Identification of Probabilistic Cellular Automata

S.A.Billings Y.X.Yang

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

Research Report No.752

July 1999

University of Sheffield

Identification of Probabilistic Cellular Automata

S.A.Billings Y.X.Yang
Dept. of Automatic Control and Systems Engineering
Univ. of Sheffield, Mappin Street, Sheffield S1 3JD, UK

Abstract

The identification of Probabilistic Cellular Automata (PCA) is studied using a new two stage neighbourhood detection algorithm. It is shown that a Binary Probabilistic Cellular Automaton (BPCA) can be described by an integer-parameterised polynomial corrupted by noise. Searching for the correct neighbourhood of a BPCA is then equivalent to selecting the correct terms, which constitute the polynomial model of the BPCA, from a large initial term set. It is proved that the contribution values for the correct terms can be calculated independently of the contribution values for the noise terms. This allows the neighbourhood detection technique developed for deterministic rules in [16] to be applied with a larger cutoff value to discard the majority of spurious terms and to produce an initial pre-search for the BPCA neighbourhood. A multi-objective GA search with integer constraints is then evolved to refine the reduced neighbourhood and to identify the polynomial rule which is equivalent to the probabilistic rule with the largest probability. A probability table representing the BPCA can then be determined based on the identified neighbourhood and the deterministic rule. The new algorithm is tested over a large set of 1-D, 2-D and 3-D BPCA rules. Simulation results demonstrate the efficiency of the new method.

1 Introduction

Probabilistic Cellular Automata (PCA), which are refered to as Stochastic Cellular Automata (SCA) by some authors, are constructed by introducing probabilistic elements into deterministic local CA rules. The probabilistic elements are generally regarded as a form of noise, which unlike the classical definition of noise in other systems, is essential in investigating the dynamical behavior of PCA's. Deterministic CA's can have a large number of attractors, but the inclusion of noise can cause jumps between attractors and leads to the selection of a small number of physical states [1]. Noise also plays an important role in phase transitions when CA's are employed as a modelling tool to approximate both equilibrium and non-equilibrium systems.

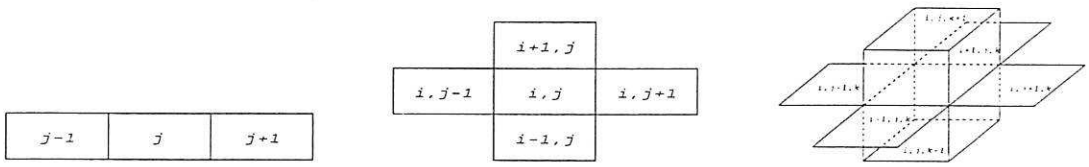
PCA's have been widely studied in recent years. The combined simplicity of PCA rules together with the rich dynamical behavior exhibited in the spatio-temporal patterns produced in the evolution of these systems has attracted the attention of many researchers. This has made the PCA a prototype in the study and testing of certain aspects of complex systems including oscillations in reaction-diffusion processes [2], population growth [3] and the spread of damage [4]. However, a review of the literature shows that the study of PCA has largely been focused on simulating dynamical systems [5], [6], [7], [8] and investigations of the dynamical behavior revealed by PCA models [9], [10], [11], [12], [13].

But the important problem of identification of PCA rules from given patterns of data seems to have been largely ignored.
The identification problem consists of determining the probabilistic local transition rules and the associated neighbourhood over which the rule is operated, from a given set of spatio-temporal patterns generated by the PCA evolution. The identified PCA rule should be parsimonious so that the rule set is as small as possible and the size of the neighbourhood is minimal. Only a few authors have studied this problem. In [14] a genetic algorithm was designed in search for an appropriate probabilistic CA rule through a space of possible PCA's constructed over a number of dimensions. However, the neighbourhood selection process was complicated and it was unclear if the neighbourhood obtained was minimal. Both sequential and parallel algorithms were introduced in [15] for the identification of PCA's. The neighbourhood was found by incrementing the radius by one at each iteration until a pre-formulated condition was satisfied. Although this produced small neighbourhoods the search process was not very flexible or efficient. The simulation results in [16] suggested that the correct neighbourhood and local transition rules may still be obtained under certain levels of noise when the GA search developed in the paper for deterministic rules was applied. However, the rule space was constructed over the complete assumed neighbourhood and this can involve an exceptionally large number of possible rules, and the search time can be extremely long.
This paper considers Binary Probabilistic Cellular Automata (BPCA) and shows for the first time that a class of BPCA can be described by simple integer-parameterised polynomials corrupted by noise (the probabilistic elements). It is proved that the contribution values for cell terms that define the neighbourhood can be calculated without a knowledge of the noise. This is important because this will allow the neighbourhood detection scheme developed in [19] for deterministic rules to be employed as a preliminary neighbourhood detection tool in the presence of noise. By increasing the cutoff value for the contributions the preliminary neighbourhood detection technique can be used to discard most of the spurious terms included in the selected term set and therefore to produce a much reduced neighbourhood. As a result the number of terms in the candidate term set can be dramatically reduced. A multi-objective genetic algorithm with integer constraints is then introduced to refine the pre-selected neighbourhood to the minimum and to find the polynomial that best represents the BPCA rule with the largest probability. It is shown that the efficiency of the search is considerably improved because the genetic algorithm now only has to search through the reduced candidate term set.
The paper is organized as follows. In Section 2 a class of BPCA's and the polynomial representation are introduced. Section 3 discusses the preliminary neighbourhood selection process. A multi-objective GA with integer constraints is constructed in Section 4 and Section 5 provides the simulation results and some discussions. Section 6 contains the conclusions.

2 Probabilistic Cellular Automata

A binary probabilistic cellular automaton comprises a lattice of cells, each taking only 0 or 1 as the state, and a probabilistic local transition rule which specifies at any discrete time

step the state of a cell as a function of the states in previous time steps of the cells within a given neighbourhood. Some examples of PCA neighbourhoods are shown in Figure 1. Note that these neighborhoods only involve cells from time step $t-1$ although BPCA neighborhoods can take cells from various spatial and temporal scales. For simplicity, this paper only considers neighborhoods composed of cells from time step $t-1$, but the results are not restricted to this case.



(a) (b) (c)

Figure 1: Some examples of PCA neighbourhood (a) 1-D von Neumann neighbourhood (b) 2-D von Neumann neighbourhood (c) a 3-D neighbourhood

The probabilistic local rule is composed of 2^n (n is the size of the neighbourhood) rule components, where each represents a possible state of the neighbourhood. The probabilistic rule is constructed by specifying one or more (not all) of the rule components to be 1 or 0 with probability p (denoted as $1/p$ or $0/p$ respectively), and 0 or 1 with probability $(1-p)$ (denoted as $0/(1-p)$ or $1/(1-p)$ respectively) while the other components are deterministic (0 only or 1 only). Varying the probability p between 1 and 0 leads to a transition from one deterministic rule (corresponding to p=1) to another rule (corresponding to p=0). A typical rule of a 1-D 3-site BPCA with von Neumann neighbourhood is shown in Table 1.

Table 1. An example of a 1-D BPCA

$t-1$:	000	001	010	011	100	101	110	111
t :	0	0	$1/p$ $0/(1\text{-}p)$	1	$1/p$ $0/(1\text{-}p)$	1	0	0

This probabilistic rule causes a transition from *Rule*60 (following Voorhees' nomenclature scheme [17]) to *Rule*40. It can be seen that the noise is added to *Rule*60 through making rule components 010 and 100 dependent on the probability parameter p. For *Rule*60, the states that are governed by these two rule components are no longer exclusively updated to 1 at time step t but may be flipped to 0 with probability $1-p$. These flipped states represent the noise.

3 Preliminary Neighbourhood detection

Determining the neighbourhood which defines the spatial and temporal connections that specify the CA rule is an important first step in CA identification. Even complex patterns

can be generated from simple neighbourhoods and it is important to develop procedures that can identify parsimonious CA model forms from CA pattern data. This problem will be addressed in the following sections by introducing an algorithm which shows the contribution that each term makes to the CA rule.

