This is a repository copy of *Identification of probabilistic cellular automata* .

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/790/

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Identification of Probabilistic Cellular Automata

Stephen A. Billings and Yingxu Yang

Abstract—The identification of probabilistic cellular automata (PCA) is studied using a new two stage neighborhood detection algorithm. It is shown that a binary probabilistic cellular automaton (BPCA) can be described by an integer-parameterized polynomial corrupted by noise. Searching for the correct neighborhood 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 neighborhood detection technique developed for deterministic rules in [14] to be applied with a larger cutoff value to discard the majority of spurious terms and to produce an initial presearch for the BPCA neighborhood. A multiobjective genetic algorithm (GA) search with integer constraints is then evolved to refine the reduced neighborhood 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 neighborhood and the deterministic rule. The new algorithm is tested over a large set of one-dimensional (1-D), two-dimensional (2-D), and three-dimensional (3-D) BPCA rules. Simulation results demonstrate the efficiency of the new method.

Index Terms—Genetic algorithms (GAs), identification, probabilistic cellular automata (PCA), spatio–temporal systems.

I. INTRODUCTION

PROBABILISTIC cellular automata (PCA), which are referred 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 PCAs. Deterministic CAs 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 [2]. Noise also plays an important role in phase transitions when CAs are employed as a modeling tool to approximate both equilibrium and nonequilibrium systems.

PCAs 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 [3],

population growth [4], and the spread of damage [5]. However, a review of the literature shows that the study of PCA has largely been focused on simulating dynamical systems [6]–[8] and investigations of the dynamical behavior revealed by PCA models [9]–[11]. However, an 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 neighborhood 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 neighborhood is minimal. Only a few authors have studied this problem. In [12], a genetic algorithm (GA) was designed in search of an appropriate probabilistic CA rule through a space of possible PCAs constructed over a number of dimensions. However, the neighborhood selection process was complicated and it was unclear if the neighborhood obtained was minimal. Both sequential and parallel algorithms were introduced in [13] for the identification of PCAs. The neighborhood was found by incrementing the radius by one at each iteration until a preformulated condition was satisfied. Although this produced small neighborhoods the search process was not very flexible or efficient. The simulation results in [1] suggested that the correct neighborhood 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 neighborhood 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-parameterized polynomials corrupted by noise (the probabilistic elements). It is proved that the contribution values for cell terms that define the neighborhood can be calculated without a knowledge of the noise. This is important because this will allow the neighborhood detection scheme developed in [14] for deterministic rules to be employed as a preliminary neighborhood detection tool in the presence of noise. By increasing the cutoff value for the contributions the preliminary neighborhood 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 neighborhood. As a result the number of terms in the candidate term set can be dramatically reduced. A multiobjective GA with integer constraints is then introduced to refine the preselected neighborhood 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 GA now only has to search through the reduced candidate term set.
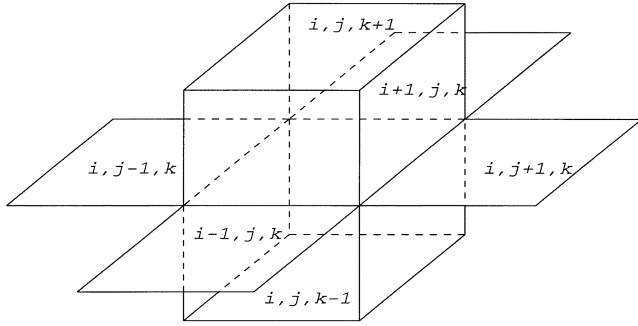
Fig. 1.   Three-dimensional neighborhood.

The paper is organized as follows. In Section II, a class of BPCAs and the polynomial representation are introduced. Section III discusses the preliminary neighborhood selection process. A multiobjective GA with integer constraints is constructed in Section IV and Section V provides the simulation results and some discussions. Section VI contains the conclusions.

II. PROBABILISTIC CELLULAR AUTOMATA

Background knowledge regarding the basics of cellular automata are well-documented in many standard CA text books and publications [1], [15]. For simplicity this is not repeated in the present paper where only the details specific to the current study are introduced (see [1, Sect. 2.0 and 3.1] or [15] for a more detailed introduction to CAs).

