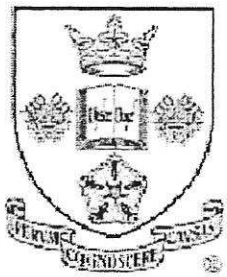
Monograph:
Mei, S.S, Billings, S.A. and Guo, L.Z. (2003) A Neighbourhood Selection Method For Cellilar Automata Models. Research Report. ACSE Report 832 . Department of Automatic Control and Systems Engineering

---

A Neighbourhood Selection Method For Cellular Automata Models

S. S. Mei, S. A. Billings and L. Z. Guo



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

Research Report No. 832
April 2003

A Neighbourhood Selection Method For Cellular Automata Models

S. S. Mei, S. A. Billings and L. Z. Guo
Department of Automatic Control and System Engineering
University of Sheffield
Mappin Street, Sheffield S1 3JD
United Kingdom

Abstract:
A new neighbourhood selection method is presented for both deterministic and probabilistic cellular automata models. The detection criteria are built explicitly on the corresponding contribution which is made to the value of each updated cell from each detected cell in the evolution. Theoretical analysis and numerical simulations demonstrate the effectiveness of this new method.

Keywords: Cellular automata, Neighbourhood, Probabilistic cellular automata, System identification

1. Introduction

Cellular Automata (CA) which were introduced in the 1950's and 1960's as models of self-reproduction (von Neumann 1951) have stimulated a great deal of interest and research. The two recent books by Ilachinski (2001) and by Wolfram (2002) discuss many of the results that are currently available.

CA can be classified into deterministic CA (DCA) and probabilistic CA (PCA) depending on the determinacy of the local transition rule. Both DCA and PCA have been widely used to model a variety of systems with local interactions in physics, chemistry, biology, and the social sciences. In almost all of these applications, the local transition rules were pre-chosen and the neighbourhood was predefined so as to capture the basics of the local interactions of the real systems involved. That is, both transition rules and neighbourhood were manually initialised. More recently, system identification methods were introduced into CA applications so that the transition rule could be extracted automatically from experimental data. Corno *et al* (2000) proposed a solution based on a genetic algorithm to directly identify the CA rule and successfully used the identified CA in fault coverage. A genetic algorithm was also used to extract the CA rule from CA patterns (Yang and Billings 2000a). Billings *et al* (2003a) reformulated the binary rules as simple polynomial models, and introduced the modified orthogonal least squares algorithm into CA identification. However, in most of these studies, the neighbourhood was still manually predefined as the cells that were immediately close to the cell to be updated. For instance, most CA use either the von Neumann, the Moore neighbourhoods, or some proposed larger neighbourhoods in order to model systems with long-range interactions (Kier and Cheng 1994), while others simply choose a minimal radius of neighbourhood (Adamatzky 1997). This usually introduces additional difficulties in CA transition rule identification because the rule space will be hugely expanded if redundant cells are incorrectly selected as neighbourhood cells. For example, a three-site one-dimensional CA will have $N = 2^{(2^3)} = 256$ possible rules while the number of possible rules will explode to $N = 2^{(2^9)} = 1.341e+154$ for a nine-site one-dimensional CA. The worst case arises if the neighbourhood is predefined incorrectly. For example, if one of the neighbourhood cells is not included in the predefined neighbourhood, then it may be impossible to find the correct transition rule. Yang and Billings 2000 presented a neighbourhood detection method to determine the neighbourhood before the transition rule was identified. But

this was essentially a term selection method. The distinction between neighbourhood and terms in CA identification is important and can be illustrated using the CA rule presented as $x_j(t) = \theta_1 s_1 s_2 + \theta_2 s_1 s_3 + \theta_3 s_2 s_3 + \theta_4 s_1 s_2 s_3$. In this equation, s_1, s_2 and s_3 are the neighbourhood cells while $s_1 s_2$, $s_1 s_3$, $s_2 s_3$ and $s_1 s_2 s_3$ are terms, and θ_i $(i = 1, \cdots, 4)$ represent model parameters. Note that the number of neighbourhood cells is often considerably less than the number of the cells included in CA model terms. Determining the neighbourhood cells is therefore a fundamentally important preliminary step in CA rule identification and in the present paper new neighbourhood selection methods for both deterministic and probabilistic CA are introduced. The new methods are based on detecting the contribution made to the value of each updated cell in the CA evolution. Neighbourhood selection for deterministic CA is presented in Section 3.1, and this is extended to solve the neighbourhood selection problem for probabilistic CA in Section 3.2. Simulation results are given in Section 4, which show that the new method can find the neighbourhood cell exactly, and conclusions are presented in Section 5.

2. Cellular Automata

Cellular automata are systems that evolve in discrete time over lattice structures composed of a large quantity of cells. The next state of each cell in a cellular automaton is updated synchronously according to local rules which depend on a given neighbourhood. If the state of the cells can only take either the value 0 or 1, this defines a binary CA. Attention in this study will be restricted to binary CA. When the transition rules are deterministic, these CA will be referred to as deterministic cellular automata (DCA).