3.1 Representations of BPCA

Every deterministic binary CA rule with a neighbourhood $\{cell(x_1), \cdots, cell(x_n)\}$ of size n can be expressed by a Boolean function of the form [16]

$$s_d(x_j) = a_0 \oplus a_1 s(x_1) \oplus \cdots \oplus a_V (s(x_1) * \cdots * s(x_n)) \tag{1}$$

where $V = 2^n - 1$, x_j is the cell to be updated, $s(x_i)$ is the state of $cell(x_i)$ at time step $t-1$, $s_d(x_j)$ is the next state in $cell(x_j)$ at time step t, and the subscript d is used to indicate the rule is deterministic. a_i $(i = 0, \cdots, V)$ 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 $*$ denote XOR and AND operators respectively. The XOR and AND operators can be represented using the normal algebraic plus $+$ and multiplication $\times$ operators to give

$$h_1 * h_2 = h_1 \times h_2, \quad h_1 \oplus h_2 = h_1 + h_2 - 2h_1 \times h_2$$

Applying this to equation (1) shows that every binary deterministic CA rule with a neighbourhood $\{cell(x_1), \cdots, cell(x_n)\}$ can be represented by a polynomial of the form

$$s_d(x_j) = \theta_1 s(x_1) + \cdots \theta_n s(x_n) + \cdots + \theta_V s(x_1) \times \cdots \times s(x_n) \tag{2}$$

where $V = 2^n - 1$ and θ_i $(i = 1, \cdots, V)$ are integers.
According to the definition, a BPCA can be represented by

$$s_p(x_j) = \begin{cases} s_d(x_j) & \text{with probability } 1 - p_1 \\ 1 - s_d(x_j) & \text{with probability } p_1 \end{cases} \tag{3}$$

where $p_1 = (1 - p) \times p_2$ $(p > 0.5)$, p_2 is the probability with which the probabilistic components will appear in the spatio-temporal pattern and the p subscript is used to indicate that the rule is defined in terms of probabilities. p_1 is related to and smaller than p.
Equation (3) shows that a probabilistic rule is equivalent to the deterministic rule defined by specifying all the probabilistic components to be 1, with a probability $1 - p_1$ and the conjugate deterministic rule constructed by assigning 0 to all the probabilistic components, with a probability p_1. Since $s_p(x_j)$, $s_d(x_j)$ and $s(x_i)$ $(i = 1, \cdots, n)$ are all temporally dependent, the probabilistic rule can also be viewed as a binary time series defined by (2) where the data is corrupted by a time dependent noise signal which occasionally flips the updated state. To make later discussions clearer, temporal symbols are introduced into the notation and $s_p(x_j)$, $s_d(x_j)$ and $s(x_i)$ $(i = 1, \cdots, n)$ can then be denoted as $s_p(x_j; t)$, $s_d(x_j; t)$ and $s(x_i; t-1)$ $(i = 1, \cdots, n)$ respectively.

The noise signal is defined as

$$\eta(t) = s_p(x_j; t) - s_d(x_j; t) \tag{4}$$

So that substituting (3) into (4) and using the new notation above yields

$$\eta(t) = \begin{cases} 0 & 1 - p_1 \\ 1 - 2s_d(x_j; t) & p_1 \end{cases} \tag{5}$$

It can be seen that $\eta(t)$ is a signal with only three states -1, 0 and 1 and the nonzero states appear with probability p_1.
The statistics of the noise signal are unknown and difficult to predict since the occurrence of the nonzero states is dependent on the initial conditions and the evolution of the BPCA. The noise signal can therefore be assumed to be nonstationary and nonlinear. According to equation (26.3) in [18], $\eta(t)$ can be expanded as

$$\eta(t) = \sum_{i=1}^{m_\eta} g_i(\xi(t-1), \cdots, \xi(t-n_\eta)) \times \beta_i(t) + \xi(t) \tag{6}$$

where the $g_i()$'s $(i = 1, \cdots, m_\eta)$ are a set of nonlinear functions and $\xi(t)$ is a white noise sequence. The nonstationary nature of $\eta(t)$ is denoted by the temporal dependence of the parameters β_i $(i = 1, \cdots, m_\eta)$.
From the discussions above, every binary probabilistic CA rule with a neighbourhood $\{cell(x_1), \cdots, cell(x_n)\}$ can be represented by

$$s_p(x_j; t) = \theta_1 s(x_1; t-1) + \cdots \theta_n s(x_n; t-1) + \cdots + \theta_V s(x_1; t-1) \times \cdots \times s(x_n; t-1) + \eta(t) \tag{7}$$

where $V = 2^n - 1$ and θ_i $(i = 1, \cdots, V)$ are integers.
Before moving on to the next section, the relationships between a cell. a neighbourhood, a term and a CA rule need to be clarified. A neighbourhood is composed of one or more cells. For example, the 2-D von Neumann neighbourhood in Figure 1 (b) consists of five cells, $cell(i+1, j)$, $cell(i, j-1)$, $cell(i, j)$, $cell(i, j+1)$ and $cell(i-1, j)$. An assumed neighbourhood normally includes other cells that are not within the real neighbourhood to be identified. A term is a product of the states of one or more cells within a neighbourhood. A CA rule is constructed over one or more terms. The correct term set for a CA rule is the collection of only those terms that define the CA rule. For example, 1-D $Rule60$ over the von Neumann neighbourhood can be written as

$$s_d(j; t) = s(j-1; t-1) + s(j; t-1) - 2 \times s(j-1; t-1) \times s(j; t-1)$$

The correct term set for this rule therefore consists of three terms $s(j-1: t-1)$, $s(j; t-1)$ and $s(j-1; t-1) \times s(j.: t-1)$.

3.2 Orthogonalization and the noise model

Equation (2) can also be expressed as

$$s_d(x_j;t) = \mathbf{s}(t-1) \times \bar{\theta} \tag{8}$$

where

$$\bar{\theta} = \left[\begin{array}{cccc} \theta_1 & \theta_2 & \cdots & \theta_V \end{array} \right]^T$$

and

$$\mathbf{s}(t-1) = \left[\begin{array}{cccc} s(x_1;t-1) & \cdots & s(x_n;t-1) & \cdots \quad s(x_1;t-1) \times \cdots \times s(x_n;t-1) \end{array} \right]$$

Or in matrix form

$$\mathbf{s}_d = \mathbf{S} \times \bar{\theta} \tag{9}$$

where

$$\mathbf{s}_d = \left[\begin{array}{cccc} s_d(x_j;1) & s_d(x_j;2) & \cdots & s_d(x_j;N) \end{array} \right]^T$$

$$\mathbf{S} = \left[\begin{array}{ccccc} \mathbf{s}^T(0) & \mathbf{s}^T(1) & \cdots & \mathbf{s}^T(N-1) \end{array} \right]^T = \left[\begin{array}{ccc} \mathbf{s}_1 & \cdots & \mathbf{s}_V \end{array} \right]$$

Matrix $\mathbf{S}$ can be decomposed as $\mathbf{S} = \mathbf{E} \times \mathbf{Q}$, where

$$\mathbf{E} = \left[\begin{array}{ccc} e_1(0) & \cdots & e_V(0) \\ \vdots & & \vdots \\ e_1(N-1) & \cdots & e_V(N-1) \end{array} \right] = \left[\begin{array}{ccc} \mathbf{e}_1 & \cdots & \mathbf{e}_V \end{array} \right]$$

is an orthogonal matrix.

$$\mathbf{E}^T \times \mathbf{E} = Diag \left[\begin{array}{ccc} \mathbf{e}_1^T \times \mathbf{e}_1 & \cdots & \mathbf{e}_V^T \times \mathbf{e}_V \end{array} \right]$$

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

$$\mathbf{Q} = \left[\begin{array}{ccccc} 1 & q_{12} & q_{13} & \cdots & q_{1V} \\ & 1 & q_{23} & \cdots & q_{2V} \\ & & \ddots & \ddots & \vdots \\ & & & 1 & q_{V-1V} \\ & & & & 1 \end{array} \right]$$

Equation (9) can then be represented as

$$\mathbf{s}_d = \mathbf{E} \times \mathbf{Q} \times \bar{\theta} = \mathbf{E} \times \tilde{\theta} \tag{10}$$

where $\tilde{\theta} = \mathbf{Q} \times \bar{\theta} = [\tilde{\theta}_1 \cdots \tilde{\theta}_V]^T$.

Equation (2) can therefore be expressed as

$$s_d(x_j;t) = \sum_{i=1}^{V} e_i(t-1) \times \tilde{\theta}_i \tag{11}$$

Substituting (11) into (7) gives

$$s_p(x_j;t) = \sum_{i=1}^{V} e_i(t-1) \times \tilde{\theta}_i + \eta(t) \tag{12}$$

The neighbourhood detection algorithm should be designed to select the correct neighbourhood $\{cell(x_1), \cdots, cell(x_n)\}$ from an initial large neighbourhood $\{cell(x_1), \cdots, cell(x_n), cell(x_{n+1}), \cdots, cell(x_m)\}$ of size m. The algorithm proposed in [19] selects the relevant V_r ($V_r \leq V$) terms from the initial term set $\{\mathrm{s}_i,\ i = 1, \cdots, V_m\}$ ($V_m = 2^m - 1$) by calculating the contribution

$$[ct]_i = \frac{\tilde{\theta}_i^2 \times \mathbf{e}_i^T \times \mathbf{e}_i}{\mathbf{s}_p^T \times \mathbf{s}_p} \tag{13}$$

each term s_i in equation (9) makes to s_p. This guarantees that all the correct terms are in the assumed term set and without the corruption of noise, is able to select a correct set of terms that represent the rule and cover the correct neighbourhood. However, terms not included in the correct term set but constructed over cells within the neighbourhood and terms constructed over cells out of the neighbourhood may also be chosen if noise is introduced (see the simulated examples in Section 5). This means that the effects of the noise can be restrained to two parts: inclusion of terms out of the correct term set but constructed over cells within the real neighbourhood and terms constructed over cells within the assumed neighbourhood but out of the real neighbourhood. Following an analogous procedure as above but now for $\eta(t)$ yields

$$\eta(t) = \sum_{i=1}^{V_w} e_i^w(t-1) \times \tilde{\theta}_i^w(t) + \sum_{i=1}^{V_o} e_i^o(t-1) \times \tilde{\theta}_i^o(t) + \xi(t) \tag{14}$$

where the first and second sum represent the orthogonalized noise terms constructed over cells within and out of the real neighbourhood respectively. The nonstationary nature of $\eta(t)$ exhibited in $\beta_i(t)$ in equation (6) is now expressed by the time dependence of $\tilde{\theta}_i^w(t)$ and $\tilde{\theta}_i^o(t)$.