A BPCA comprises a lattice of cells, each taking only zero or one 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 neighborhood. An example of a three-dimensional (3-D) PCA neighborhoods is shown in Fig. 1. Note the 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. The probabilistic local rule is composed of 2^n (n is the size of the neighborhood) rule components, where each represents a possible state of the neighborhood. The probabilistic rule is constructed by specifying one or more (not all) of the rule components to be one or zero with probability p (denoted as $1/p$ or $0/p$, respectively), and zero or one with probability $(1-p)$ [denoted as $0/(1-p)$ or $1/(1-p)$, respectively] while the other components are deterministic (zero only or one only). Varying the probability p between one and zero leads to a transition from one deterministic rule (corresponding to $p = 1$) to another rule (corresponding to $p = 0$). A typical rule of a one–dimensional (1-D) 3-site BPCA with von Neumann neighborhood is shown in Table I. This probabilistic rule causes a transition from *Rule 60* (following Voorhees' nomenclature scheme [15]) 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 one at time step t, but may be flipped to 0 with probability $1-p$. These flipped states represent the noise.

TABLE I
EXAMPLE OF 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

III. PRELIMINARY NEIGHBORHOOD DETECTION

Determining the neighborhood 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 neighborhoods 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.

A. Representations of BPCA

Every deterministic binary CA rule with a neighborhood of size n consisting of a set of cells within the CA pattern denoted as $\{cell(x_1), \ldots, cell(x_j), \ldots, cell(x_n)\}$, where x_j is the location of the cell, can be expressed by a Boolean function of the form [1]

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

where

$V = 2^n - 1, x_j$ cell to be updated;
$s(x_i)$ state of $cell(x_i)$ at time step $t-1$;
$s_d(x_j)$ next state in $cell(x_j)$ at time step t;
subscript d used to indicate the rule is deterministic.

a_i $(i = 0, \ldots, V)$ 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. To simplify the notation, $cell(x_j)$ will be replaced by x_j throughout the present study.

The XOR and AND operators can be represented using the normal algebraic plus $+$ and multiplication $\times$ operators to give $h_1 * h_2 = h_1 \times h_2$, $h_1 \oplus h_2 = h_1 + h_2 - 2h_1 \times h_2$. Applying this to (1) shows that every binary deterministic CA rule with a neighborhood $\{cell(x_1), \ldots, cell(x_n)\}$ can be represented by a polynomial of the form

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

where $V = 2^n - 1$ and θ_i $(i = 1, \ldots, V)$ are a set of integers such that (2) maps $s_d(x_j)$ into $\{0, 1\}$.

According to the definition, a BPCA can be represented by

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

where $p_1 = (1 - q) \times p_2$ ($q > 0.5$), q is the probability parameter; p_2 is the probability with which the probabilistic components will appear in the spatio–temporal pattern; and the p subscript indicates that the rule is defined in terms of probabilities. p_1 is related to and smaller than q.

Equation (3) shows that a probabilistic rule is equivalent to the deterministic rule defined by specifying all the probabilistic components to be one, with a probability $1 - p_1$ and the conjugate deterministic rule constructed by assigning zero to all the

probabilistic components, with a probability p_1. Since $s_p(x_j)$, $s_d(x_j)$, and $s(x_i)$ $(i = 1, \ldots, n)$ 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 $s(x_i)$ $(i = 1, \ldots, n)$ can then be denoted as $s_p(x_j; t)$, $s_d(x_j; t)$, and $s(x_i; t-1)$ $(i = 1, \ldots, n)$, 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 [16, eq. (26.3)], $\eta(t)$ can be expanded as

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

where the $g_i()$s $(i = 1, \ldots, m_\eta)$ 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 $(i = 1, \ldots, m_\eta)$.

From the discussions above, every binary probabilistic CA rule with a neighborhood $\{cell(x_1), \ldots, cell(x_n)\}$ 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 $(i = 1, \ldots, V)$ are a set of integers the combinations of which in (7) map $s_d(x_j;\ t)$ into $\{0, 1\}$.