Consider a d-dimensional lattice $\boldsymbol{L}$ consisting of the set of all integer coordinate vectors $j = (j_1, \cdots, j_d) \in \mathbf{Z}^d$. The n-cell DCA model of a spatio-temporal dynamical system defined over the lattice L can be expressed as follows

$$x_j(t) = f(N(x_j(t))) \tag{1}$$

where $x_j(t) \in \mathbf{B}$ is the updated state of the jth cell in $\boldsymbol{L}$ at time step t, f is the transition function which describes the local transition rule, and $N(x_j(t))$ is the neighbourhood of the jth cell in $\boldsymbol{L}$ at time step t. The neighbourhood is defined as follows

$$\begin{aligned} N(x_j(t)) = (&x_{j+p_1(1)}(t-1), x_{j+p_2(1)}(t-1), \cdots, x_{j+p_{m(1)}(1)}(t-1), \cdots, \\ &x_{j+p_1(k)}(t-k), x_{j+p_2(k)}(t-k), \cdots, x_{j+p_i(k)}(t-k), \cdots, x_{j+p_{m(k)}(k)}(t-k), \cdots, \\ &x_{j+p_1(\tau)}(t-\tau), x_{j+p_2(\tau)}(t-\tau), \cdots, x_{j+p_{m(\tau)}(\tau)}(t-\tau)) \end{aligned} \tag{2}$$

where $x_{j+p_i(k)}(t-k)$, $\left(1 \le k \le \tau,\ 1 \le i \le m(k), \text{ and } \sum_{k=1}^{\tau} m(k) = n\right)$ is the state of the l'th $\left(l = i + \sum_{1 \le k' < k} m(k')\right)$ entry in the neighbourhood region of the jth cell at time step t, $p_i(k)$ is the integer coordinate difference vector between the jth cell and the lth entry in the corresponding neighbourhood region, which specifies the spatial location of the neighbourhood cell, and k gives the temporal location of the corresponding neighbourhood cells. The neighbourhood structure is therefore defined by these parameters.

For convenience, the neighbourhood cells will be coded from 1 to n, and the neighbourhood cell $x_{j+p_i(k)}(t-k)$ will be denoted as $c_l^j(t)$ $(l = 1, \cdots, n)$. The neighbourhood described in eqn(2) can then be rewritten as

$$N(x_j(t)) = (c_1^j(t), c_2^j(t), \cdots, c_n^j(t)) \tag{3}$$

In some real systems, the transition rules of the CA will be statistical rather than deterministic, and CA with these properties are called statistical CA or probabilistic CA (PCA). The neighbourhood of PCA are defined in the same way as those for DCA, but the transition rules of PCA will be probabilistic. For an n-cell PCA with the neighbourhood defined as in eqn(3), the conditional probability for the cell x_j to take the value 1 when this cell evolves from a certain neighbourhood state is

$$a_i = P(1 | N(x_j(t)) = N_i) \tag{4}$$

where N_i stands for the i'th neighbourhood state, in which i is given by the neighbourhood cell value $i = \sum_{1 \le l \le n} 2^{n-l} c_l^j(t)$.

An alternative expression for the transition rule of a PCA can be obtained by decomposing the statistical evolution into a virtual DCA evolution together with a noise term (Yang and Billings 2000). The updated state of the cell x_j can be expressed as

$$x_j(t) = x_j^d(t) + \varepsilon(N(x_j(t))) = f(N(x_j(t))) + \varepsilon(N(x_j(t))) \tag{5}$$

where the term $x_j^d(t)$ is defined as the state of the updated cell in a virtual DCA and the function $f(\cdot)$ is a deterministic function, which represents the local rule associated with this virtual DCA. The output of this function only depends on the neighbourhood states. The noise term $\varepsilon(N(x_j(t)))$ may take the values 0,1 or -1 since the cell state in a binary CA can only be 0 or 1. From the Appendix, it is clear that the noise $\varepsilon(N(x_j(t)))$ is stationary up to order1 (Priestley 1981) when the cell is updated from the same neighbourhood state N_i. That is

$$E(\varepsilon(N(x_j(t)))) = \mu_{\varepsilon_i} \text{ for all } N(x_j(t)) = N_i,\ 1 \le i < 2^n \tag{6}$$

where μ_{ε_i} is a constant independent of t.

3. CA Neighbourhood Selection Method

The transition rule specifies the relationship between an updated cell and the corresponding neighbourhood. The neighbourhood shown as eqn(2) defines the cells that are involved in the evolution of each cell in the CA. The neighbourhood together with the local rule produce the dynamic behaviour of the CA system. Neighbourhood selection is therefore one of the most important issues in CA modelling. In this section, new CA neighbourhood selection methods will be introduced for both deterministic and probabilistic CA.

3.1 Neighbourhood Detection for Deterministic CA

To introduce the new neighbourhood detection algorithm, consider the more general problem of determining a component in a continuous function initially, where the determination can be made for each component based on the corresponding contribution to the values of this function. The results will then be modified to the binary CA case.

Let $\mathbf{R}^n$ be the n-dimensional real space, $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbf{R}^n$, and $f : \mathbf{R}^n \to \mathbf{R}$ a differentiable function. For any given component x_i in $\mathbf{x}^n$, let p_i be the natural projection from $\mathbf{R}^n$ to $\mathbf{R}^{n-1}$ with respect to x_i, that is, $p_i(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n) \equiv (x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$. Assume that f can be decomposed as

$$f = f' \circ p_i \tag{7}$$

where $f' : \mathbf{R}^{n-1} \subset \mathbf{R}^n \to \mathbf{R}$, and $\circ$ denotes composition.

