Monograph:
Yang, Y.X. and Billings, S.A. (1999) Neighbourhood Detection and Rule Selection From Cellular Automata Patterns. Research Report. ACSE Research Report 742 . Department of Automatic Control and Systems Engineering

Neighbourhood Detection and Rule Selection from Cellular Automata Patterns

Y.X.Yang S.A.Billings

Department of Automatic Control and System Engineering
University of Sheffield
Mappin Street, Sheffield S1 3JD
United Kingdom

Research Report No.742

22 March 1999

University of Sheffield

Neighbourhood Detection and Rule Selection from Cellular Automata Patterns

Y.X.Yang S.A.Billings

Dept. of Automatic Control and System Engineering

Univ. of Sheffield, Mappin Street, Sheffield S1 3JD, UK

Abstract

Using GA's to search for CA rules from spatio-temporal patterns produced in CA evolution is usually complicated and time-consuming when both the neighbourhood structure and the local rule are searched simultaneously. The complexity of this problem motivates the development of a new search which separates the neighbourhood detection from the GA search. In this paper, the neighbourhood is determined by independently selecting certain terms from a large term set on the basis of the contribution each term makes to the next state of the cell to be updated. The GA search is then started with a considerably smaller set of candidate rules pre-defined by the detected neighbourhood. This approach is tested over a large set of 1-D and 2-D CA rules. Simulation results illustrate the efficiency of the new algorithm.

1 Introduction

A cellular automaton (CA) is a discrete system which evolves in discrete time over a lattice structure composed of a large quantity of cells. The states of the cells are discrete and are updated synchronously according to a local rule operating on a given neighbourhood. The study of CA's has been the focus of attention from a wide range of researchers [1]-[6]. One of the most important topics in CA studies is the identification of the CA, that is, to extract the neighbourhood and the governing local rule from a given set of spatio-temporal patterns produced by the CA evolution.

Ideally the identification technique should be designed to produce an optimal CA expression which consists of a clear, minimal neighbourhood structure and a correct local rule. In [7], although correct rules were generated by applying a set of sequential and parallel algorithms, the associated neighbourhood was not clearly presented and the identification process was complicated and time-consuming. GA's were employed in [8] in search for a matching rule from a large rule set of all possible rules. Again no satisfactory neighbourhood structure was obtained. In [9] the minimal neighbourhood problem was addressed by introducing a second search objective to the GA's on the basis of reformulating the CA rules into a uniform Boolean expression. Because the assumed neighbourhood which determines the run time is usually much larger than the actual neighbourhood, the search process can be very long, sometimes taking several hours for a single run, even for a very simple 1-D CA rule (see Table 6 in [9]). However it might be possible to substantially reduce the run time if the assumed neighbourhood for the GA search was correct and minimal. One way to achieve this would be to determine the neighbourhood before starting the rule search and this is the main objective of the present study.

In this paper a new neighbourhood detection technique is introduced which is capable of extracting the correct and minimal neighbourhood from a large set of candidate neighbourhoods without having to define the rule at the same time. The algorithm is simple and easy to implement and forms the first stage in a CA identification procedure. A simple GA is then employed, starting with the obtained neighbourhood, to search for the best matching local rule. The remainder of the paper is structured as follows. In Section 2.1, the notation and background to 1-D and 2-D CA neighbourhoods and rules is introduced. The new neighbourhood detection technique is then developed in Section 2.2 based on the Boolean expressions of the CA rules. Section 3 describes the GA search for the correct local rule using the obtained neighbourhood. Simulation studies are presented in Section 4, and Section 5 discusses the efficiency of the algorithm.

2 Neighbourhood detection

2.1 Cellular Automata

A cellular automaton is composed of three parts: a discrete lattice, a neighbourhood and a rule for local transitions. All cells in the lattice are updated synchronously according to the local rule.

2.1.1 Neighbourhoods

The neighbourhood of a cell is the group of the cells which are able to directly affect the evolution. Some of the most frequently used neighbourhoods are illustrated in Figure 1. For simplicity, this paper only considers neighbourhoods composed of cells from time step $t-1$, but the results are not restricted to this case.

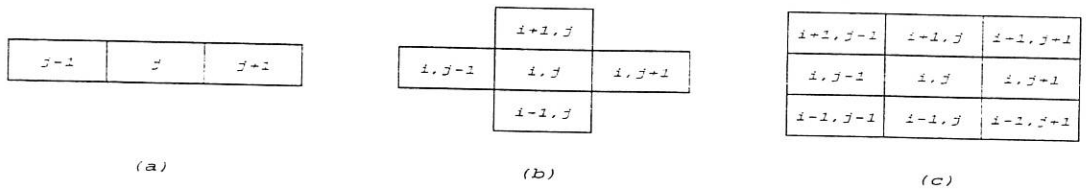


Figure 1. Examples of some of the most frequently used 1-D and 2-D neighbourhoods

(a) the 1-D von Neumann neighbourhood (b) the 2-D von Neumann neighbourhood (c) the 2-D Moore neighbourhood

2.1.2 Boolean form of CA rules

There are various representations for a CA rule. In this paper Boolean expressions will be considered. From [9], every CA rule with an n site neighbourhood $\{cell(x_1), \cdots, cell(x_n)\}$ can be written as

$$s_{new}(x_j) = a_0 \oplus a_1 s(x_1) \oplus \cdots \oplus a_P(s(x_1) * \cdots * s(x_n)) \qquad (1)$$

where $P = 2^n - 1$. x_j is the cell to be updated. $s(x_i)$ is the state of $cell(x_i)$ at time step $t-1$. $s_{new}(x_j)$ is the next state in $cell(x_j)$. a_i $(i = 0, \cdots, P)$ are binary numbers and $a_i = 1$ indicates that the following term is included in the Boolean expression while $a_i = 0$ indicates that the following term is not included. $\oplus$ and * represent XOR and AND operators respectively.