Before moving on to the next section, the relationships between a cell, a neighborhood, a term, and a CA rule need to be clarified. A neighborhood is composed of one or more cells. An assumed neighborhood normally includes other cells that are not within the real neighborhood to be identified. A term is a product of the states of one or more cells within a neighborhood. 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 *Rule 60* over the von Neumann neighborhood can be written as

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

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

B. Orthogonalization and the Noise Model

Equation (2) can also be expressed as

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

where

$$\overline{\theta} = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_V]^T$$

and

$$\mathbf{s}(t-1) = [s(x_1;\ t-1)\ \cdots\ s(x_n;\ t-1)\ \cdots\ s(x_1;\ t-1)$$
$$\times \cdots \times s(x_n;\ t-1)].$$

Or in matrix form

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

where

$$\mathbf{s}_d = [s_d(x_j;\ 1)\ s_d(x_j;\ 2)\ \cdots\ s_d(x_j;\ N)]^T$$

$$\mathbf{S} = [\mathbf{s}^T(0)\ \mathbf{s}^T(1)\ \cdots\ \mathbf{s}^T(N-1)]^T = [\mathbf{s}_1\ \cdots\ \mathbf{s}_V]$$

and N is the total number of time steps in CA evolution. Matrix $\mathbf{S}$ can be decomposed as $\mathbf{S} = \mathbf{E} \times \mathbf{Q}$, where

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

is an orthogonal matrix

$$\mathbf{E}^T \times \mathbf{E} = Diag\left[\mathbf{e}_1^T \times \mathbf{e}_1\ \cdots\ \mathbf{e}_V^T \times \mathbf{e}_V\right]$$

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

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

Equation (9) can then be represented as

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

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

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 neighborhood detection algorithm should be designed to select the correct neighborhood $\{cell(x_1), \ldots, cell(x_n)\}$ from an initial large neighborhood $\{cell(x_1), \ldots, cell(x_n), cell(x_{n+1}), \ldots, cell(x_m)\}$ of size m. The algorithm proposed in [14] selects the relevant V_r $(V_r \leq V)$ terms from the initial term set $\{\mathbf{s}_i, i = 1, \ldots, V_m\}$ $(V_m = 2^m - 1)$ 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 $\mathbf{s}_i$ in (9) makes to $\mathbf{s}_p$. 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 neighborhood. However, terms not included in the correct term set but constructed over cells within the neighborhood and terms constructed over cells out of the neighborhood may also be chosen if noise is introduced (see the simulated examples in Section V). This means that the effects of the noise can be restrained to two parts:

1) inclusion of terms out of the correct term set but constructed over cells within the real neighborhood
2) terms constructed over cells within the assumed neighborhood but out of the real neighborhood.

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 neighborhood, respectively. The nonstationary nature of $\eta(t)$ exhibited in $\beta_i(t)$ in (6) is now expressed by the time dependence of $\tilde{\theta}_i^w(t)$ and $\tilde{\theta}_i^o(t)$.

C. The Effects of Noise on Term Contribution

The term contribution defined in (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 $e_i(t-1) \times \tilde{\theta}_i$ $(i = 1, \ldots, V)$ are present in (12) assume that only $V_r \leq V$ define the correct BPCA model. The irrelevant terms are assumed to correspond to θ_is which are zero in (7). Therefore, (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 (15) and taking the expected value gives

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

Because the $e_i(t-1)$s are orthogonal, $e_i(t-1) \times e_j(t-1) = 0$ $(i \neq j)$, and