Differentiating f with respect to x_i yields

$$\frac{\partial f}{\partial x_i} = \frac{\partial f'}{\partial p_i} \cdot \frac{\partial p_i}{\partial x_i} \tag{8}$$

Because p_i is the natural projection from $\mathbf{R}^n$ to $\mathbf{R}^{n-1}$ with respect to x_i, it is clear that

$$\frac{\partial p_i}{\partial x_i} \equiv 0 \tag{9}$$

Substituting (9) into (8) yields

$$\frac{\partial f(x_1, x_2, \cdots, x_n)}{\partial x_i} \equiv 0 \tag{10}$$

which implies that,

$$f(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n) \equiv f(x_1, \cdots, x_{i-1}, x_i', x_{i+1}, \cdots, x_n)$$

$$\text{for all } (x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) \in \mathbf{R}^{n-1}, \text{ and } x_i' \in \mathbf{R} \tag{11}$$

On the other contrary, if $f(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n) \equiv f(x_1, \cdots, x_{i-1}, x_i', x_{i+1}, \cdots, x_n)$, for all $(x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) \in \mathbf{R}^{n-1}$, and $x_i' \in \mathbf{R}$, which implies that $\frac{\partial f(x_1, x_2, \cdots, x_n)}{\partial x_i} \equiv 0$, then it can be proved that f can be decomposed as $f = f' \circ p_i$. In fact, for any $\mathbf{x}' = (x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) \in \mathbf{R}^{n-1}$, let

$$S_{x'} = p_i^{-1}(\mathbf{x}') =: \{\mathbf{x} \in \mathbf{R}^n \,|\, p_i(\mathbf{x}) = \mathbf{x}'\} \subseteq \mathbf{R}^n \tag{12}$$

Then $S_{x'} = \{(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n) \in \mathbf{R}^n \,|\, x_i \subseteq \mathbf{R}^n\}$.

Consider the image set of $S_{x'}$ under f

$$f(S_{x'}) = \{f(\mathbf{x}) \,|\, \mathbf{x} \in S_{x'}\} \subset \mathbf{R} \tag{13}$$

It is clear that the set $f(S_{x'})$ only contains a single point, denoted by $y_{x'}$. So that defining a function $f' : \mathbf{R}^{n-1} \to \mathbf{R}$, $f'(\mathbf{x}') = y_{x'}$, then f' satisfies $f = f' \circ p_i$.

Based on the above analysis, the following conclusions can be made. For any given component x_i in $\mathbf{x}^n$, there exists a map $f':\mathbf{R}^{n-1}\subset\mathbf{R}^n\to\mathbf{R}$, such that $f=f'\circ p_i$ if and only if $f(x_1,\cdots,x_n)\equiv f(x_1,\cdots,x_{i-1},x_i',x_{i+1},\cdots,x_n)$ for all $(x_1,x_2,\cdots,x_{i-1},x_{i+1},\cdots,x_n)\in\mathbf{R}^{n-1}$, and $x_i'\in\mathbf{R}$.

Eqn (7) implies that the system can be described without the ith component x_i. This conclusion provides a significant clue that can be used for variable selection in system modelling. Essentially this means that a variable is redundant if no contribution is made to the system output from this particular variable. However, it will often be difficult to investigate all the input vectors in the space $\mathbf{R}^n$ to determine which variables satisfy eqn(11). Even for a low dimensional discrete space, the number of the possible states of the variables which would need to be explored could be too large. But Boolean systems where the input and the output can only take values of 0 and 1 seem to be an exception to this observation. For Boolean systems, the following corollary can be drawn.

Let $y=f(x_1,x_2\cdots,x_n)$ be a Boolean function from $\mathbf{B}^n\to\mathbf{B}$, $\mathbf{B}=\{0,1\}$. Then for any given component $x_i\in\mathbf{B}$, there exists a Boolean function $f':\mathbf{B}^{n-1}\subset\mathbf{B}^n\to\mathbf{B}$, such that $f=f'\circ p_i$ where p_i is the natural projection from $\mathbf{B}^n$ to $\mathbf{B}^{n-1}$, if and only if $f(x_1,\cdots,x_{i-1},x_i,x_{i+1},\cdots,x_n)\equiv f(x_1,\cdots,x_{i-1},1-x_i,x_{i+1},\cdots,x_n)$ for all $(x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_n)\in\mathbf{B}^{n-1}$, and $x_i\in\mathbf{B}$.

If this extended result is applied to neighbourhood selection for the DCA model, a new neighbourhood detection algorithm can be derived. The basis of the new algorithm is to select a neighbourhood candidate set initially. Then each cell in the neighbourhood candidate set will be assessed according to the contribution made to the cell to be updated. The initial neighbourhood candidate set must be large enough to include all potential neighbourhood cells. The procedure is summarised below.