3.3 The effects of noise on term contribution

The term contribution defined in equation (13) is important in the term selection process. For deterministic rules the correct term set can be selected by calculating the contributions $[ct]_i$. In this subsection the effects of noise on term contribution will be discussed.
Although all the terms of $e_i(t-1) \times \tilde{\theta}_i$ ($i = 1, \cdots, V$) are present in equation (12) assume that only $V_r \leq V$ define the correct BPCA model. The irrelevant terms are assumed to

correspond to θ_i's which are zero in equation (7). Therefore equation (12) can be rewritten as

$$s_p(x_j;t) = \sum_{i=1}^{V_r} e_i(t-1) \times \tilde{\theta}_i + \eta(t) \tag{15}$$

Squaring both sides of equation (15) and taking the expected value gives

$$E[s_p^2(x_j;t)] = E[(\sum_{i=1}^{V_r} e_i(t-1) \times \tilde{\theta}_i)^2] + 2 \times E[(\sum_{i=1}^{V_r} e_i(t-1) \times \tilde{\theta}_i) \times \eta(t)] + E[\eta^2(t)] \tag{16}$$

Because the $e_i(t-1)$'s are orthogonal, $e_i(t-1) \times e_j(t-1) = 0$ $(i \neq j)$, and

$$E[(\sum_{i=1}^{V_r} e_i(t-1) \times \tilde{\theta}_i)^2] = E[\sum_{i=1}^{V_r} e_i^2(t-1) \times \tilde{\theta}_i^2] \tag{17}$$

Substituting (17) into (16) yields

$$\frac{1}{N}\sum_{t=1}^{N} s_p^2(x_j;t) - \sum_{i=1}^{V_r} \frac{1}{N}\sum_{t=1}^{N} e_i^2(t-1) \times \tilde{\theta}_i^2 = \gamma \tag{18}$$

where

$$\gamma = 2 \times E[(\sum_{i=1}^{V_r} e_i(t-1) \times \tilde{\theta}_i) \times \eta(t)] + E[\eta^2(t)] \tag{19}$$

Replacing $\eta(t)$ in (19) by (14)

$$\begin{aligned}
\gamma &= 2 \times E[(\sum_{i=1}^{V_r} e_i(t-1) \times \tilde{\theta}_i) \times (\sum_{i=1}^{V_w} e_i^w(t-1) \times \tilde{\theta}_i^w(t) + \sum_{i=1}^{V_o} e_i^o(t-1) \times \tilde{\theta}_i^o(t) + \xi(t))] \\
&\quad + E[(\sum_{i=1}^{V_w} e_i^w(t-1) \times \tilde{\theta}_i^w(t) + \sum_{i=1}^{V_o} e_i^o(t-1) \times \tilde{\theta}_i^o(t) + \xi(t))^2] \\
&= 0 + 0 + 0 + E[\sum_{i=1}^{V_w} (e_i^w(t-1) \times \tilde{\theta}_i^w(t))^2 + \sum_{i=1}^{V_o} (e_i^o(t-1) \times \tilde{\theta}_i^o(t))^2 + \xi^2(t)] \\
&= \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{V_w} (e_i^w(t-1) \times \tilde{\theta}_i^w(t))^2 + \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{V_o} (e_i^o(t-1) \times \tilde{\theta}_i^o(t))^2 + \frac{1}{N}\sum_{t=1}^{N} \xi^2(t)
\end{aligned}$$

Substituting γ back into (18) produces

$$\begin{aligned}
1 - \frac{\sum_{i=1}^{V_r} \frac{1}{N}\sum_{t=1}^{N} e_i^2(t-1) \times \tilde{\theta}_i^2}{\frac{1}{N}\sum_{t=1}^{N} s_p^2(x_j;t)} - \frac{\sum_{i=1}^{V_r} \frac{1}{N}\sum_{t=1}^{N} (e_i^w(t-1) \times \tilde{\theta}_i^w(t))^2}{\frac{1}{N}\sum_{t=1}^{N} s_p^2(x_j;t)} &= \frac{\sum_{i=1}^{V_r} \frac{1}{N}\sum_{t=1}^{N} (e_i^o(t-1) \times \tilde{\theta}_i^o(t))^2}{\frac{1}{N}\sum_{t=1}^{N} s_p^2(x_j;t)} \\
&\quad + \frac{1}{N}\sum_{t=1}^{N} \xi^2(t)
\end{aligned} \tag{20}$$

From the definition of $[ct]$ in (13), equation (20) can finally be expressed as

$$1 - \sum_{i=1}^{V_r} [ct]_i^r - \sum_{i=1}^{V_w} [ct]_i^w = \sum_{i=1}^{V_o} [ct]_i^o + \frac{\delta_\xi^2}{\delta_{s_p}^2} \tag{21}$$

where the subscripts and superscripts r, w and o are used to indicate the terms that are in the real term set, the noise terms that are only related to cells within the correct neighbourhood and the noise terms that are related to both cells within and out of the correct neighbourhood. Equation (21) implies that the $[ct]$ values for terms that are in the correct term set ($[ct]^r$) and noise terms that are out of the correct term set but constructed over cells within the correct neighbourhood ($[ct]^w$) can be calculated independently of the noise terms that are constructed over cells out of the correct neighbourhood ($[ct]^o$). This suggests that the neighbourhood detection algorithm in [19] can still be employed to detect the neighbourhood. The problem is the cutoff point C_{off} because any noise term which has a $[ct]^o$ value larger than $\frac{\delta_\xi^2}{\delta_{s_p}^2}$ will be incorrectly included if C_{off} is still set to 0 as in [19]. Ideally C_{off} should be set to $\sum_{i=1}^{V_o} [ct]_i^o + \frac{\delta_\xi^2}{\delta_{s_p}^2}$ and learned online in order to determine the appropriate cutoff and hence the exact correct term set. However, because the statistics of $\eta(t)$ are unknown and the signal:noise ratio is 100 percent occurring with probability p_1, which makes it difficult to use conventional methods to minimise the effects of the noise, it is not easy to calculate C_{off} online. But equation (21) is still a valuable result because even if the correct cutoff cannot be easily found the application of $[ct]$ can still be employed to eliminate many inappropriate terms. However the test results in Section 5.1 suggest that the correct or almost correct neighbourhood can still be found if C_{off} is chosen within the range $[0.05, 0.1]$.

4 Rule selection using multi-objective GA's with integer constraints

The full neighbourhood search can be dramatically reduced by examining the terms selected using $[ct]$ in Section 3 above. Denote the selected terms as $\{sl_1, \cdots, sl_{V_u}\}$ and the associated neighbourhood as $\{cell(x_1), \cdots, cell(x_u)\}$ ($V_u \leq 2^u - 1$). The deterministic rule which is equivalent to the probabilistic rule with probability $1 - p_1$ can then be assumed as

$$s_d(x_j) = \theta_1 sl_1 + \cdots + \theta_{V_u} sl_{V_u} \tag{22}$$

where θ_i ($i = 1, \cdots, V_u$) are integers.

Although the size of the assumed neighbourhood and the number of terms in the assumed polynomial have both been considerably reduced, the noise effect is not completely eliminated when using data extracted from BPCA patterns to determine the polynomial model in (22). A further noise reduction technique is therefore required.