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

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

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

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

$$\gamma = 2 \times E\left[\left(\sum_{i=1}^{V_r} e_i(t-1) \times \tilde{\theta}_i\right) \times \left(\sum_{i=1}^{V_w} e_i^w(t-1)\right.\right.$$
$$\times \tilde{\theta}_i^w(t) + \sum_{i=1}^{V_o} e_i^o(t-1) \times \tilde{\theta}_i^o(t) + \xi(t)\bigg)\bigg]$$
$$+ E\left[\left(\sum_{i=1}^{V_w} e_i^w(t-1) \times \tilde{\theta}_i^w(t) + \sum_{i=1}^{V_o} e_i^o(t-1)\right.\right.$$
$$\left.\left.\times \tilde{\theta}_i^o(t) + \xi(t)\right)^2\right]$$
$$= 0 + 0 + 0 + E\left[\sum_{i=1}^{V_w} (e_i^w(t-1) \times \tilde{\theta}_i^w(t))^2\right.$$
$$\left. + \sum_{i=1}^{V_o} (e_i^o(t-1) \times \tilde{\theta}_i^o(t))^2 + \xi^2(t)\right]$$
$$= \frac{1}{N} \sum_{t=1}^{N} \sum_{i=1}^{V_w} (e_i^w(t-1) \times \tilde{\theta}_i^w(t))^2$$
$$+ \frac{1}{N} \sum_{t=1}^{N} \sum_{i=1}^{V_o} (e_i^o(t-1) \times \tilde{\theta}_i^o(t))^2 + \frac{1}{N} \sum_{t=1}^{N} \xi^2(t).$$

Substituting γ back into (18) produces

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

From the definition of $[ct]$ in (13), (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 neighborhood and the noise terms that are related to both cells within, and out of the correct neighborhood. 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 neighborhood ($[ct]^w$) can be calculated independently of the noise terms that are constructed over cells out of the correct neighborhood ($[ct]^o$). This suggests that the neighborhood detection algorithm in [14] can still be employed to detect the neighborhood. The problem is the cutoff point C_{off} because any noise term which has a $[ct]^o$ value larger than $\delta_\xi^2/\delta_{s_p}^2$ will be incorrectly included if C_{off} is still set to zero as in [14]. Ideally, C_{off} should be set to $\sum_{i=1}^{V_o} [ct]_i^o + (\delta_\xi^2/\delta_{s_p}^2)$ 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% occurring with probability p_1, which makes it difficult to use conventional methods to minimize the effects of the noise, it is not easy to calculate C_{off} online. However, (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 V-A suggest that the correct or almost correct neighborhood can still be found if C_{off} is chosen within the range $[0.05, 0.1]$.

IV. RULE SELECTION USING MULTIOBJECTIVE GAs WITH INTEGER CONSTRAINTS

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

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

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

Although the size of the assumed neighborhood 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 [17]–[19] 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. GAs, however, seem to be appropriate and allow integer constraints to be added to polynomial rule selection. GAs will therefore be briefly introduced in the following sections.

A. 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 III, and to determine the associated integer parameters so that the resulting polynomial represents a parsimonious model representation of the BPCA pattern.

TABLE II
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

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 II. The leading zero in the binary vectors denotes plus and the leading one means negative. For integer polynomial parameters of CAs with a dimension under four our results suggest that a range of $[-6, 6]$, $\mu = 4$ is large enough. If higher dimensional or more complex BPCAs 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 (22)

$$c_i(1 : \mu) \to sl_1, \quad c_i(\mu + 1 : 2\mu) \to sl_2, \ldots,$$
$$c_i((V_u - 1)\mu + 1 : V_u\mu) \to sl_{V_u}.$$

If the jth parameter θ_j $(j = 1, \ldots, V_u)$ 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 (22). Define

$$f = [sl_1 \quad sl_2 \quad \cdots \quad sl_{V_u}]$$
$$C = [c_1 \quad c_2 \quad \cdots \quad c_{np}]^T$$
$$D = de(C) = [d_1 \quad d_2 \quad \cdots \quad d_{np}]^T$$

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

B. Multiobjective 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 neighborhood is correct and minimal.

The preliminary neighborhood detection technique in Section III produces the reduced neighborhood $\{cell(x_1), \ldots, cell(x_u)\}$ for the GA search, which should be considerably smaller than the original neighborhood $\{cell(x_1), \ldots, cell(x_n), \ldots, cell(x_u), \ldots, cell(x_m)\}$, but this may still be larger than the true neighborhood $\{cell(x_1), \ldots, cell(x_n)\}$.

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 multiobjective fitness function based on a ranking scheme according to the concept of Pareto optimality [20]. This will guarantee equal probability of reproduction to all nondominant chromosomes and should generate a solution nearest to the optimal. The multiobjective 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
ERROR	$Mer(1)$	$\cdots$	$Mer(i)$	$Mer(i+1)$
RANK	i	$\cdots$	np	
ERROR	$Mer(i+2)$	$\cdots$	$Mer(np)$	

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

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

  iii) The multiobjective fitness function of the ith chromosome is finally defined as

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

To avoid the GA search becoming trapped at a local optima two subpopulations will be introduced which evolve in parallel with the main population [21]. 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.