Define the initial candidate neighbourhood of the cell x_j at time step t as $N_0^c(x_j(t))=(c_1^{(j)}(t),c_2^{(j)}(t),\cdots,c_n^{(j)}(t))$, where $c_i^{(j)}(t),(i=1,2,\cdots,n)$ denotes the corresponding candidate cell. The initial neighbourhood candidate set is $NCS_0=\{c_1^{(j)}(t),c_2^{(j)}(t),\cdots,c_n^{(j)}(t)\}$. Set the initial number of selected cells as $l(0)$=0. The state of cell x_j at time step t can be obtained through the evolution from the candidate neighbourhood $N_0^c(x_j(t))$. A series of data pairs can then be constructed as $(N_0^c(x_j(t)),x_j(t))$, (t=1,2,...,m) from the observed CA pattern. At the k'th iteration step, the candidate neighbourhood is $N_{k-1}^c(x_j(t))$, the neighbourhood candidate set is $NCS_{k-1}=\{c_1^{(j)}(t),c_2^{(j)}(t),\cdots,c_{n+1-k}^{(j)}(t),c_{d_1}^{(j)}(t),\cdots,c_{d_{l(k-1)}}^{(j)}(t)\}$, where d_1, d_2, $\cdots$, $d_{l(k-1)}$ are the $l(k-1)$ indices of the cells that are selected as neighbourhood cells from $\{c_{n+2-k}^{(j)}(t),\cdots,c_n^{(j)}(t)\}$ after (k-1) steps detection. Assume that the detection decision is to be made for the cell $c_{n+1-k}^{j}(t)$, where the data pairs $(N_{k-1}^c(x_j(t)),x_j(t))$ (t=1,2,...,m) are obtained from the observed CA pattern. These data points are then searched to determine if the conditions

$$\begin{cases} c_i^{(j)}(t_0)=c_i^{(j)}(t_1) & \text{for all } i<n+1-k \text{ and } i=d_1,d_2,\cdots,d_{l(k-1)} \\ c_{n+1-k}^{(j)}(t_0)=1-c_{n+1-k}^{(j)}(t_1) & \\ x_j(t_0)\big|_{NCS_k(x_j,t_0)}\neq x_j(t_1)\big|_{NCS_k(x_j,t_1)} & \end{cases} \tag{14}$$

hold, where t_0 and t_1 ($t_0 \in [1\ m]$, $t_1 \in [1\ m]$) are two different data points, and $A|_B$ denotes the cell state A is updated from the neighbourhood state B. If eqn(14) is satisfied, the cell $c^j_{n+1-k}(t)$ will be selected and is kept in the neighbourhood candidate set, so that,
$l(k) = l(k-1)+1$, $c^j_{l(k)} = c^{(j)}_{n+1-k}(t)$, $NCS_k = NCS_{k-1}$.

Otherwise, the cell $c^j_{n+1-k}(t)$ is removed from the neighbourhood candidate set, so that $l(k) = l(k-1)$, $NCS_k = NCS_{k-1} \setminus \{c^j_{n+1-k}\}$ where A\B denotes $A \cap \overline{B}$. After n iteration steps, all cells in the initial candidate neighbourhood will have been processed, and all the $l(n)$ cells in the final neighbourhood candidate set NCS_n will be the selected neighbourhood cells.

The data pairs obtained from the CA pattern reveal the information about the relationship between the cell to be updated and the other cells. Therefore, the neighbourhood selection will be more reliable if more data pairs are obtained. In principle, for an n-cell neighbourhood, the number of data pairs should be near 2^n so that the information will be enough to make the neighbourhood selection. If the data pairs are not sufficient, the neighbourhood selection will produce an incorrect result, which should be detected at the validation stage.

3. 2 Neighbourhood Detection for Probabilistic CA

For PCA, the cell state may be flipped during the evolution of the CA, this is usually referred to as dynamic noise and is much more difficult to deal with than DCA (Billings and Yang 2003b). The criteria used for neighbourhood selection given in Section 3.1 can not be applied directly in the PCA case. Consider the dynamic noise CA case and for a given updated cell $x_j(t)$, define the initial neighbourhood candidate set as $NCS_0 = \{c^{(j)}_1(t), c^{(j)}_2(t), \cdots, c^{(j)}_n(t)\}$. Assume that this neighbourhood candidate set has been selected to be large enough to include all potential neighbours. Denoting the neighbourhood set as *NS*, the objective of neighbourhood selection is to determine the cells in the neighbourhood candidate set *NCS* and select all the cells $c^j_{k_i}(t)$ ($i = 1,2,\cdots l$) which satisfy $c^j_{k_i}(t) \in NS$. Assume that the detection decision is to be made for cell $c^j_i(t)$, and $c^j_i(t) \notin NS$. Therefore the change of cell $c^j_i(t)$ will not affect the neighbourhood states. In other words, the updated cell $x_j(t)$ will have the same neighbourhood state no matter what value the cell $c^j_i(t)$ takes. That is,

$$N(x_j(t))\Big|_{c^j_1=s_1,\cdots,c^j_{i-1}=s_{i-1},c^j_i=s_i,c^j_{i+1}=s_{i+1}\cdots,c^j_n=s_n} = N(x_j(t'))\Big|_{c^j_1=s_1,\cdots,c^j_{i-1}=s_{i-1},c^j_i=1-s_i,c^j_{i+1}=s_{i+1}\cdots,c^j_n=s_n}$$
$$\forall\ (s_1,s_2,\cdots s_n) \in \mathbf{B}^n \qquad (15)$$

where all the cells except $c^j_i(t)$ take the same value on both sides of the equation.