2.2 Neighbourhood detection

Define the vector XOR operator $\hat{\oplus}$ as

$$\begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} \hat{\oplus} \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_n \end{bmatrix} = (b_1 * c_1) \oplus (b_2 * c_2) \oplus \cdots \oplus (b_n * c_n)$$

where b_i and c_i $(i = 1, \cdots, n)$ are binary numbers. Equation (1) can then be represented as

$$s_{new}(x_j) = \mathbf{s} \hat{\oplus} \mathbf{a} \tag{2}$$

where

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_P \end{bmatrix}^T$$

and

$$\mathbf{s} = \begin{bmatrix} 1 & s(x_1) & \cdots & s(x_n) & s(x_1) \times s(x_2) & \cdots & s(x_1) \times \cdots \times s(x_n) \end{bmatrix}$$

Applying $b_1 * b_2 = b_1 \times b_2$, $b_1 \oplus b_2 = b_1 - b_2 - 2 \times b_1 \times b_2$, and $b^m = b$ (where b_1, b_2, b are binary numbers and m is a positive integer) to equation (2) yields

$$s_{new}(x_j) = \mathbf{s} \times \bar{\mathbf{a}} \tag{3}$$

where $\bar{\mathbf{a}}$ is a $P \times 1$ integer vector.

One way to detect actual cells in the assumed neighbourhood $\{cell(x_1), \cdots, cell(x_n)\}$ of $cell(x_j)$ is to calculate the contribution each cell makes to $s_{new}(x_j)$. Alternatively, since there is no direct way to evaluate the performance of each cell, terms in $\mathbf{s}$, such as $s(x_1)$, $s(x_n)$, $s(x_1) \times s(x_2)$, $s(x_1) \times \cdots \times s(x_n)$, which are formed from various combinations of all the possible cells, can be exploited for this purpose. However, the effect each term has on $s_{new}(x_j)$ is entangled in $\mathbf{s}$, it is therefore not easy to assess the individual performance directly from equation (3). The new procedure below is therefore introduced to overcome this problem and to calculate each contribution independently.

Equation (3) can be written as

$$\mathbf{s}_{new} = \mathbf{S} \times \bar{\mathbf{a}} \tag{4}$$

where

$$\mathbf{s}_{new} = \begin{bmatrix} s_{new}(x_j(1)) & s_{new}(x_j(2)) & \cdots & s_{new}(x_j(N)) \end{bmatrix}^T$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}^T(1) & \mathbf{s}^T(2) & \cdots & \mathbf{s}^T(N) \end{bmatrix}^T = \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{bmatrix}$$

$$\mathbf{s}^T(t) = \begin{bmatrix} 1 & s(x_1(t)) & \cdots & s(x_n(t)) & s(x_1(t)) \times s(x_2(t)) & \cdots & s(x_1(t)) \times \cdots \times s(x_n(t)) \end{bmatrix}$$

and $s_{new}(x_j(t))$ is the updated state of cell x_j at time step t. $s(x_j(t))$ is the state of cell x_j at time step t. Matrix $\mathbf{S}$ can be decomposed as $\mathbf{S} = \mathbf{E} \times \mathbf{Q}$, where

$$\mathbf{E} = \begin{bmatrix} e_1(1) & \cdots & e_P(1) \\ \vdots & & \vdots \\ e_1(N) & \cdots & e_P(N) \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \cdots & \mathbf{e}_P \end{bmatrix}$$

is an orthogonal matrix,

$$\mathbf{E}^T \times \mathbf{E} = Diag \begin{bmatrix} \mathbf{e}_1^T \times \mathbf{e}_1 & \cdots & \mathbf{e}_P^T \times \mathbf{e}_P \end{bmatrix}$$

and $\mathbf{Q}$ is an upper triangular matrix with unity diagonal elements

$$\mathbf{Q} = \begin{bmatrix} 1 & q_{12} & q_{13} & \cdots & q_{1P} \\ & 1 & q_{23} & \cdots & q_{2P} \\ & & \ddots & \ddots & \vdots \\ & & & 1 & q_{P-1P} \\ & & & & 1 \end{bmatrix}$$

Equation (4) can then be represented as

$$\mathbf{s}_{new} = \mathbf{E} \times \mathbf{Q} \times \tilde{\mathbf{a}} = \mathbf{E} \times \check{\mathbf{a}} \tag{5}$$

where $\check{\mathbf{a}} = \mathbf{Q} \times \tilde{\mathbf{a}} = [\check{a}_1 \cdots \check{a}_P]^T$. Therefore,

$$\mathbf{s}_{new}^T \times \mathbf{s}_{new} = \check{\mathbf{a}}^T \times \mathbf{E}^T \times \mathbf{E} \times \check{\mathbf{a}} \tag{6}$$