C. 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 [20]. 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 [22] and [23] for details).

The new multiobjective GA search for polynomial rules with integer constraints can be summarized as follows.

  1) Initialize the two subpopulations and the main population on the basis of the preliminary neighborhood obtained in Section III.
  2) Evaluate the three populations according to Mer, St, and Mer combined with St, respectively, using the ranking technique.
  3) Apply the parent selection technique to the two subpopulations.
  4) Employ crossover and mutation to the two subpopulations separately.
  5) Employ crossover and mutation to the two subpopulations combined to produce new candidates for the main population.
  6) Repeat 2) and insert new populations to replace the three old populations, respectively.
  7) If all chromosomes in the new main population converge to a single individual then stop, otherwise return to 3) and repeat.

After the deterministic model, which is corrupted by noise, is found the minimal neighborhood 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 neighborhood, the occurrences of zero and one in $s_p(x_j; t)$ are recorded. For the deterministic rule components the occurrences of zero and one in $s_p(x_j; t)$ cannot be both nonzero. This probability table represents the identified BPCA.

V. Simulation Studies

A. Preliminary Neighborhood Detection

Three examples, for 1-D, 2-D, and 3-D BPCAs, will be used to demonstrate the preliminary neighborhood detection and the crucial role the increased C_{off} plays in reducing the number of insignificant terms in the selected term set. The initial neighborhoods used in the three examples are defined in Table III. To simplify the notation all the neighborhoods are assumed to

TABLE III
INITIAL NEIGHBORHOODS USED IN EXAMPLES 1–3

| 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	$cell(i,j,k)$	$cell(i+1,j,k)$	$cell(i-1,j,k)$	$cell(i,j+1,k)$	$cell(i,j-1,k)$

Example	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	$cell(i,j,k+1)$	$cell(i,j,k-1)$	$cell(i-1,j+1,k)$	$cell(i-1,j-1,k)$

be 9-site neighborhoods. The candidate term set SET which is determined from the initial neighborhoods 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 neighborhoods. The mappings between entries in SET and cells in the neighborhoods are illustrated in Table III. 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 s_i, $i = 1, \ldots, V$ in matrix $\mathbf{S}$ in (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 s_i, $i = 1, \ldots, V$.

1) Example 1: Example 1 uses the 1-D BPCA rule given in Table I. This rule is equivalent to *Rule 60* occurring with probability p and *Rule 40* occurring with probability $1 - p$. The spatio–temporal patterns produced by the evolution of this BPCA rule with varying p are shown in Fig. 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 where the state initial densities of ones and zeros are both 0.5. Fig. 2(a) and (b) can be considered as patterns produced by the evolution of *Rule 60* with noise at two 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 Fig. 2(a) to a random tree structure in Fig. 2(b) where the noise level is $1 - 0.7 = 0.3$. The noise is introduced by flipping the
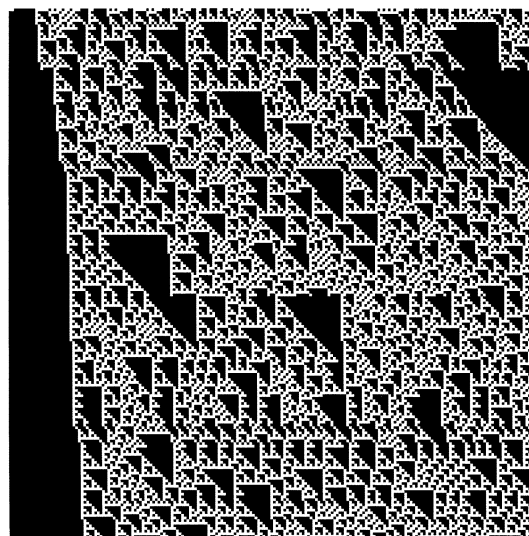
(a)
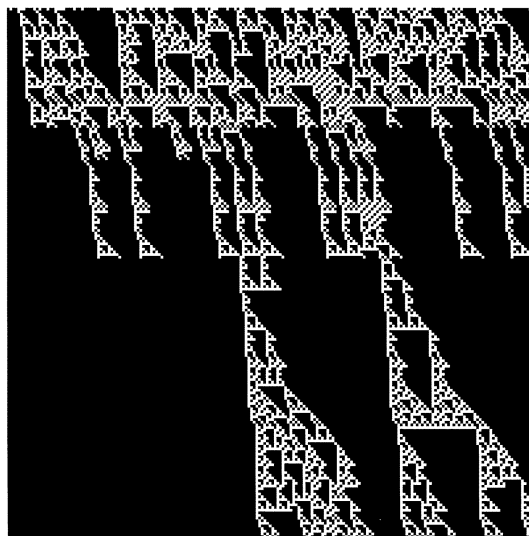
(b)