Equation (2) shows that the parameters in the polynomial model of a CA must be integers. In fact a large number of simulation tests suggest that the parameters in the polynomial model of a CA are often integers within a finite range, for example $[-6, 6]$. This suggests

that it may be possible to find the correct equivalent deterministic rule from the noise contaminated data by constraining the parameters to be integers within a limited range. However most of the available optimization methods treat the variables as continuous and are therefore not appropriate when the parameters are integers. Although some of these algorithms may produce integer solutions by first solving the continuous problem and then employing round-off techniques the solutions may be far from optimal. The optimal integer solution can only be obtained by an exhaustive search. However, this is impractical due to time and memory constraints even for small scale problems. Various methods ([20], [21], [22]) have been designed to solve the integer optimization problem, but each has drawbacks including low efficiency, limited reliability and becoming trapped at a local optimum. Genetic Algorithms (GA) however seem to be appropriate and allow integer constraints to be added to polynomial rule selection. GA's will therefore be briefly introduced in the following sections.

4.1 Population

The objective of the GA search is to select the appropriate terms from the reduced term set, which has been determined using contribution $[ct]$ in Section 3, and to determine the associated integer parameters so that the resulting polynomial represents a parsimonious model representation of the BPCA pattern.
Each term and associated parameter is represented by a $1 \times \mu$ binary vector. The length μ is determined by the range of integer parameters which define the search space. For example for $\mu = 4$ the mapping between the $1 \times \mu$ binary vectors and the corresponding integers is shown in Table 2. The leading 0 in the binary vectors denotes plus and the leading 1 means negative.

Table 2. The relationship between the binary vectors and the corresponding integers

Binary vectors	Integers	Binary vectors	Integers
[0 0 0 0]	0	[1 0 0 0]	-0
[0 0 0 1]	1	[1 0 0 1]	-1
[0 0 1 0]	2	[1 0 1 0]	-2
[0 0 1 1]	3	[1 0 1 1]	-3
[0 1 0 0]	4	[1 1 0 0]	-4
[0 1 0 1]	5	[1 1 0 1]	-5
[0 1 1 0]	6	[1 1 1 0]	-6
[0 1 1 1]	7	[1 1 1 1]	-7

For integer polynomial parameters of CA's with a dimension under 4 our results suggest that a range of $[-6, 6]$, $\mu = 4$ is large enough. If higher-dimensional or more complex BPCA's are involved, μ may need to be increased to give $2^{\mu-1} \geq max - min$, where max and min represent the largest and the smallest possible integer parameters respectively. The length λ of the binary GA chromosome can therefore be computed from $\lambda = \mu \times V_u$.

The ith GA chromosome c_i will be defined as a $1 \times \lambda$ binary vector so that starting from the first bit, every μ bits in the chromosome represent an integer parameter and correspond to a term in equation (22):

$$c_i(1:\mu) \rightarrow sl_1,\ c_i(\mu+1:2\mu) \rightarrow sl_2,\ \cdots,\ c_i((V_u-1)\mu+1:V_u\mu) \rightarrow sl_{V_u}$$

If the jth parameter θ_j $(j = 1, \cdots, V_u)$ which is represented by $c_i(j\mu+1:(j+1)\mu)$ is identified as zero, then the corresponding term sl_j is not included in Equation (22). Define

$$f = \left[\begin{array}{cccc} sl_1 & sl_2 & \cdots & sl_{V_u} \end{array}\right]$$

$$C = \left[\begin{array}{cccc} c_1 & c_2 & \cdots & c_{np} \end{array}\right]^T$$

$$D = de(C) = \left[\begin{array}{cccc} d_1 & d_2 & \cdots & d_{np} \end{array}\right]^T$$

where np is the population size and de is the decoding function which maps the binary vectors back to integers according to Table 1. The whole population C is initialized by assigning each chromosome as a randomly generated binary vector with λ bits.

4.2 Multi-objective fitness function

The fitness function is designed to measure the performance of polynomial rules represented by the chromosomes in regenerating the observed spatio-temporal patterns. An important measure in the present problem is the modulus of errors function defined as $Mer(i) = \sum_r^{SET} |y(i,j) - \hat{y}(i,j)|$, where r is the number of data points in the data set extracted from the BPCA patterns, $y(i,j)$ is the original measured state at data point j for chromosome i and $\hat{y}(i,j) = d_i \times f_j$ is the predicted state.
If Mer is chosen as the fitness function the GA search will find a solution with the least modulus of errors. However it is not guaranteed that the associated neighbourhood is correct and minimal.
The preliminary neighbourhood detection technique in Section 3 produces the reduced neighbourhood $\{cell(x_1), \cdots, cell(x_u)\}$ for the GA search, which should be considerably smaller than the original neighbourhood $\{cell(x_1), \cdots, cell(x_n), \cdots, cell(x_u), \cdots, cell(x_m)\}$, but this may still be larger than the true neighbourhood $\{cell(x_1), \cdots, cell(x_n)\}$.
Notice that there may be more than one model that produces a minimum modulus of errors. However the principle of parsimony implies that the best model will have the least terms. Therefore another search objective must be added to direct the GA evolution to produce a parsimonious polynomial with minimal modulus of errors.
In the present study the two search objectives are to minimize the modulus of errors and to minimize the number of terms in all models with the same Mer. An efficient way of combining these two search objectives is to construct a multi-objective fitness function based on a ranking scheme according to the concept of Pareto optimality [23]. This will guarantee equal probability of reproduction to all non-dominant chromosomes and should generate a solution nearest to the optimal. The multi-objective fitness function is constructed as follows.

i) For the current population with size np, each chromosome is ranked with respect to Mer. The chromosome with the least error occupies the first position, the chromosome with the second least error occupies the second position and so on. Chromosomes with the same error share the same rank. So that

RANK	1	$\cdots$	i	i	i	$\cdots$	np
ERROR	$Mer(1)$	$\cdots$	$Mer(i)$	$Mer(i+1)$	$Mer(i+2)$	$\cdots$	$Mer(np)$

with

$$Mer(1) < \cdots < Mer(i) = Mer(i+1) = Mer(i+2) < \cdots < Mer(np).$$

ii) Map the binary vectors back to integer parameters using Table 2 and define the structure function $St(i)$ for the ith chromosome as the number of nonzero integers in the chromosome. Resort the orders of chromosomes sharing the same rank in proportion to the associated $St(i)$ and keep the ranking of the remainder unchanged. Thus,

RANK	1	$\cdots$	i	$i+1$	$i+1$	$\cdots$	np
ERROR	$Mer(1)$	$\cdots$	$Mer(i)$	$Mer(i+1)$	$Mer(i+2)$	$\cdots$	$Mer(np)$
STRUCTURE	$St(1)$	$\cdots$	$St(i)$	$St(i+1)$	$St(i+2)$	$\cdots$	$St(np)$

with $St(1) < \cdots < St(i) < St(i+1) = St(i+2) < \cdots < St(p)$.

iii) The multi-objective fitness function of the ith chromosome is finally defined as

$$fit(i) = \frac{MAX(rank(i)) - rank(i)}{MAX(rank(i)) - MIN(rank(i))}$$

To avoid the GA search becoming trapped at a local optima two subpopulations will be introduced which evolve in parallel with the main population [24]. The subpopulations are evolved separately under two different search objectives. One is to minimize the modulus of errors Mer, the other to minimize the structure function St. The main population will then evolve synchronously with the subpopulations under an objective jointly determined by the two objectives. Each candidate in the main population is produced by genetic communication between the two subpopulations and is subject to evaluation by the ranking technique.

4.3 Reproduction

In the reproduction process, chromosomes are first selected as parents to reproduce offspring according to the corresponding fitness values. The purpose of parent selection is to give more reproductive chances to those chromosomes that are the most fit. This paper uses the roulette wheel parent selection technique from [23]. The selected population is then used for genetic operations in the breeding process. There are two principle genetic

operators for producing new chromosomes during the breeding process. The crossover operator cuts segments from both parents and combines these segments to produce new chromosomes. The mutation operator arbitrarily alters the bits in a chromosome according to a predetermined probability, the mutation rate. See [25] and [26] for details.

The new multi-objective GA search for polynomial rules with integer constraints can be summarized as:

i) Initialize the two subpopulations and the main population on the basis of the preliminary neighbourhood obtained in Section 3.

ii) Evaluate the three populations according to Mer, St, and Mer combined with St respectively using the ranking technique.

iii) Apply the parent selection technique to the two subpopulations.

iv) Employ crossover and mutation to the two subpopulations separately.

v) Employ crossover and mutation to the two subpopulations combined to produce new candidates for the main population.

vi) Repeat (ii) and insert new populations to replace the three old populations respectively.

vii) If all chromosomes in the new main population converge to a single individual then stop, otherwise return to (iii) and repeat.

After the deterministic model which is corrupted by noise, is found the minimal neighbourhood can be retrieved. The probabilistic elements and the associated probability can then easily be found from the data set by collecting a probability table. For every rule component which is determined by the minimal neighbourhood, the occurrences of 0 and 1 in $s_p(x_j;t)$ are recorded. For the deterministic rule components the occurrences of 0 and 1 in $s_p(x_j;t)$ cannot be both nonzero. This probability table represents the identified BPCA.