From eqn(5), the transition from $(s_1,\cdots s_{i-1},s_i,s_{i+1},\cdots s_n)$ to $x_j(t)$, there exists

$$\begin{aligned} & x_j(t)\Big|_{c^j_1=s_1,\cdots,c^j_{i-1}=s_{i-1},c^j_i=s_i,c^j_{i+1}=s_{i+1}\cdots,c^j_n=s_n} \\ & = x^d_j(t)\Big|_{c^j_1=s_1,\cdots,c^j_{i-1}=s_{i-1},c^j_i=s_i,c^j_{i+1}=s_{i+1}\cdots,c^j_n=s_n} + \varepsilon(N(x_j(t)))\Big|_{c^j_1=s_1,\cdots,c^j_{i-1}=s_{i-1},c^j_i=s_i,c^j_{i+1}=s_{i+1}\cdots,c^j_n=s_n} \end{aligned} \qquad (16)$$

Taking the expected value of both sides of eqn(16), and considering the determinacy of the virtual DCA, yields

$$\begin{aligned} & E(x_j(t)\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \\ & = x_j^d(t)\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n} + E(\varepsilon(N(x_j(t)))\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \end{aligned} \tag{17}$$

Similarly, consider another transition from $(s_1,\cdots s_{i-1},1-s_i,s_{i+1},\cdots s_n)$ to $x_j(t')$, where all the neighbourhood candidate cell states, except the cell $c_i(t)$, are the same as those in the transition shown in eqn(17), so that

$$\begin{aligned} & E(x_j(t')\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \\ & = x_j^d(t')\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n} + E(\varepsilon(N(x_j(t')))\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \end{aligned} \tag{18}$$

Consider the assumption that $c_i^j(t)$ is redundant, applying the results from Section 3.1 to the virtual DCA, yields

$$x_j^d(t)\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n} = x_j^d(t')\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n} \quad \forall\,(s_1,s_2,\cdots s_n)\in\mathbf{B}^n \tag{19}$$

Consider the results of eqn(15) and eqn(6), which shows that $\varepsilon(N(x_j(t))$ is stationary up to order 1, for all the same neighbourhood states, so that

$$\begin{aligned} & E(\varepsilon(N(x_j(t)))\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) = E(\varepsilon(N(x_j(t'))\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \\ & \qquad \forall\,(s_1,s_2,\cdots s_n)\in\mathbf{B}^n \end{aligned} \tag{20}$$

Substituting (19) and (20) into (18) and comparing to (17), yields

$$E(x_j(t)\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) = E(x_j(t')\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \tag{21}$$

Therefore, eqn(21) will hold if the cell $c_i^j(t)$ is not a neighbourhood cell of the updated cell $x_j(t)$. In other words, if there exists a $(s_1,s_2,\cdots s_n)\in\mathbf{B}^n$, such that

$$E(x_j(t)\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \neq E(x_j(t')\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n}) \tag{22}$$

then $c_i^j(t)$ is one of the neighbourhood cells of the updated cell $x_j(t)$.

In practice, the expectations in eqn (22) will have to be replaced by the sample means computed over the data obtained from the observed CA pattern. Eqn (22) should therefore be modified to allow for a tolerance.

$$< x_j\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n} > - < x_j\Big|_{c_1^j=s_1,\cdots,c_{i-1}^j=s_{i-1},c_i^j=1-s_i,c_{i+1}^j=s_{i+1}\cdots,c_n^j=s_n} > \geq \varepsilon_{cutoff} \tag{23}$$

where $<\cdot>$ denotes the mean value, and $\varepsilon_{cutoff}\in[0.02\ 0.20]$ is a range of typical tolerance values. The tolerance value can be chosen depending on the number of historical data from the observed CA patterns. The tolerance value should be reduced the more data that are available.

After n iteration steps, all the cells in the neighbourhood candidate set should have been detected and all the selected cells constitute the CA neighbourhood.

3.3 Verification

Verifying the selected neighbourhood cell is not straightforward. This is because often *no a-priori* information regarding which cells are involved in the CA evolution is available since only the spatio-temporary patterns generated by the CA rule are observable. However, the selected neighbourhood can be verified implicitly. Using the selected neighbourhood, the CA rule can be identified from the observed pattern with the help of a suitable CA identification algorithm. With the selected neighbourhood and the identified CA rule, a prediction of the CA pattern can be generated and this can be used to verify the identified model and the neighbourhood.