Fig. 2. Spatio–temporal patterns produced from Example 1 for the evolution of a 1-D BPCA with varying p. (a) $p = 0.9$; (b) $p = 0.7$

states of the updated cells governed by rule components 010 and 100 in *Rule 60* from one to zero and therefore only takes two values, -1 and 0. The impact of the noise is by no means trivial as can be seen in the pattern changes in Fig. 2. This is because the signal/noise ratio is 100% at each point the noise appears.

Data extracted from a noise free simulation of the *Rule 60* was used for detecting the term set of the deterministic rule. C_{off} was chosen as zero 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 three terms have been selected from the SET of 511 terms and the neighborhood that is determined by these terms is $\{cell(j),\, cell(j-1)\}$, which is the same as listed in [15, Appendix 1] and is minimal and correct.

Data extracted from Fig. 2(b) was used in determining the neighborhood 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 zero. Although only 23 out of 511 terms have been selected, the neighborhood covered by these terms is exactly the same as the 9-site neighborhood 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 neighborhood determined by these terms is $\{cell(j),\, cell(j-1),\, cell(j+1),\, cell(j-3)\}$, which fully covers the correct neighborhood but is much smaller than the initial neighborhood. 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.

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1295 \\
1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0562 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0440
\end{bmatrix}
$$
Model $1 - (a)$

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

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

2) Examples 2 and 3: The data used for preliminary neighborhood 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 *Rule 24 235* to be zero with probability p and one with probability $1 - p$ over the neighborhood $\{cell(i+1, j),\, cell(i, j-1),\, cell(i, j+1),\, cell(i-1, j)\}$. Model $2 - (a)$ shows the result when $p = 1$ and C_{off} is set as zero. The nine terms selected constitute the deterministic *Rule 24 235* and referring back to Table III, determine the correct and minimal neighborhood. 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 $C_{\text{off}} = 0.09$ is given in model $2 - (b)$, where 18 terms were selected. A close inspection shows that these 18 terms cover all the nine correct terms in model $2 - (a)$. The remaining terms are composed of two types: 1) five terms made up of cells within the correct neighborhood and 2) four terms made up of cells within the correct neighborhood but including a cell outside the correct neighborhood, $cell(i-1, j+1)$. Example 3 illustrates the preliminary neighborhood detection for a 3-D BPCA. This probabilistic rule was created over a 6-site neighborhood 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, and 111 110 in a deterministic rule (01 111 110 11 010 111 11 100 110 11 011 110 01 000 100 01 011 110 01 110 110 01 010 110) to be one with probability p and zero with probability $1 - p$.

$$
\begin{bmatrix}
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4871 \\
3 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0631 \\
2 & 3 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0.0363 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0217 \\
2 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0292 \\
3 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0481 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0377 \\
2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0322 \\
2 & 3 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0199
\end{bmatrix}
$$
Model $2 - (a)$

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)

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)

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)

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:

1) 36 correct terms in model $3 - (a)$ selected for the deterministic rule when $C_{\text{off}} = 0$;
2) 11 terms which are constructed only over the cells within the correct neighborhood;
3) seven terms which are composed of cells within the correct neighborhood but also one cell out of the correct neighborhood, $cell(i, j + 1, k)$.