Due to the orthogonality of matrix $\mathbf{E}$, the contribution each term $\mathbf{s}_i$ $(i = 1, \cdots, P)$ makes to $\mathbf{s}_{new}$ can be calculated from equation (6) as

$$[ct]_i = \frac{\check{a}_i^2 \times \mathbf{e}_i^T \times \mathbf{e}_i}{\mathbf{s}_{new}^T \times \mathbf{s}_{new}} \tag{7}$$

The neighbourhood selection is entirely dependent on $[ct]_i$. The selection process can be summarized as follows

1. All the terms $\mathbf{s}_i$ $(i = 1, \cdots, P)$ are considered as candidates for $\mathbf{s}_{new}$. For $i = 1, \cdots, P$, calculate

$$\mathbf{e}_1^{(i)} = \mathbf{s}_i, \quad \tilde{a}_1^{(i)} = \frac{\mathbf{e}_1^{(i)} \times \mathbf{s}_{new}}{(\mathbf{e}_1^{(i)})^T \times \mathbf{e}_1^{(i)}}, \quad [ct]_1^{(i)} = \frac{(\tilde{a}_1^{(i)})^2 \times (\mathbf{e}_1^{(i)})^T \times \mathbf{e}_1^{(i)}}{\mathbf{s}_{new}^T \times \mathbf{s}_{new}}$$

   If $[ct]_1^{(j)} = max\{[ct]_1^{(i)}, i = 1, \cdots, P\}$, then the jth term $\mathbf{s}_j$ is selected. Let $\mathbf{e}_1 = \mathbf{e}_1^{(j)}$, $\tilde{a}_1 = \tilde{a}_1^{(j)}$ and $[ct]_1 = [ct]_1^{(j)}$.

2. All the terms $\mathbf{s}_i$ $(i = 1, \cdots, P, i \neq j)$ are considered as candidates for $\mathbf{s}_{new}$. For $i = 1, \cdots, P, i \neq j$, calculate

$$q_{12} = \frac{\mathbf{e}_1^T \times \mathbf{s}_i}{\mathbf{e}_1^T \times \mathbf{e}_1}, \quad \mathbf{e}_2^{(i)} = \mathbf{s}_i - q_{12}\mathbf{e}_1, \quad \tilde{a}_2^{(i)} = \frac{\mathbf{e}_2^{(i)} \times \mathbf{s}_{new}}{(\mathbf{e}_2^{(i)})^T \times \mathbf{e}_2^{(i)}}, \quad [ct]_2^{(i)} = \frac{(\tilde{a}_2^{(i)})^2 \times (\mathbf{e}_2^{(i)})^T \times \mathbf{e}_2^{(i)}}{\mathbf{s}_{new}^T \times \mathbf{s}_{new}}$$

If $[ct]_2^{(k)} = max\{[ct]_2^{(i)}, i = 1, \cdots, P, i \neq j\}$, then the kth term $\mathbf{s}_k$ is selected. Let $\mathbf{e}_2 = \mathbf{e}_2^{(k)}$. $\tilde{a}_2 = \tilde{a}_2^{(k)}$ and $[ct]_2 = [ct]_2^{(k)}$.

3. Follow the procedure in (2) until either $1 - \sum_{i=1}^{P_f}[ct]_i < c_{off}$, $P_f < P$ or when $P_f = P$. c_{off} is a desired tolerance value.

3 Rule selection

The correct and minimal neighbourhood can be detected from the set of terms selected using the procedure in Section 2.2 (for details see Section 4.1). Denote the neighbourhood obtained as $\{cell(x_{n_1}), \cdots, cell(x_{n_2})\}$, then the Boolean form of the rule to be identified can be written as

$$s_{new}(x_j) = a_0 \oplus a_1 s(x_{n_1}) \oplus \cdots \oplus a_{P_1}(s(x_{n_1}) * \cdots * s(x_{n_2})) \tag{8}$$

where $P_1 = 2^{n_2 - n_1 + 1} - 1$, $(n_2 > n_1)$. However the selected terms do not correspond to the terms in equation (8) due to the significant difference between $\oplus$ and $\times$ operators. It is therefore necessary to use a Genetic Algorithm (GA) to search for the matching rule. However, now the number of rules the GA can select from has been considerably reduced because the neighbourhood the rule is operating on has been determined by the neighbourhood detection procedure in Section 2.2.

The GA used in this paper is composed of three parts: a population, an evaluation function, and a reproduction process, these are described below:

3.1 The population

The GA search is designed to select the appropriate terms from a term set which comprises all the possible combinations of states of cells identified in the neighbourhood detection procedure. The ith individual in the GA population is therefore defined as an $1 \times P_1$ binary vector c_i. Each entry in c_i corresponds to a term in the set

$$c_i(1) \rightarrow 1,\ c_i(2) \rightarrow s(x_{n_1}),\ c_i(3) \rightarrow s(x_{n_1+1}),\ \cdots,\ c_i(n_2 - n_1 + 2) \rightarrow s(x_{n_2}),$$
$$c_i(n_2 - n_1 + 3) \rightarrow s(x_{n_1}) * s(x_{n_1+1}),\ \cdots,\ c_i(P_1) \rightarrow s(x_{n_1}) * \cdots * s(x_{n_2})$$

where $c_i(j) = 1$ indicates that the associated term has been selected and $c_i(j) = 0$ otherwise. Define

$$f_t = \left[\begin{array}{cccc} 1 & s(x_{n_1}(t)) & \cdots & s(x_{n_1}(t)) * \cdots * s(x_{n_2}(t)) \end{array}\right],\quad C = \left[\begin{array}{cccc} c_1 & c_2 & \cdots & c_m \end{array}\right]^T$$

where m is the population size and t indicates the position of the data point. The starting population is generated by filling each chromosome with a randomly generated binary vector of P_1 bits.