5 Simulation Studies

5.1 Preliminary Neighbourhood detection

Three examples, for 1-D, 2-D and 3-D BPCA, will be used to demonstrate the preliminary neighbourhood detection and the crucial role the increased C_{off} plays in reducing the number of insignificant terms in the selected term set. The initial neighbourhoods used in the three examples are defined in Table 3. To simplify the notation all the neighbourhoods are assumed as to be 9-site neighbourhoods. The candidate term set SET which is determined from the initial neighbourhoods is therefore the same for all the three examples, and is 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 cells in the assumed neighbourhoods. The mappings between entries in SET and cells in the neighbourhoods are illustrated in Table 3. For example, entry 5 is associated with $cell(j+2)$ in Example 1, $cell(i-1, j)$ in Example 2 and $cell(i, j-1, k)$ in Example 3, and so on. The whole SET consists of $2^9 - 1 = 511$ rows. Each row represents a candidate term which corresponds to an $\mathbf{s}_i, i = 1, \cdots, V$ in matrix $\mathbf{S}$ in equation (9). For example, the first row (1 0 0 0 0 0 0 0 0) represents only $s(j)$ in Example 1, $s(i, j)$ in Example 2, and $s(i, j, k)$ in Example 3, while the last row (1 2 3 4 5 6 7 8 9) corresponds to a product of nine states, $s(j) \times s(j-1) \times s(j+1) \times s(j-2) \times s(j+2) \times s(j-3) \times s(j+3) \times s(j-4) \times s(j+4)$ in Example 1, $s(i,j) \times s(i+1,j) \times s(i,j-1) \times s(i,j+1) \times s(i-1,j) \times s(i+1,j-1) \times s(i+1,j+1) \times s(i-1,j-1) \times s(i-1,j+1)$ in Example 2, and $s(i,j,k) \times s(i+1,j,k) \times s(i-1,j,k) \times s(i,j+1,k) \times s(i,j-1,k) \times s(i,j,k+1) \times s(i,j,k-1) \times s(i-1,j+1,k) \times s(i-1,j-1,k)$ in Example 3. Note that an entry in SET denotes a cell while a row in SET denotes the product of the states of one or more cells, that is, a candidate term $\mathbf{s}_i, i = 1, \cdots, V$.

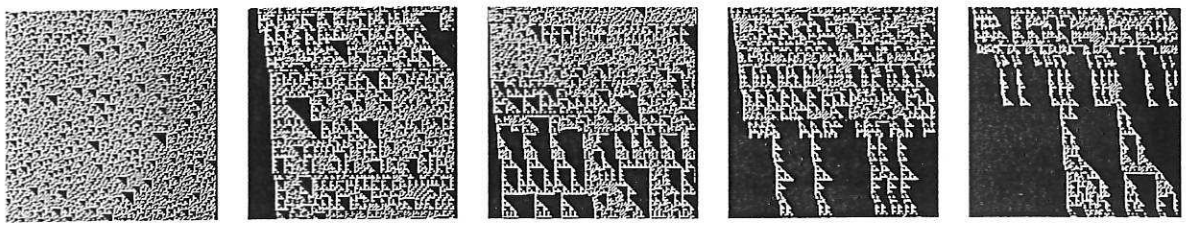
Table 3. The initial neighbourhoods used in Examples 1, 2, and 3

Example	Cells in the initial neighbourhood				
	1	2	3	4	5
1	$cell(j)$	$cell(j-1)$	$cell(j+1)$	$cell(j-2)$	$cell(j+2)$
2	$cell(i,j)$	$cell(i+1,j)$	$cell(i,j-1)$	$cell(i,j+1)$	$cell(i-1,j)$
3	$cell(i,j,k)$	$cell(i+1,j,k)$	$cell(i-1,j,k)$	$cell(i,j+1,k)$	$cell(i,j-1,k)$

Examples	Cells in the initial neighbourhood			
	6	7	8	9
1	$cell(j-3)$	$cell(j+3)$	$cell(j-4)$	$cell(j+4)$
2	$cell(i+1,j-1)$	$cell(i+1,j+1)$	$cell(i-1,j-1)$	$cell(i-1,j+1)$
3	$cell(i,j,k+1)$	$cell(i,j,k-1)$	$cell(i-1,j+1,k)$	$cell(i-1,j-1,k)$

5.1.1 Example 1

Example 1 uses the 1-D BPCA rule given in Table 1. This rule is equivalent to $Rule60$ occurring with probability p and $Rule40$ occurring with probability $1-p$. The spatio-temporal patterns produced by the evolution of this BPCA rule with varying p are shown in Figure 2. All the patterns 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.



$(a)p = 1$ $(b)p = 0.9$ $(c)p = 0.85$ $(d)p = 0.8$ $(e)p = 0.7$

Figure 2: Spatio-temporal patterns produced from Example 1 for the evolution of a 1-D BPCA with varying p

Figure 2 (a) shows the evolution of the deterministic *Rule*60 (because $p = 1$). Figure 2 (b)-(e) can be considered as patterns produced by the evolution of *Rule*60 with noise of various levels defined by the probability $1 - p$. It can be seen that the introduction of noise/probabilistic elements results in considerable changes in the patterns, from a distribution of triangles in Figure 2 (a) to a random tree structure in Figure 2 (e) where the noise level is $1 - 0.7 = 0.3$.

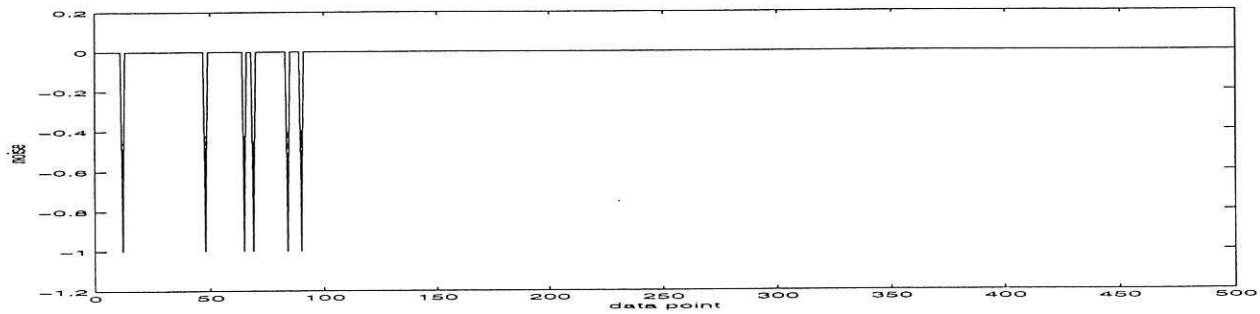


Figure 3: The noise signal generated by flipping the states of the updated cells governed by rule components 010 and 100 in *Rule*60 from 1 to 0 with probability $1 - p = 0.3$

The noise signal at level $1 - p = 0.3$ is shown in Figure 3. The noise is introduced by flipping the states of the updated cells governed by rule components 010 and 100 in *Rule*60 from 1 to 0 and therefore only takes two values, -1 and 0. Although the nonzero values appear with a low occurrence, 6 out of a possible 500, the impact is by no means trivial as can be seen in the pattern changes in Figure 2. This is because the signal/noise ratio is 100 percent at each point the noise appears.
Data extracted from Figure 2 (a) was used for detecting the term set of the deterministic rule. C_{off} was chosen as 0 and the result is shown in model $1-(a)$. The last entry in each row represents the normalized value of the contribution the term in that row makes to s_p. The sum of all the $[ct]$ values will be unity so if $[ct]$ were multiplied by 100 this would give the percentage contribution the term in that row makes to s_p. The same also applies to the other models. In model $1-(a)$ only 3 terms have been selected from the SET of 511 terms and the neighbourhood that are determined by these terms is $\{cell(j), cell(j-1)\}$,

which is the same as listed in Appendix 1 in [17] and is minimal and correct.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1295 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0562 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0440 \end{bmatrix}$$

Model $1-(a)$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3600 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1626 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3600 \\ 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0079 \\ 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0029 \\ 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0044 \\ 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0029 \end{bmatrix}$$

Model $1-(c)$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3600 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1626 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3600 \end{bmatrix}$$

Model $1-(d)$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3600 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1626 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3600 \\ 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0029 \\ 1 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0061 \\ 1 & 3 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0.0042 \\ 1 & 2 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0046 \\ 1 & 3 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0.0030 \\ 2 & 6 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0036 \\ 2 & 6 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0.0103 \\ 2 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0027 \\ 1 & 2 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0011 \\ 1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0029 \\ 1 & 5 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0.0026 \\ 1 & 4 & 5 & 0 & 8 & 0 & 0 & 0 & 0 & 0.0018 \\ 1 & 3 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0028 \\ 2 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0038 \\ 1 & 5 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0022 \\ 1 & 3 & 5 & 8 & 0 & 0 & 0 & 0 & 0 & 0.0029 \\ 2 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0007 \\ 2 & 8 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0009 \\ 1 & 2 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0008 \\ 1 & 6 & 8 & 9 & 0 & 0 & 0 & 0 & 0 & 0.0015 \end{bmatrix}$$

Model $1-(b)$

Data extracted from Figure 2 (e) was used in determining the neighbourhood when noise was introduced at a level defined by $1-p=0.3$. Model 1-(b) shows the result from this data set when C_{off} is set as 0. Although only 23 out of 511 terms have been selected, the neighbourhood covered by these terms is exactly the same as the 9-site neighbourhood assumed and little can be gained from this result. Model $1-(c)$ shows the chosen terms when C_{off} is chosen as 0.1. The neighbourhood determined by these terms is $\{cell(j), cell(j-1), cell(j+1), cell(j-3)\}$, which fully covers the correct neighbourhood but is much smaller than the initial neighbourhood. When C_{off} is increased to 0.2, the three terms selected in model $1-(d)$ are exactly the same as the correct terms in model $1-(a)$. This suggests that increasing the cutoff value can reduce the number of irrelevant terms included in the identified model and in some cases, for example in model $1-(d)$, can even discard all the spurious terms.