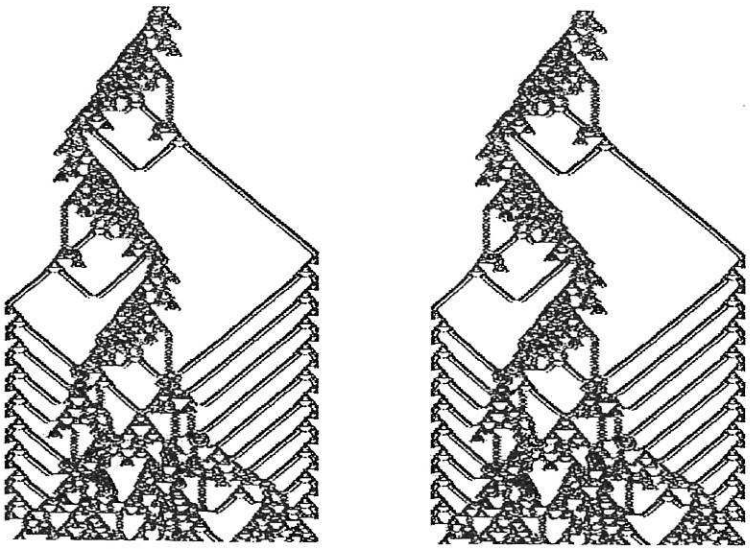
4. Simulation Results

4.1 Neighbourhood Selection for a Deterministic CA

Consider the 1-dimensional CA on a 250×1 lattice with the totalistic rule T88:

$$x_j(t) = \begin{cases} 1 & \text{if } \sum_{i=j-3}^{j+3} x_i(t-1) = 3,4 \text{ or } 6 \\ 0 & \text{else} \end{cases} \tag{24}$$

which is defined on the one-dimensional Moore Neighbourhood with radius r=3. The initial configuration was chosen as '0...00111111111101100...0'. The pattern illustrated in Figure 1 (a), which is known as Park's glider gun (Ilachinski 2001), was generated after 500 time steps of evolution. For each cell location 500 data points were generated. Periodic boundary conditions were used when considering the neighbourhood candidate data points. The initial candidate neighbourhood of cell $x_j(t)$ was chosen as $N(x_j(t)) = (x_j(t-1), x_{j-5}(t-1), x_{j-4}(t-1), x_{j-3}(t-1), x_{j-2}(t-1), x_{j-1}(t-1), x_{j+1}(t-1), x_{j+2}(t-1), x_{j+3}(t-1), x_{j+4}(t-1), x_{j+5}(t-1))$. The series of data pairs $(N(x_j(t)), x_j(t+1))$ were reconstructed from the observed CA pattern shown as Figure 1 (a). The data pairs were exactly the same for all the cells which were updated from the same neighbourhood candidate states and were only considered as one data pair. Finally the data pairs were collected from 250×500 raw data points and were then used for neighbourhood selection using the algorithm presented in Section 3.1. The results of using the new algorithm were that the cells $x_j(t-1)$, $x_{j-3}(t-1)$, $x_{j-2}(t-1)$, $x_{j-1}(t-1)$, $x_{j+1}(t-1)$, $x_{j+2}(t-1)$, $x_{j+3}(t-1)$ were selected as the CA neighbourhood. To verify the selection result, the reconstructed data pairs were further used to identify the CA rule based on the selected neighbourhood. An equivalent integer polynomial CA model was identified using the CA



(a) (b)

Figure 1: Example for a DCA
(a) Park's gun gliders pattern (b)Pattern generated using the identified model

Orthogonal Least Squáres algorithm (Billings *et al* 2003a). After 500 time steps of evolution with this identified polynomial CA rule, the pattern shown in Figure 1 (b) was generated from the initial configuration chosen as '0...00111111111101100...0', This produced either a left- or right- moving glider once every 238 iterations and was exactly the same as the pattern shown in Figure 1 (a). This implies that the neighbourhood cell selection and the identified CA model are correct.

4.2 Neighbourhood Selection for the Stochastic Version of Conway's Game of Life

The Game of Life (GoL) created by John Conway (Berlekamp 1982) is a two-dimensional lattice system in which the state of each lattice site depends on the neighbourhood cells according to deterministic local rules. Conway's original deterministic rule is an outer totalistic(code OT224) rule defined on the two-dimensional Moore Neighbourhood, which is given as:

$$x_{(i,j)}(t) = x_{3,\sum_{N(x_{(i,j)}(t))}' x(t-1)} + x_{(i,j)}(t)x_{2,\sum_{N(x_{(i,j)}(t))}' x(t-1)} \tag{25}$$

where $x_{\alpha,\beta}$ is the Kronecker delta and $\sum_{N(x_{(i,j)}(t))}' x(t-1)$ is a sum over all neighbours of the site $x_{(i,j)}(t)$. This can be extended into a PCA version and the rule of the extended stochastic GoL is

$$x_{(i,j)}(t) = P_B x_{3,\sum_{N(x_{(i,j)}(t))}' x(t-1)} + P_S x_{(i,j)}(t)x_{2,\sum_{N(x_{(i,j)}(t))}' x(t-1)} \tag{26}$$

where P_B, P_S are the birth probability and survival probability respectively. When $P_B = 1$ and $P_S = 1$ this stochastic rule becomes the same rule as the original deterministic GoL rule.