3.2 The evaluation function

The evaluation function is used to assess the performance of each chromosome in regenerating the behavior of the observed spatio-temporal evolution. Firstly, define the error function as $Error(i) = \sum_j^{SET} |o(i,j) - \hat{o}(i,j)|$, where $o(i,j)$ is the original measured state at data point j

for chromosome i and $\hat{o}(i,j) = c_i \oplus f_j$ is the predicted state.
The evaluation function

$$eva(i) = \frac{MAX(Error(i)) - Error(i)}{MAX(Error(i)) - MIN(Error(i))}$$

is then introduced to normalize the error function and act as the driving force to minimize the error.

3.3 The reproduction process

The reproduction process contains two stages: parent selection and genetic operation. The purpose of parent selection is to give more reproductive chances, on the whole, to those chromosomes that are the most fit. This paper uses the roulette wheel parent selection technique in [10]. The selected parent is then used for genetic operations in the breeding process. Crossover and mutation are the two most commonly used genetic operators. Crossover produces new chromosomes which have some segments of both parents' genetic structure. Mutation randomly alters one or more bits in a chromosome with a probability equal to the mutation rate. For details see [11] and [12].

When the GA is run, the population, evaluation function and reproduction process work in combination to affect the evolution in the search for a matching CA rule. After initializing the population, each chromosome is evaluated by the evaluation function and a series of cycles of replacing the current population by a new population begins. In each cycle, the evaluation, the parent selection, the genetic operation and the insertion of the new population to replace the old population are performed sequentially. The search process terminates when all chromosomes in the new population converge to a single individual.
Compared to the algorithm in [9] where a multi-objective GA with subpopulations was employed, the GA in this paper is much simpler and easier to implement. The pre-determined minimal neighbourhood which is obtained in the neighbourhood detection procedure enables the second search objective to minimize the neighbourhood structure in the GA to be discarded. It is also possible to eliminate the subpopulations designed for the multi-objective approach of the earlier method. These simplications will therefore considerably accelerate the rule identification process.

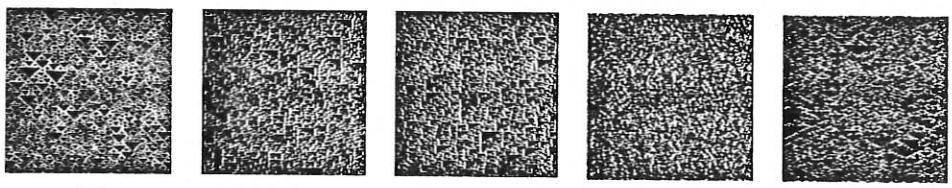
4 Simulation studies

Simulation results will be presented initially to illustrate the neighbourhood detection algorithm. The identified neighbourhood will then be used, in Section 4.2, as the input to the GA routine to determine the CA rule.

4.1 Neighbourhood detection

4.1.1 Spatio-temporal patterns produced by 1-D CA *Rule*22 on various 3-site neighbourhoods

The spatio-temporal patterns produced by 1-D CA *Rule*22 on various 3-site neighbourhoods are shown in Figure 2. All of these were developed on a 200×200 lattice with time evolution from top to bottom and a periodic boundary condition. That is the lattice is taken as a circle in the horizontal dimension, so the first and last sites are identified as if they lay on a circle of finite radius. The evolution started from an initial condition of a randomly generated binary vector. The neighbourhoods of $cell(j)$ for $(a)-(e)$ are $\{cell(j-1), cell(j), cell(j+1)\}$, $\{cell(j-2), cell(j-1), cell(j)\}$, $\{cell(j), cell(j+1), cell(j+2)\}$, $\{cell(j-4), cell(j-1), cell(j+3)\}$, and $\{cell(j-2), cell(j), cell(j+2)\}$ respectively.



(a) (b) (c) (d) (e)

Figure 2. Evolution of 1-D CA *Rule*22 on various 3-site neighbourhoods

Although the patterns were all produced under the same rule, Figure 2 clearly shows the diversity induced by the different neighbourhoods associated with the rule. Figure 2 (a) was produced by the symmetric von Neumann neighbourhood and is therefore composed of inverted symmetric triangles of varying sizes. However, the triangle structures in Figure 2 (b) and (c) only represent the left and right half of the triangles in Figure 2 (a). This is due to the left- and right-shift nature of the corresponding neighbourhoods $\{cell(j-2), cell(j-1), cell(j)\}$ and $\{cell(j), cell(j+1), cell(j+2)\}$. The irregularity of the neighbourhood $\{cell(j-4), cell(j-1), cell(j+3)\}$ produced the blurred and twisted triangles in Figure 2 (d). The increase of the distance between neighboring cells in the neighbourhood $\{cell(j-2), cell(j), cell(j+2)\}$ (compared to $\{cell(j-1), cell(j), cell(j+1)\}$) is clearly illustrated in Figure 2 (e) where the triangles are flattened.