5.1.2 Examples 2 and 3

The data used for preliminary neighbourhood detection in Example 2 was produced by the evolution of a 2-D BPCA which was constructed by specifying the states of the updated cells governed by rule components 0010, 0110, 1000 and 1101 in $Rule24235$ to be 0 with probability p and 1 with probability $1-p$ over the neighbourhood $\{cell(i+1,j), cell(i,j-1), cell(i,j+1), cell(i-1,j)\}$. Model $2-(a)$ shows the result when $p=1$ and C_{off} is set as 0. The 9 terms selected constitute the deterministic $Rule24235$ and referring back to Table 3, determine the correct and minimal neighbourhood. When the noise is introduced by setting p as 0.7, the cutoff value was incremented to decrease the number of spurious terms included in the identified model. The result for $C_{off}=0.09$ is given in model $2-(b)$, where 18 terms were selected. A close inspection shows that these 18 terms cover all the 9 correct terms in model $2-(a)$. The remaining terms are composed of two types, 5 terms made up of cells within the correct neighbourhood and 4 terms made up of cells within the correct neighbourhood but including a cell outside the correct neighbourhood, $cell(i-1,j+1)$.

3	0	0	0	0	0	0	0	0	0.4871
3	5	0	0	0	0	0	0	0	0.0631
2	3	4	5	0	0	0	0	0	0.0363
5	0	0	0	0	0	0	0	0	0.0217
2	5	0	0	0	0	0	0	0	0.0292
3	4	5	0	0	0	0	0	0	0.0481
2	0	0	0	0	0	0	0	0	0.0377
2	3	0	0	0	0	0	0	0	0.0322
2	3	5	0	0	0	0	0	0	0.0199

Model $2-(a)$

3	0	0	0	0	0	0	0	0	0.4961
2	4	0	0	0	0	0	0	0	0.0703
3	5	0	0	0	0	0	0	0	0.0323
9	0	0	0	0	0	0	0	0	0.0289
2	3	4	5	0	0	0	0	0	0.0342
2	5	9	0	0	0	0	0	0	0.0433
3	4	9	0	0	0	0	0	0	0.0462
2	3	4	5	9	0	0	0	0	0.0209
5	0	0	0	0	0	0	0	0	0.0152
2	4	5	0	0	0	0	0	0	0.0381
3	4	5	0	0	0	0	0	0	0.0129
2	5	0	0	0	0	0	0	0	0.0275
2	0	0	0	0	0	0	0	0	0.0215
2	3	4	0	0	0	0	0	0	0.0098
4	0	0	0	0	0	0	0	0	0.0037
2	3	0	0	0	0	0	0	0	0.0020
2	3	5	0	0	0	0	0	0	0.0052
3	4	0	0	0	0	0	0	0	0.0047

Model $2-(b)$

3	0	0	0	0	0	0	0	0	0.6756
2	3	4	5	6	7	0	0	0	0.0384
7	0	0	0	0	0	0	0	0	0.0364
6	7	0	0	0	0	0	0	0	0.0186
6	0	0	0	0	0	0	0	0	0.0227
2	3	5	0	0	0	0	0	0	0.0148
2	3	4	0	0	0	0	0	0	0.0153
2	3	5	6	0	0	0	0	0	0.0189
4	0	0	0	0	0	0	0	0	0.0096
4	6	0	0	0	0	0	0	0	0.0077
2	3	5	7	0	0	0	0	0	0.0160
2	5	6	0	0	0	0	0	0	0.0076
2	4	5	6	0	0	0	0	0	0.0106
2	6	7	0	0	0	0	0	0	0.0150
2	3	6	0	0	0	0	0	0	0.0079
2	3	4	6	7	0	0	0	0	0.0035
2	3	6	7	0	0	0	0	0	0.0045
2	3	5	6	7	0	0	0	0	0.0091
5	6	7	0	0	0	0	0	0	0.0027
4	5	6	0	0	0	0	0	0	0.0079
4	5	0	0	0	0	0	0	0	0.0029
5	7	0	0	0	0	0	0	0	0.0036
3	4	6	0	0	0	0	0	0	0.0031
3	4	5	0	0	0	0	0	0	0.0049
3	5	0	0	0	0	0	0	0	0.0024
3	7	0	0	0	0	0	0	0	0.0025
2	4	5	7	0	0	0	0	0	0.0039
2	3	4	6	0	0	0	0	0	0.0011
2	3	4	5	6	0	0	0	0	0.0022
3	6	0	0	0	0	0	0	0	0.0008
3	5	6	0	0	0	0	0	0	0.0005
3	4	5	6	0	0	0	0	0	0.0004
4	6	7	0	0	0	0	0	0	0.0003
2	4	6	7	0	0	0	0	0	0.0007
4	7	0	0	0	0	0	0	0	0.0006
4	5	6	7	0	0	0	0	0	0.0003
2	3	4	5	0	0	0	0	0	0.0003
2	4	5	0	0	0	0	0	0	0.0007
3	6	7	0	0	0	0	0	0	0.0002
3	5	6	7	0	0	0	0	0	0.0004
4	5	7	0	0	0	0	0	0	0.0002
2	3	0	0	0	0	0	0	0	0.0002
2	3	7	0	0	0	0	0	0	0.0006
2	3	4	5	7	0	0	0	0	0.0003
2	3	4	7	0	0	0	0	0	0.0005
3	4	0	0	0	0	0	0	0	0.0009
3	4	5	7	0	0	0	0	0	0.0005
4	6	7	8	0	0	0	0	0	0.0001
4	5	6	7	8	0	0	0	0	0.0002
3	4	6	7	8	0	0	0	0	0.0002
3	4	5	6	7	8	0	0	0	0.0004
2	4	6	7	8	0	0	0	0	0.0002
2	3	4	6	7	8	0	0	0	0.0004
2	4	5	6	7	8	0	0	0	0.0002

Model $3-(b)$

2	3	4	5	6	7	0	0	0	0.0447
7	0	0	0	0	0	0	0	0	0.0305
6	7	0	0	0	0	0	0	0	0.0317
6	0	0	0	0	0	0	0	0	0.0306
2	3	5	6	7	0	0	0	0	0.0159
4	6	0	0	0	0	0	0	0	0.0105
4	0	0	0	0	0	0	0	0	0.0115
4	5	6	0	0	0	0	0	0	0.0066
2	3	5	0	0	0	0	0	0	0.0102
2	3	6	7	0	0	0	0	0	0.0068
2	3	4	6	7	0	0	0	0	0.0085
2	5	6	0	0	0	0	0	0	0.0048
2	3	5	6	0	0	0	0	0	0.0054
2	3	4	5	0	0	0	0	0	0.0018
2	3	7	0	0	0	0	0	0	0.0037
2	3	6	0	0	0	0	0	0	0.0016
2	3	4	5	6	0	0	0	0	0.0013
2	3	4	5	7	0	0	0	0	0.0020
2	4	5	6	0	0	0	0	0	0.0010
3	4	5	6	0	0	0	0	0	0.0012
2	3	4	7	0	0	0	0	0	0.0007
3	4	5	7	0	0	0	0	0	0.0007
3	5	6	7	0	0	0	0	0	0.0010
2	3	4	0	0	0	0	0	0	0.0008
2	3	5	7	0	0	0	0	0	0.0006
2	3	0	0	0	0	0	0	0	0.0004
2	3	4	6	0	0	0	0	0	0.0006
4	6	7	0	0	0	0	0	0	0.0005
4	7	0	0	0	0	0	0	0	0.0026
4	5	6	7	0	0	0	0	0	0.0008
2	4	6	7	0	0	0	0	0	0.0005
2	4	5	7	0	0	0	0	0	0.0010
2	6	7	0	0	0	0	0	0	0.0003
3	6	7	0	0	0	0	0	0	0.0009
2	4	5	0	0	0	0	0	0	0.0005
3	4	5	0	0	0	0	0	0	0.0020