Let $P_B = 0.98$, $P_S = 0.96$, then using a 2 dimensional von Neumann neighbourhood, the initial configuration of a 50×50 lattice was set as the snapshot shown in Figure 2(a). The CA pattern was obtained after 200 time steps of evolution using the rule defined in eqn(26).A set of space-time "history" data points were generated from 200 snapshots of the CA pattern, and these were used for neighbourhood selection using the new algorithm presented in Section 3.2. Fourteen cells were chosen as the initial neighbourhood candidate cells and the tolerance value ε_{cutoff} was set to 0.18. The cells $x_{i,j}(t-1)$, $x_{i,j-1}(t-1)$, $x_{i,j+1}(t-1)$, $x_{i-1,j}(t-1)$, and $x_{i+1,j}(t-1)$ were selected as neighbourhood cells.

To verify the selection result, the selected neighbourhood cells were then used to identify the CA rule using the CA Orthogonal Forward Regression algorithm (Billings *et al.* 1988). An equivalent polynomial CA model was estimated for the deterministic part of the model. Using the same initial configuration, the predicted CA pattern was then obtained based on the identified model. To compare these two 2-dimensional CA patterns, the column density and the row density were used. The column density and the row density are given as

$$dx(i,t) = \frac{1}{n_r(2i_0+1)} \sum_{j=1}^{n_r} \sum_{k=i-i_0}^{k=i+i_0} x_{(k,j)}(t) \tag{27}$$

$$dy(j,t) = \frac{1}{n_c(2j_0+1)} \sum_{i=1}^{n_c} \sum_{k=j-j_0}^{k=j+j_0} s(x_{i,k},t) \tag{28}$$

where $dx(i,t)$, $dy(j,t)$ are the row density and the column density, n_r, n_c are row size and column size respectively. Periodic boundaries were used when these densities were calculated. The column density and the row density of the original CA pattern are shown as Figure2 (c), (f), while those of the predicted CA pattern are shown as Figure2 (d), (g). Note that Figure (2)(c), (f) and Figure 2 (d), (g) are not exactly the same. This is to be expected because the observed CA

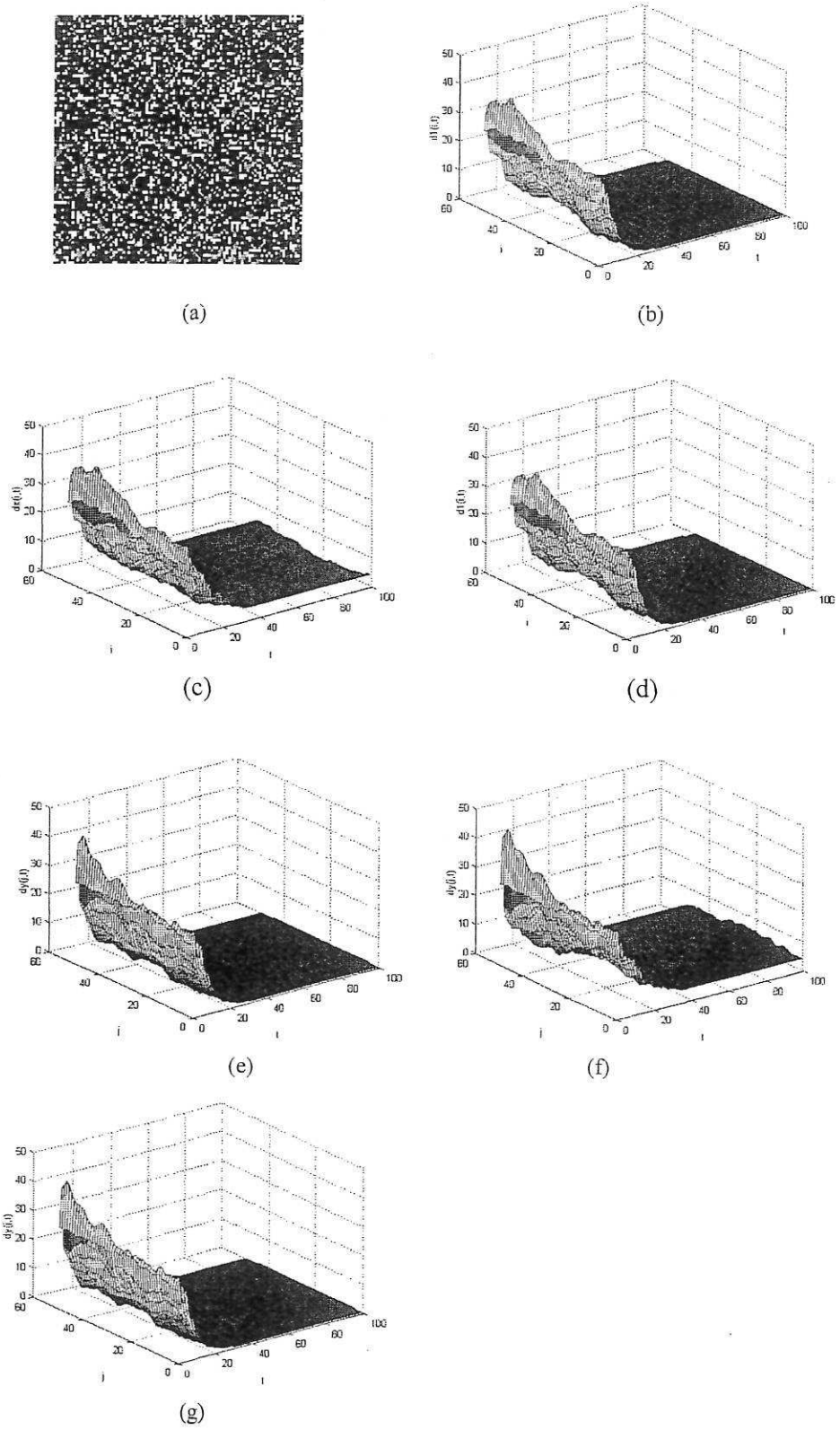


Figure 2. Stochastic Game of Life and the prediction results
(a) The initial random configuration; (b) The row density of the DCA pattern; (c) The row density of the PCA pattern; (d) The row density of the predicted CA pattern; (e) The column density of the DCA pattern; (f) The column density of the PCA pattern; (g) The column density of the predicted CA pattern;