4.1.2 Neighbourhood detection of 1-D CA *Rule*22

Assume initially that the largest possible neighbourhood is a 9-site neighbourhood defined by $\{cell(j), cell(j-4), cell(j-3), cell(j-2), cell(j-1), cell(j+1), cell(j+2), cell(j+3), cell(j+4)\}$. Define the neighbourhood vector **nei** as

$$\mathbf{nei} = \begin{bmatrix} cell(j) & cell(j-4) & cell(j-3) & cell(j-2) & cell(j-1) \\ cell(j+1) & cell(j+2) & cell(j+3) & cell(j+4) \end{bmatrix}^T$$

The candidate term set SET which is based on this assumed neighbourhood will initially be constructed as

$$SET = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

where $1, 2, 3, 4, 5, 6, 7, 8, 9$ denote the elements in **nei**. For instance, entry 5 represents the fifth element in **nei** and is therefore associated with $cell(j-1)$, and so on. The full SET consists of $P = 2^9 - 1 = 511$ terms/rows. Each row in SET represents a candidate term which corresponds to an $\mathbf{s}_i, i = 1, \cdots, P$ in matrix **S** in equation (4) in Section 2.2. For instance, the first row (1 0 0 0 0 0 0 0 0) represents $s(j)$ only while the last row (1 2 3 4 5 6 7 8 9) corresponds to a product of nine states $s(j) \times s(j-4) \times s(j-3) \times s(j-2) \times s(j-1) \times s(j+1) \times s(j+2) \times s(j+3) \times s(j+4)$. Data extracted from the spatio-temporal patterns in Figure 2 $(a) - (e)$ were used in the neighbourhood detection. For each pattern, 1000 data points were used and the tolerance value C_{off} was set as 0. Applying the neighbourhood detection technique in Section 2.2 to each pattern produced matrixes (a)-(e). The last term on each row in each matrix represents the contribution $[ct]_i, (i = 1, \cdots, 7)$ the term on the same row makes to $s_{new}(j)$. The maximum contribution is 1.0 so multiplying the last term in each row by 100 would give the percentage contribution of the term on the same row.

$$\begin{bmatrix} 6&0&0&0&0&0&0&0&0&0.2578\\ 1&6&0&0&0&0&0&0&0&0.0661\\ 1&0&0&0&0&0&0&0&0&0.0734\\ 1&5&0&0&0&0&0&0&0&0.1322\\ 5&0&0&0&0&0&0&0&0&0.0608\\ 1&5&6&0&0&0&0&0&0&0.0559\\ 5&6&0&0&0&0&0&0&0&0.1059 \end{bmatrix}$$
(a)

$$\begin{bmatrix} 4&0&0&0&0&0&0&0&0&0.1818\\ 4&5&0&0&0&0&0&0&0&0.2078\\ 5&0&0&0&0&0&0&0&0&0.1515\\ 1&5&0&0&0&0&0&0&0&0.0836\\ 1&0&0&0&0&0&0&0&0&0.1564\\ 1&4&5&0&0&0&0&0&0&0.0430\\ 1&4&0&0&0&0&0&0&0&0.1759 \end{bmatrix}$$
(b)

$$\begin{bmatrix} 7&0&0&0&0&0&0&0&0&0.1867\\ 1&7&0&0&0&0&0&0&0&0.1604\\ 1&0&0&0&0&0&0&0&0&0.1265\\ 1&6&0&0&0&0&0&0&0&0.1034\\ 1&6&7&0&0&0&0&0&0&0.0471\\ 6&7&0&0&0&0&0&0&0&0.2508\\ 6&0&0&0&0&0&0&0&0&0.1250 \end{bmatrix}$$
(c)

$$\begin{bmatrix} 2&0&0&0&0&0&0&0&0&0.0614\\ 2&5&0&0&0&0&0&0&0&0.0915\\ 5&0&0&0&0&0&0&0&0&0.1261\\ 2&8&0&0&0&0&0&0&0&0.0604\\ 2&5&8&0&0&0&0&0&0&0.1409\\ 5&8&0&0&0&0&0&0&0&0.1191\\ 8&0&0&0&0&0&0&0&0&0.2139 \end{bmatrix}$$
(d)

$$\begin{bmatrix} 4&0&0&0&0&0&0&0&0&0.1333\\ 1&4&0&0&0&0&0&0&0&0.2657\\ 1&0&0&0&0&0&0&0&0&0.1211\\ 1&7&0&0&0&0&0&0&0&0.0702\\ 7&0&0&0&0&0&0&0&0&0.2059\\ 1&4&7&0&0&0&0&0&0&0.0572\\ 4&7&0&0&0&0&0&0&0&0.1466 \end{bmatrix}$$
(e)

The matrixes above show that for each pattern just 7 terms were selected from the original set of 511 terms. For the patterns in Figure 2 $(a) - (e)$, the positions of the cells in the neighbourhoods as selected from **nei** are $(1, 5, 6)$, $(1, 4, 5)$, $(1, 6, 7)$, $(2, 5, 8)$, $(1, 4, 7)$ respectively. Referring back to the definition of **nei**, the corresponding neighbourhoods are the von Neumann neighbourhood, $\{cell(j-2), cell(j-1), cell(j)\}$, $\{cell(j), cell(j+1), cell(j+2)\}$, $\{cell(j-4), cell(j-1), cell(j+3)\}$, and $\{cell(j-2), cell(j), cell(j+2)\}$. These are the same as the known neighbourhoods used to produce the patterns and are all absolutely correct. Despite the fact that the 3 elements which constitute the 7 terms are all different for the five patterns, the way the 7 terms are formed is the same. This is because although the neighbourhoods are different, the underlying local rules are the same, that is, 1-D 3-site $Rule22$. However, the corresponding $[ct]_i$'s are largely dependent on the different data set tested and are therefore not necessarily the same in each case.