Model $3-(a)$

Example 3 illustrates the preliminary neighbourhood detection for a 3-D BPCA. This probabilistic rule was created over a 6-site neighbourhood defined by $\{cell(i-1,j,k), cell(i+1,j,k), cell(i,j-1,k), cell(i,j+1,k), cell(i,j,k-1), cell(i,j,k+1)\}$ by specifying the states of the updated cells governed by rule components 000010, 000101, 000111, 001100, 010010, 010101, 010111, 011100, 100001, 100110, 101001, 101111, 110001, 110011, 110111, 111110 in a deterministic rule (01111110 11010111 11100110 11011110 01000100 01011110 01110110 01010110) to be 1 with probability p and 0 with probability $1-p$. Model $3-(b)$ gives the 54 selected terms when $p = 0.7$ and C_{off} was chosen as 0.095. The 54 terms were composed of three parts, 36 correct terms in model $3-(a)$ selected for the deterministic rule when $C_{off} = 0$, 11 terms which are constructed only over the cells within the correct neighbourhood, 7 terms which are composed of cells within the correct neighbourhood but also one cell out of the correct neighbourhood, $cell(i,j+1,k)$.

Both model 2-(b) and model 3-(b) define a neighbourhood larger than the correct neighbourhood. However, the number of candidate terms in both models are considerably reduced compared to the original term set. Only 18 and 54 out of a possible 511 terms are selected in Example 2 and Example 3 respectively. This shows that although it is difficult to calculate the exact cutoff value when noise is present, C_{off} can take a range of values so that most of the spurious terms are excluded. In some cases, for instance in model $1-(d)$ in Example 1, the chosen C_{off} is even capable of selecting only the correct terms and hence the correct and minimal neighbourhood.

Note that within each of models $1-(c)$, $2-(b)$ and $3-(b)$ there is a set of terms which are not in the corresponding correct term set (see models $1-(a)$, $2-(a)$ and $3-(a)$ respectively) but which are constructed over the correct cells (cells within the correct neighbourhood). The inclusion of a term set of this kind demonstrates the probabilistic characteristic of the rule since these terms may well be from other deterministic rules with a similar component structure but with a different p. For instance in Example 1, the result in model $1-(c)$ also includes three terms $s(j-1) \times s(j+1)$, $s(j) \times s(j+1)$ and $s(j-1) \times s(j) \times s(j+1)$ which constitute exactly the deterministic $Rule40$. This is because the BPCA rule is equivalent to $Rule60$ occurring with probability $p = 0.7$ and $Rule40$ occurring with probability $1-p = 0.3$.

These three examples also demonstrate that when using the neighbourhood detection algorithm in [19], the $[ct]$ values for terms that are in the correct term set and noise terms that are out of the correct term set but constructed over cells within the correct neighbourhood can be calculated independently of the $[ct]$ values for noise terms that are constructed over cells out of the correct neighbourhood.

The preliminary neighbourhoods for Examples 1, 2 and 3 can then be retrieved from the selected term set in models $1-(c)$, $2-(b)$ and $3-(b)$ as $\{cell(j-3), cell(j-1), cell(j), cell(j+1)\}$, $\{cell(i+1,j), cell(i,j-1), cell(i,j+1), cell(i-1,j), cell(i-1,j+1)\}$ and $\{cell(i-1,j,k), cell(i-1,j+1,k), cell(i+1,j,k), cell(i,j-1,k), cell(i,j+1,k), cell(i,j,k-1), cell(i,j,k+1)\}$ respectively. These neighbourhoods can be refined using the GA's described in the next section.

5.2 Rule selection using GA's with integer constraints

The terms selected in Section 5.1 will be used as initial term sets for the application of the GA algorithm described in Section 4. The algorithm was tested over a large set of 1-D, 2-D and 3-D BPCA rules with various neighborhoods of randomly chosen radius. A sample of the results is summarized in Table 4.

Table 4. Summary of results obtained in evolving some 1-D, 2-D and 3-D BPCA polynomial rules with various sizes of neighbourhoods using the GA routine of Section 4

D	n	Rule	u	Generations		Modulus of errors		Structure		average
				mean	std.dev.	mean	std.dev.	mean	std.dev.	run time
1	3	*Rule*60	4	18.96	3.54	6	0	3	0	38.56 min.
		*Rule*18	4	22.13	4.25	28	0	6	0	40.02 min.
	4	*Rule*16798	6	104.23	6.89	16	0	15	0	200.09 min.
		*Rule*24235	5	75.48	6.06	35	0	9	0	141.26 min.
	5	$Rule1-5$	5	80.05	7.71	22	0	21	0	147.51 min.
		$Rule1-6$	6	112.92	10.58	15	0	18	0	213.42 min.
	6	$Rule1-7$	7	159.64	12.19	9	0	26	0	289.97 min.
		$Rule1-8$	7	193.79	17.55	32	0	19	0	329.44 min.
	7	$Rule1-9$	7	201.82	15.83	27	0	27	0	364.58 min.
		$Rule1-10$	7	170.37	14.44	10	0	23	0	209.83 min.
	8	$Rule1-11$	8	254.91	19.89	41	0	16	0	470.69 min.
		$Rule1-12$	9	338.26	23.25	13	0	34	0	592.85 min.
	9	$Rule1-13$	9	309.74	30.31	24	0	29	0	552.14 min.
		$Rule1-14$	9	372.55	21.25	11	0	44	0	614.32 min.
2	4	*Rule*16798	6	98.92	10.13	19	0	15	0	187.43 min.
		*Rule*24235	5	72.39	8.57	30	0	9	0	150.06 min.
	5	$Rule2-3$	6	90.21	7.74	8	0	12	0	165.29 min.
		$Rule2-4$	7	173.24	9.83	43	0	11	0	313.80 min.
	6	$Rule2-5$	6	107.38	12.78	26	0	20	0	151.91 min.
		$Rule2-6$	7	166.43	15.69	14	0	25	0	311.29 min.
	7	$Rule2-7$	8	238.56	14.32	5	0	32	0	422.36 min.
		$Rule2-8$	7	153.63	20.59	37	0	24	0	218.07 min.
	8	$Rule2-9$	9	400.07	36.77	20	0	53	0	713.46 min.
		$Rule2-10$	8	251.93	26.60	11	0	28	0	436.83 min.
	9	$Rule2-11$	9	385.28	29.89	18	0	45	0	648.31 min.
		$Rule2-12$	9	360.24	18.72	29	0	31	0	625.03 min.
3	6	$Rule3-1$	7	186.53	10.99	33	0	36	0	313.45 min.
		$Rule3-2$	7	151.38	11.48	17	0	26	0	220.96 min.
	7	$Rule3-3$	9	379.26	13.05	22	0	39	0	639.46 min.
		$Rule3-4$	7	196.44	11.58	14	0	27	0	363.51 min.
	8	$Rule3-5$	8	290.00	14.45	29	0	32	0	537.09 min.
		$Rule3-6$	8	223.76	17.89	23	0	24	0	416.37 min.
	9	$Rule3-7$	9	335.14	20.31	7	0	29	0	589.69 min.
		$Rule3-8$	9	390.56	22.98	25	0	35	0	674.18 min.

D represents the dimension of the rule. n indicates the size of the real neighbourhood. u denotes the size of the preliminarily detected neighbourhood before GA evolution. 100 trials were made for each problem. The "Generations" column indicates the number of generations reached before the solution converged.

For each rule, 100 trials were conducted with different initial populations. The data used for the GA search were extracted from spatio-temporal patterns generated by evolving the BPCA rules constructed by specifying the states of the updated cells governed by a quarter of all the rule components in the associated deterministic rules to be 1 with probability $p = 0.7$ and 0 with probability $1 - p = 0.3$. The numerical labels for these deterministic rules are listed in the "**Rule**"column. Only the deterministic rules with small neighbourhoods will be enumerated. This is due to the fact that the numerical label and the component form of the deterministic rules can be very cumbersome when the neighbourhood size is larger than 4. For simplicity only the average and standard deviation (*std.dev.*) values are listed in Table 4.

Inspection of Table 4 shows that the modulus of errors did not converge to zero because the data used for the GA search are corrupted by probabilistic noise. The number of errors is actually the number of contaminated data points.

