assayag, G. (2005). Improvisation planning and jam session design using concepts of sequence variation and flow experience. In *Proceedings of International Conference: Sound and Music Computing SMC'05, Salerno, Italy, 26–26 November* (pp. 49–56). Paris: IRCAM.

Ebcioglu, K. (1992). An expert system for harmonizing four-part chorales. In M. Balaban, K. Ebcioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (Chap. 12, pp. 294–333). Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press.

Fry, C. (1980). Computer improvisation. *Computer Music Journal*, *4*(3), 48–58.

Hild, H., Feulner, J., & Menzel, W. (1991). Harmonet: A neural net for harmonizing chorales in the style of J.S. Bach. In R.P. Lippmann, J.E. Moody, & D.S. Touretzky (Eds.), *Advances in Neural Information Processing 4* (pp. 267–274). New York: Morgan Kaufman.

Hiller, L. (1970). Music composed with computers: A historical survey. In H.B. Lincoln (Ed.), *The Computer and Music* (Chap. 4, pp. 42–96). Ithaca, NY: Cornell University Press.

Lang, K.J., & Hinton, G.F. (1988). *The development of the time-delay neural network architecture for speech recognition* (Technical report cmu-cs-88-152). Pittsburgh, PA: Carnegie-Mellon University.

Law, E.H.H., & Phon-Amnuaisuk, S. (2008). Towards music fitness evaluation with the hierarchical SOM. In *Proceedings of EvoWorkshops: Applications of Evolutionary Computing* (LNCS 4974, Lecture Notes in Computer Science, pp. 443–452). Berlin: Springer.

Lerdahl, F., & Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. Cambridge, MA: MIT Press.

Marsden, A. (2005). Generative structural representation of tonal music. *Journal of New Music Research*, *34*(4), 409–428.

McAlpine, K., Miranda, E., & Hoggar, S. (1999). Making music with algorithm: A case study. *Computer Music Journal*, *23*(2), 19–30.

Millen, D. (2004). An interactive cellular automata music application in COCOA. In *Proceedings of the International Computer Music Conference (ICMC 2004)* (pp. 51–54). San Francisco: International Computer Music Association.

Miranda, E.R. (1993). Cellular automata music: An interdisciplinary project. *Interface*, *22*(1), 3–21.

Mozer, M.C. (1999). Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. In N. Griffith & P.M. Todd (Eds.), *Musical Networks: Parallel Distributed Perception and Performance* (pp. 227–260). Cambridge, MA: MIT Press.

Oliwa, T., & Wagner, M. (2008). Composing music with neural networks and probabilistic finite-state machines. In *Proceedings of EvoWorkshops: Applications of Evolutionary Computing* (LNCS 4974, Lecture Notes in Computer Science, pp. 503–508). Berlin: Springer.

Phon-Amnuaisuk, S., Smaill, A., & Wiggins, G. (2006). Chorale harmonisation: A view from a search control perspective. *Journal of New Music Research*, *35*(4), 279–305.

Riemenschneider, A. (1941). *371 Harmonized Chorales and 69 Chorale Melodies with Figured Bass*. New York: G. Schirmer, Inc.

Smaill, A., Wiggins, G., & Miranda, E. (1993). Music representation: Between the musician and the computer. In M. Smith, G. Wiggins, & A. Smaill (Eds.), *Music Education: An Artificial Intelligence Approach; Proceedings of a Workshop Held as Part of AI-ED 93, World Conference on AI in Education, Edinburgh* (pp. 108–119). Berlin: Springer-Verlag. (Also Research Paper 668, Department of Artificial Intelligence, University of Edinburgh, UK).

Steedman, M. (1984). A generative grammar for jazz chord sequences. *Music Perception*, *2*, 52–77.

Temperley, D. (2007). *Music and Probability*. Cambridge, MA: MIT Press.

Todd, P.M. (1989). A connectionist approach to algorithmic composition. *Computer Music Journal*, *13*(4), 27–43.

von Neumann, J. (1966). *Theory of Self-reproduction Automata*. Urbana: University of Illinois Press.

West, R., Howell, P., & Cross, I. (1991). Musical structure and knowledge representation. In P. Howell, R. West, & I. Cross (Eds.), *Representing Musical Structure* (Chap. 1, pp. 1–30). New York: Academic Press.

Wolfram, S. (2002). *A New Kind of Science*. Champaign, IL: Wolfram Media.

Wolframtones. (2009). *WolframTones*. http://tones.wolfram.com

Xenakis, I. (1992). *Formalized Music: Thought and Mathematics in Composition* (Rev. ed.). Hillsdale, NY: Pendragon Press.