Other 1-D CA rules with neighbourhood sizes larger than three were also tested using the same procedure and the results are all correct. For simplicity the results are not presented in the paper.

4.1.3 Neighbourhood detection of a 2-D CA Rule

Data extracted from the spatio-temporal patterns produced by the evolution of the 2-D $Rule$(01011111 10100011 01001001 01000000) on the 2-D von Neumann neighbourhood $\{cell(i-1,j), cell(i,j-1), cell(i,j), cell(i,j+1), cell(i+1,j)\}$ will be used to illustrate the neighbourhood detection procedure for the 2-D case.
The initial neighbourhood was assumed to be a 2-D 9-site Moore Neighbourhood defined by $\{cell(i,j), cell(i+1,j-1), cell(i+1,j), cell(i+1,j+1), cell(i,j-1), cell(i,j+1), cell(i-1,j-1), cell(i-1,j), cell(i-1,j+1)\}$, and the neighbourhood vector **nei** was defined as

$$\mathbf{nei} = \begin{bmatrix} cell(i,j) & cell(i+1,j-1) & cell(i+1,j) & cell(i+1,j+1) & cell(i,j-1) \\ cell(i,j+1) & cell(i-1,j-1) & cell(i-1,j) & cell(i-1,j+1) \end{bmatrix}^T$$

The candidate term set SET for the 2-D rule was constructed exactly as for 1-D $Rule22$ but with entries pointing to different cells. For example, entry 5 denotes the fifth element in **nei** but is now related to $cell(i,j-1)$ and similarly for the other assignments. So that for example, the last row (1 2 3 4 5 6 7 8 9) represents $s(i,j) \times s(i+1,j-1) \times s(i+1,j) \times s(i+1,j+1) \times s(i,j-1) \times s(i,j+1) \times s(i-1,j-1) \times s(i-1,j) \times s(i-1,j+1)$ in the 2-D case.
A total of 1000 data points were used for the neighbourhood detection and the tolerance value C_{off} was set as 0. Twenty terms were selected from the original set of 511 terms and these are shown in matrix (f).

$$\begin{bmatrix}
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2324 \\
1 & 3 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0.0349 \\
3 & 5 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0276 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0326 \\
1 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0547 \\
1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0382 \\
1 & 5 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0422 \\
5 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0507 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0344 \\
3 & 5 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0.0157 \\
1 & 3 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0.0240 \\
1 & 3 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0252 \\
1 & 3 & 5 & 6 & 8 & 0 & 0 & 0 & 0 & 0.0123 \\
3 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0153 \\
1 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0038 \\
1 & 3 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0045 \\
3 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0087 \\
1 & 3 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0092 \\
1 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0096 \\
3 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0209
\end{bmatrix}$$

(f)

The last term in each row in the matrix represents the contribution the term on the same row makes to $s_{new}(j)$. These 20 terms cover five elements $1, 3, 5, 6, 8$, which referring back to the neighbourhood vector **nei**, represent the neighbourhood given by $\{cell(i,j), cell(i+1,j), cell(i,j-1), cell(i,j+1), cell(i-1,j)\}$. This identified neighbourhood is exactly the same as the original neighbourhood which was used to generate the data set.

4.2 Identification of the CA rules

4.2.1 Selection of 1-D CA rules

The genetic algorithm described in Section 3 will be used to identify the CA rules based on the neighbourhood structure which was obtained from the neighbourhood detection algorithm. The GA search was tested over a large set of 1-D CA rules with neighbourhoods of various sizes which were identified in Section 4.1.2. Some of the results are shown in Table 1. For each

rule, 100 trials were conducted with different initial populations. The search was terminated when 400 generations had been reached. For simplicity only the average and standard deviation (*std.dev.*) values are listed in Table 1.

For *Rule*22 the data points for the GA search can be extracted from any of the five spatio-temporal patterns in Figure 2. This is possible because all the patterns were produced under *Rule*22, although over different neighbourhoods. Now the neighbourhoods have been determined it will not make any difference which pattern is used in the GA search. This also applies to the other rules. Assume the neighbourhood for $cell(x_2)$ is $\{cell(x_1), cell(x_2), cell(x_3)\}$, the Boolean form of *Rule*22 is then identified as $s_{new}(x_2) = s(x_1) \oplus s(x_2) \oplus s(x_3) \oplus (s(x_1) * s(x_2) * s(x_3))$.