Rules in Table 4 are selected from a set of no more than $2^4 \times 2^{2^u-1}$ possible rules. For example, when $u = 4$, for the 1-D 3-site rules in Table 4, the rule set for the GA search with preliminary neighbourhood detection comprises a maximum of $2^4 \times 2^{2^4-1} = 524288$ possible rules. In particular, in Example 1 when C_{off} is chosen as 0.1, the number of possible rules determined by model $1 - (c)$ which is used to prime the GA search is dramatically reduced to only $2^4 \times 2^7 = 2048$. In comparison when $m = 9$ (m is the size of the initial neighbourhood) the rule set for the GA search with no preliminary neighbourhood detection which generates the results in Table 6 in [16] consisted of a massive $2^{2^9} = 1.3408e + 154$ rules. Hence the average run time in Table 4 is considerably smaller than in Table 6 in [16].

The preliminary neighbourhood detection and the GA search with integer constraints are insensitive to the dimensionality of the BPCA rules because what matters is the number of terms searched, not the dimensionality. Identification of rules of the same construction but different dimensions should therefore be able to produce the same St and similar Mer and average run times. These properties are demonstrated in the results for the two 1-D 4-site rules $Rule16798$ and $Rule24235$ and the two 2-D 4-site rules of the same rule number. The slight discrepancy in Mer and the average run time is caused by the different initial conditions and the randomness of genetic operations in the GA evolution.

Each identified polynomial produces a correct truth table that matches the component form of the deterministic rule which represents the probabilistic rule with probability $1 - p_1$. This polynomial also determines a correct and minimal neighbourhood for the corresponding probabilistic rule. The probabilistic rule components and the associated probability can then easily be identified from the data set by collecting a probability table. This is achieved by recording the occurrences of 0 and 1 in $s_p(x_j; t)$ for every rule component which is determined by the identified neighbourhood. This collected probability table represents the identified BPCA. The probability table for the BPCA rule in Example 2 was collected from a data set of 1000 data points and is presented in Table 5. For simplicity the probability tables for other BPCA rules discussed are not included in the paper.

Table 5. The probability table for the BPCA rule in Example 2

$t-1$:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
t:	1	1	$0/\frac{19}{27}$ $1/\frac{8}{27}$	1	0	1	$0/\frac{54}{78}$ $1/\frac{24}{78}$	1	$0/\frac{28}{41}$ $1/\frac{13}{41}$	1	1	1	1	$0/\frac{69}{98}$ $1/\frac{29}{98}$	1	0

Table 5 shows that 0 appeared in the updated cells governed by rule components 0010, 0110, 1000 and 1101 with a occurrence of 19 times out of 27, 54 times out of 78, 28 times out of 41 and 69 times out of 98, and 1 appeared with a occurrence of 8 times out of 27, 24 times out of 78, 13 times out of 41 and 69 times out of 98 respectively while the updated cells governed by the other rule components are either occupied by 0 only or 1 only. This shows that rule components 0010, 0110, 1000 and 1101 are probabilistic and the probability p is approximately 0.7 ($\frac{19}{27} = 0.7037$, $\frac{54}{78} = 0.6923$, $\frac{28}{41} = 0.6829$, $\frac{69}{98} = 0.7041$).

6 Conclusions

Despite the fact that probabilistic cellular automata have been widely used in generating complex spatio-temporal patterns, very few investigators have studied how to identify the PCA rules given only the patterns. A two-step solution to this important problem has been developed in the present study based on a mapping of the PCA rules to a polynomial rule space. It has been shown that a class of binary probabilistic cellular automata can be represented as integer-parameterised polynomials contaminated by noise. On the basis of these polynomials it has then been proved that the contribution values for the correct terms that are related only to the cells within the neighbourhood can be calculated independently of the noise terms that are also associated with cells out of the neighbourhood. This allows the neighbourhood detection technique, originally developed for deterministic rules, to be used to select a preliminary neighbourhood even in the presence of noise by increasing the contribution cutoff value. This preliminary neighbourhood detection stage can yield significant improvements in efficiency by reducing the number of candidate rules from 2^{2^m} to less than $2^{\mu} \times 2^{2^u - 1}$. For example as shown in Example 1 where $m = 9$, $\mu = 4$ and $u = 4$, the number of possible rules are reduced from $1.3408e+154$ to only 2048. However, the choice of an exact contribution cutoff value that discards all of the spurious terms still needs further study. Integer constraints were added to the GA search to restrain the preliminarily selected neighbourhood to the minimum and to direct the search so that the deterministic polynomial rule which represents the probabilistic rule with the largest probability can be retrieved. Several simulated examples of 1-D, 2-D and 3-D PCA rules demonstrated the effectiveness of the new approach.

7 Acknowledgment

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

References

[1] Mayerkress, G. and Kaneko, K. *Spatiotemporal Chaos and Noise*. Journal of Statistical Physics, 1989, vol.54, no.5-6, pp.1489-1508.

[2] Vieira, F.M.C. and Bisch, P.M. *Oscillatory Patterns in An Autocatalytic Reaction Ring Network Simulated by A Probabilistic Cellular Automaton*, Physica A, 1993, vol.199, no.1, pp.40-54.

[3] Boccara, N. etc. *A Probabilistic Automata Network Epidemic Model with Births and Deaths Exhibiting Cyclic Behavior*, Journal of Physics A, 1994, vol.27, no.5, pp.1585-1597.

[4] Hinrichsen, H. etc. *Universality Properties of the Stationary States in the One-dimensional Coagulation-diffusion Model with External Particle Input*, Journal of Statistical Physics, 1997, vol.86, No.5-6, pp.1203-1235.

[5] de Oliverira, M.J. and Satulovsky, J.E. *Renormalization Group of Probabilistic Cellular Automata with One Absorbing State*, Physical Review E, 1997, vol.55, no.6 PtA, pp.6377-6383.

[6] Hiebeler, D. *Stochastic Spatial Models: From Simulations to Mean Field and Local Structure Approximations*, Journal of Theoretical Biology, 1997, vol.187, no.3, pp.307-319.

[7] Wan, H.A. *Modeling Stock Markets by Probabilistic 1-D Cellular Automata*, International Journal of Computer Mathematics, 1994, vol.53, no.3-4, pp.167-176.

[8] Boccara, N. and Cheong, K. *Critical Behavior of A Probabilistic Automata Network SIS Model for the Spread of An Infectious Disease in A Population of Moving Individuals*, Journal of Physics A, 1993, vol.26, no.15, pp.3707-3717.

[9] Kaneko, K. and Akutsu, Y. *Phase Transitions in Two-dimensional Stochastic Cellular Automata*, Journal of Physics A, 1986, vol.19, no.2, pp.L69-L75.

[10] Menyhard, N. *Inhomogeneous Mean-field Approximation for Phase Transitions in Probabilistic Cellular Automata: An Example*, Journal of Physics A, 1988, vol.21, no.5, pp.1283-1292.

[11] Katori, M. and Tsukahara, H. *2-neighbour Stochastic Cellular Automata and Their Planar Lattice Duals*, Journal of Physics A, 1995, vol.28, no.14, pp.3935-3957.

[12] Bhattacharyya, P. *Critical Phenomena in A One-dimensional Probabilistic Cellular Automaton*, Physica A, 1996, vol.234, no.1-2, pp.427-434.

[13] Niels, K. etc. *Phase Transitions in An Elementary Probabilistic Cellular Automaton*. Physica A, 1997, vol.235, no.3-4, pp.473-485.

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

[15] Adamatskii, A.I. *Identification of Probabilistic Cellular Automata* Adamatskii, A.I. Soviet Journal of Computer and Systems Sciences, 1992, vol.30, no.3, pp.118-123.

[16] Y.X.Yang and S.A.Billings *Extracting Boolean Rules from CA Patterns*, submitted to IEEE Trans System Man and Cybernetics.

[17] B.H.Voorhees, *Computational Analysis of One-dimensional Cellular Automata*, World Scientific Series on Nonlinear Science, World Scientific, 1996.

[18] P.Young, "Time Variable and State Dependent Modelling of Non-stationary and Nonlinear Time Series "in *Developments in Time Series Analysis* edited by T.S.Rao, Chapman and Hall, 1993.

[19] Y.X.Yang and S.A.Billings, "Neighbourhood Detection and Rule Selection from Cellular Automata Patterns ", submitted to *IEEE Trans System Man and Cybernetics.*

[20] C.C.Caroe and R.Schultz, "Dual Decomposition in Stochastic Integer Programming", *Operations Research Letters*, vol.24, no.1-2, pp.37-45, 1999.

[21] A.Hassibi and S.Boyd, "Integer Parameter Estimation in Linear Models with Applications to GPS", *IEEE Trans Signal Processing*, vol.46, No.11, pp.2938-2952, 1998.

[22] T.J.Cole, "Scaling and Rounding Regression Coefficients to Integers", *Applied Statitics-Journal of The Royal Statistical Society Series C*, vol.42, no.1, pp.261-268, 1993.

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

[24] J.R.Koza, *Genetic programming: on the programming of computers by means of natural selection*, MIT, 1992.

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

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