includes the noise, where as the prediction from the model is noise free. To analyse the prediction result, the column density and the row density of the DCA pattern (observed from the CA evolution with the same rule as the above PCA but with $P_B = 1$, $P_S = 1$) are shown as Figure2 (b), (e), which are the same as those of the predicted pattern respectively. These results implicitly indicate that the neighbourhood cells were chosen correctly.

4. Conclusions

A new neighbourhood selection method has been derived for both deterministic and probabilistic CA models. The new algorithm allows the neighbourhood detection problem to be decoupled from the CA rule determination or CA model identification problem. The neighbourhood detection criteria are built explicitly on the contribution made to the value of each updated cell from each detected cell in the evolution. Simulation results on two complex CA rules confirm the effectiveness of the new method and demonstrate how CA models can be determined based on the selected neighbourhood.

Acknowledgement

SSM acknowledges the support of the University of Sheffield under the university scholarship scheme. SAB gratefully acknowledges that part of this work was supported by the Engineering and Physical Science Research Council, UK.

Appendix

Given an n-cell probabilistic CA, which has the transition rule expressed as eqn(4), the transition rule can also be represented as eqn(5). If the x_j have evolved from the neighbourhood N_i, the state of x_j can be given as follows

$$x_j(t)\Big|_{N(x_j(t))=N_i} = x_j^d(t)\Big|_{N(x_j(t))=N_i} + \varepsilon\Big|_{N(x_j(t))=N_i} \tag{A-1}$$

Taking expected value on both sides, and considering the determinacy of x_j^d, yields,

$$E(x_j(t)\Big|_{N(x_j(t))=N_i}) = x_j^d(t)\Big|_{N(x_j(t))=N_i} + E(\varepsilon\Big|_{N(x_j(t))=N_i}) \tag{A-2}$$

where the expectation of the state x_j can be obtained from the eqn(4)

$$E(x_j(t)\Big|_{N(x_j(t))=N_i}) = 1 \cdot a_i + 0 \cdot (1 - a_i) = a_i \tag{A-3}$$

Substituting (A-3) into (A-2), yields

$$E(\varepsilon\Big|_{N(x_j(t))=N_i}) = a_i - x_j^d(t)\Big|_{N(x_j(t))=N_i} = \mu_{\varepsilon_i} \tag{A-4}$$

where μ_{ε_i} is a constant independent of t. This implies that the noise term in eqn(5) is stationary up to order1 when the cell is updated from the same neighbourhood state N_i.

References:

Adamatzky A., (1997), Automatic Programming of Cellular Automata: Identification Approach, *Kybernetes*, 26 (2-3): 126-133.

Berlekamp, E., Conway, J. and Guy, R., (1982), *Wining Ways For Your Mathematical Plays*. Academic Press, New York.

Billings S. A., Korenberg M. J., Chen S., (1988), Identification of non-linear output-affine systems using an orthogonal least-squares algorithm, *International Journal of System Science* 19 (8): 1559-1568.

Billings S. A. and Yang, Y. X., (2003a), Identification of the neighbourhood and CA Rules from Spatio-temporal CA patterns, *IEEE Transactions on Systems Man and Cybernetics*, Part B, 33, 332-339.

Billings S. A. and Yang, Y. X., (2003b), Identification of probabilistic cellular automata, *IEEE Transactions on Systems Man and Cybernetics*, Part B, 33, 225-236.
Corno F., Reorda M. S. and Squillero G., (2000), Evolving cellular automata for self-testing hardware, *Lecture Notes in Computer Scicence* 1801: 31-40.
Ilachinski A., (2001), *Cellular Automata, A Discrete Universe,* Singapore.
Kier L. B. and Cheng C. K., (1994), A cellular automata model of an aqueous solution, *Journal of Chemical Information and Computer Sciences,* 34 (6): 1334-1337.
Priestley M. B., (1981), *Spectral Analysis and Time Serries*, Volume 1,Academic Press.
von Neumann J., (1951), The general logical theory of automata, in *Cerebral Mechanisms in Behavior*—The Hixon Symposium, New York Wiley.
Wolfram S., (2002), *A New Kind of Science*, Wolfram Media.
Yang YX and Billings SA, (2000a), Extracting Boolean rules from CA patterns, *IEEE Transaction on System, Man, Cybernetics* B 30 (4): 573-581.
Yang Y. X. and Billings S. A., (2000b), Neighbourhood detection and rule selection from cellular automata patterns, *IEEE Transaction on System, Man, Cybernetics* A 30 (6): 840-847.