Table 1. Summary of results obtained in evolving some 1-D CA rules with various sizes of neighbourhoods using GA

n	rule	generations		errors		no. of terms		av.r.t.
		mean	std.dev.	mean	std.dev.	mean	std.dev.	
3	*Rule*22	14.68	5.12	0	0	4	0	10.11*min.*
	*Rule*54	18.91	4.37	0	0	4	0	13.56*min.*
4	*Rule*179	35.40	6.83	0	0	7	0	24.49*min.*
	*Rule*924	46.76	6.04	0	0	9	0	30.21*min.*
5	*Rule*1	81.34	7.15	0	0	15	0	58.43*min.*
	*Rule*2	89.57	6.36	0	0	18	0	63.59*min.*
6	*Rule*3	132.90	7.78	0	0	25	0	89.35*min.*
	*Rule*4	108.46	7.31	0	0	19	0	76.88*min.*
7	*Rule*5	172.81	10.03	0	0	16	0	98.85*min.*
	*Rule*6	194.46	9.52	0	0	58	0	109.94*min.*
8	*Rule*7	230.50	13.22	0	0	44	0	152.36*min.*
	*Rule*8	206.18	15.63	0	0	30	0	130.54*min.*
9	*Rule*9	300.35	26.17	0	0	52	0	210.80*min.*
	*Rule*10	312.28	20.93	0	0	54	0	226.16*min.*

n indicates the size of the neighbourhood. **av.r.t.** represents the average run time in the search for the optimal solution in one trial. 100 trials were made for each problem. The "**generations**" column indicates the number of generations reached before the optimal solution was found. "**no. of terms**" shows the number of terms selected in the optimal solution.

In Table 1, only rules with small neighbourhoods are enumerated. This is due to the fact that the numerical label and the truth table form of the rules can be very cumbersome when the neighbourhood size is larger than 4. Each identified rule in Table 1 produces a correct truth table which, together with the pre-detected minimal neighbourhood, defines a minimal and correct Boolean rule for the corresponding spatio-temporal pattern.

It can be seen from Table 1 that the average run time depends largely on the size of the neighbourhood. For each rule, the average run time in Table 1 is considerably smaller than in Table 6 Section 4 in [9], where without using the neighbourhood detection algorithm the solutions had to be selected from a substantially larger set of possible rules. For example for the 3-site 1-D rules the rule set for the GA search without the initial neighbourhood detection algorithm would comprise a massive $2^{2^9} = 1.3408e + 154$ rules. In comparison the rule set for the GA search based on the algorithm introduced in this study and which generated the results in Table 1 consisted of a substantially reduced search over just $2^{2^3} = 256$ rules. In addition since the pre-defined neighbourhood is minimal the GA used above evolves with consideration of only one objective, to minimize the matching errors, while the GA without neighbourhood detection involves a second search objective, to minimize the structure of the neighbourhood.

4.2.2 Selection of 2-D CA rules

The neighbourhood obtained in Section 4.1.3 will be used for the GA search of the 2-D CA rule in Section 4.1.3. A total of 100 trials were tested with different initial assignments. For each trial the search was terminated after 400 generations.

Table 2. The tabular form of the identified 2-D Boolean rule

neighbourhood	$s_{new}(i,j)$	B	$\stackrel{\ominus}{C}$	$\stackrel{\oplus}{D}$	$\stackrel{\oplus}{E}$	$\stackrel{\oplus}{F}$	$\stackrel{\oplus}{G}$	$\stackrel{\oplus}{H}$	$\stackrel{\ominus}{I}$	$\stackrel{\oplus}{J}$	$\stackrel{\ominus}{K}$	$\stackrel{\oplus}{L}$	$\stackrel{\ominus}{M}$
00000	0	0	0	0	0	0	0	0	0	0	0	0	0
00001	1	0	0	1	1	1	1	1	1	1	1	1	1
00010	0	0	0	0	0	0	0	0	0	0	0	0	0
00011	1	0	0	1	1	1	1	1	1	1	1	1	1
00100	1	0	1	1	1	1	1	1	1	1	1	1	1
00101	1	0	1	0	0	1	1	1	1	1	1	1	1
00110	1	0	1	1	1	1	1	1	1	1	1	1	1
00111	1	0	1	0	0	1	1	1	1	1	1	1	1
01000	1	1	1	1	1	1	1	1	1	1	1	1	1
01001	0	1	1	0	0	0	0	0	0	0	0	0	0
01010	1	1	1	1	1	1	1	1	1	1	1	1	1
01011	0	1	1	0	0	0	0	0	0	0	0	0	0
01100	0	1	0	0	0	0	0	0	0	0	0	0	0
01101	0	1	0	1	1	0	0	0	0	0	0	0	0
01110	1	1	0	0	0	0	0	0	0	0	1	1	1
01111	1	1	0	1	1	0	0	0	0	0	1	1	1
10000	0	0	0	0	0	0	0	0	0	0	0	0	0
10001	1	0	0	1	1	1	1	1	1	1	1	1	1
10010	0	0	0	0	0	0	0	0	0	0	0	0	0
10011	0	0	0	1	1	1	1	1	1	0	0	0	0
10100	1	0	1	1	1	1	1	1	1	1	1	1	1
10101	0	0	1	0	0	1	1	1	0	0	0	0	0
10110	0	0	1	1	1	1	1	0	0	0	0	0	0
10111	1	0	1	0	0	1	1	0	1	0	0	0	1
11000	0	1	1	1	0	0	0	0	0	0	0	0	0
11001	1	1	1	0	1	1	1	1	1	1	1	1	1
11010	0	1	1	1	0	0	0	0	0	0	0	0	0
11011	0	1	1	0	1	1	1	1	1	0	0	0	0
11100	0	1	0	0	1	1	0	0	0	0	0	0	0
11101	0	1	0	1	0	1	0	0	1	1	1	0	0
11110	0	1	0	0	1	1	0	1	1	1	0	0	0
11111	0	1	0	1	0	1	0	1	0	1	0	1	0

$B = s(i,j-1)$, $C = s(i,j)$, $D = s(i-1,j)$, $E = s(i-1,j) \times s(i,j-1)$, $F = s(i,j) \times s(i-1,j)$, $G = s(i+1,j) \times s(i,j-1) \times s(i,j)$, $H = s(i+1,j) \times s(i,j) \times s(i,j+1)$, $I = s(i+1,j) \times s(i,j) \times s(i-1,j)$, $J = s(i+1,j) \times s(i,j+1) \times s(i-1,j)$, $K = s(i,j-1) \times s(i,j) \times s(i,j+1)$, $L = s(i+1,j) \times s(i,j-1) \times s(i,j) \times s(i-1,j)$, $M = s(i+1,j) \times s(i,j) \times s(i,j+1) \times s(i-1,j)$

The tabular form of the identified Boolean rule is shown in Table 2. The search results are shown in Table 3. The Boolean rule is the $\ominus$ combination of the selected 12 terms from B to M. As can be seen from the second column in Table 2, which matches the rule definition (01011111 10100011 01001001 01000000) exactly, this truth table is correct. Therefore, together with the pre-identified minimal neighbourhood, the $\ominus$ combination of B to M defines the desired correct and minimal Boolean rule.

Table 3. Summary of results obtained in evolving the 2-D CA rule with GA

n	generations		errors		no. of terms		av.r.t.
	mean	std.dev.	mean	std.dev.	mean	std.dev.	
5	77.51	6.63	0	0	12	0	52.75*min.*

The average run time in Table 3 is very similar to the the run time for 5-site 1-D rules in Table 1 and is considerably shorter than the time in Table 4 in Section 4 in [9]. Again this is because the neighbourhood detection algorithm was used to predetermine the neighbourhood for a full GA search. The main element that determines the search time is the size of the neighbourhood rather than the position occupied by each cell in the neighbourhood. This means that the dimensionality of the CA does not have a crucial impact on the search. A GA search over a large set of 2-D rules was conducted using the results from the neighbourhood detection routine. Because of the insensitivity to the dimensionality of CA, the results were very similar to the 1-D case in Table 1 and are not listed in the paper.

5 Conclusions

A new CA identification technique has been introduced which breaks the identification of CA rules from given patterns of data into two problems. First a new neighbourhood detection procedure is used to establish the correct neighbourhood. Then using this identified neighbourhood a GA search is conducted to determine the minimal CA rule.
The new approach can yield significant improvements in efficiency. For example a full GA search for a 3-site 1-D rule would comprise a search over a huge $2^{2^9} = 1.3408e+154$ possible rules. But by using the neighbourhood detection procedure to prune the GA search this can be reduced to a search over just 2^{2^3} rules. Simulation results for both 1-D and 2-D CA's clearly demonstrate the potential of the new algorithm.

6 Acknowledgment

Y.X.Yang gratefully acknowledges that this work is supported by a scholarship from the University of Sheffield. S.A.Billings gratefully acknowledges that part of this work is supported by EPSRC.

References

[1] O.Lafe, "Data Compression and Encryption Using Cellular Automata Transforms", *Engineering Applications of Artificial Intelligence*, vol.10, no.6, pp.581-591, 1997.

[2] G.Hernandez and H.J.Herrann, "Cellular Automata for Elementary Image Enhancement ", *Graphical models and Image Processing*, vol.58, no.1, pp.82-89, 1996.

[3] R.Cafiero etc. "Disordered One-dimensional Contact Process", *Physical Review E*, vol.57, no.5 ptA, pp.5060-5068, 1998,

[4] R.B.Pandey, "A Stochastic Cellular Automata Approach to Cellular Dynamics for HIV: Effect of Viral Mutation", *Theory in Biosciences*, vol.117, no.1, pp.32-41, 1998.

[5] K.Lindgren, etc. "Complexity of Two-dimensional Patterns", *Journal of Statistical Physics*, vol.91, no.5-6, pp.909-951, 1998.

[6] E.C.Monteiro, etc. "A Cellular Automaton Computer Model for the Study of Magnetic Detection of Cardiac Tissue Activation During Atrial Flutter", *IEEE Transactions on Magnetics*, vol.34, no.5 pt1, pp.3451-3454, 1998.

[7] A.I.Adamatskii, "Complexity of Identification of Cellular Automata ". *Automation and Remote Control*, vol.53, no.9, pt.2. pp.1449-1458. 1992.

[8] F.C.Richards, "Extracting Cellular Automaton Rules Directly from Experimental Data ", *Physica D*, 45, pp.189-202. 1990.

[9] Y.X.Yang and S.A.Billings "Extracting Boolean rules from CA patterns "submitted to *IEEE Trans System Man and Cybernetics* for publication.

[10] Z.Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*, Springer – Verlag, 1994.

[11] D.E.Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[12] A.J.Chipperfield and P.J.Fleming, "Genetic algorithms in control systems engineering ", *Control and Computers*, vol.23, pp.88-94, 1996.