Both Model $2 - (b)$ and Model $3 - (b)$ define a neighborhood larger than the correct neighborhood. 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 Examples 2 and 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 neighborhood.

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 neighborhood). 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 the 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 *Rule 40*. This is because the BPCA rule is equivalent to *Rule 60* occurring with probability $p = 0.7$ and *Rule 40* occurring with probability $1 - p = 0.3$.

These three examples also demonstrate that when using the neighborhood detection algorithm in [14], 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 neighborhood can be calculated independently of the $[ct]$ values for noise terms that are constructed over cells out of the correct neighborhood.

The preliminary neighborhoods for Examples 1–3 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 neighborhoods can be refined using the GAs described in the next section.

B. Rule Selection Using GAs With Integer Constraints

The terms selected in Section V-A will be used as initial term sets for the application of the GA algorithm described in Section IV. The algorithm was tested over a large set of 1-D, 2-D, and 3-D BPCA rules with various neighborhoods of ran-domly chosen radius. A sample of the results is summarized in Table IV. 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 one with probability $p = 0.7$ and zero 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 neighborhoods 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 neighborhood size is larger than four. For simplicity only the average and standard deviation (*std.dev.*) values are listed in Table IV. The rules labeled with *Rule $n - k$* ($n = 1, 2, 3; k = 1, 2, \ldots, 14$) have neighborhood size larger than four and the rule numbers were randomly generated in the CA simulation tool developed by the authors.

Inspection of Table IV 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 IV 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 IV, the rule set for the GA search with preliminary neighborhood detection comprises a maximum of $2^4 \times 2^{2^4 - 1} = 524\,288$ 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 neighborhood) the rule set for the GA search with no preliminary neighborhood detection which generates the results in [1, Table VI] consisted of a massive $2^{2^9} = 1.3408e + 154$ rules. Hence, the average run time in Table IV is considerably smaller than in [1, Table VI].

The preliminary neighborhood 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 *Rule 16 798* and *Rule 24 235* 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 neighborhood 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 zero and one in $s_p(x_j; t)$ for every rule component which is determined by the identified neighborhood. This collected probability table represents the identified BPCA. The probability table for the BPCA

TABLE IV
SUMMARY OF RESULTS OBTAINED IN EVOLVING SOME 1-D, 2-D, AND 3-D BPCA POLYNOMIAL RULES WITH VARIOUS SIZES OF NEIGHBORHOODS USING THE GA ROUTINE OF SECTION IV

D	n	Rule	u	Generations		Modulus of errors		Structure		average run time
				mean	std.dev.	mean	std.dev.	mean	std.dev.	
1	3	$Rule40$	4	14.15	2.36	4	0	2	0	25.49 min.
		$Rule60$	4	18.96	3.54	6	0	3	0	38.56 min.
		$Rule18$	4	22.13	4.25	28	0	6	0	40.02 min.
	4	$Rule16798$	6	104.23	6.89	16	0	15	0	200.09 min.
		$Rule24235$	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	$Rule16798$	6	98.92	10.13	19	0	15	0	187.43 min.
		$Rule24235$	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.

TABLE V
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

rule in Example 2 was collected from a data set of 1000 data points and is presented in Table V. For simplicity the probability tables for other BPCA rules discussed are not included in the paper. Table V shows that zero 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 eight 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 zero only or one only. This shows that rule components 0010, 0110, 1000, and 1101 are probabilistic and the probability p is approximately 0.7 ($19/27 = 0.7037$, $54/78 = 0.6923$, $28/41 = 0.6829$, $69/98 = 0.7041$).

VI. Conclusions

Despite the fact that PCA 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 BPCA can be represented as integer-parameterized 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 neighborhood can be calculated independently of the noise terms that are also associated with cells out of the neighborhood. This allows the neighborhood detection technique, originally developed for deterministic rules, to be used to select a preliminary neighborhood even in the presence of noise by increasing the contribution cutoff value. This preliminary neighborhood 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 neighborhood 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.

References

[1] Y. X. Yang and S. A. Billings, "Extracting Boolean rules from CA patterns," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 573–580, Aug. 2000.

[2] G. Mayerkress and K. Kaneko, "Spatio temporal chaos and noise," *J. Stat. Phys.*, vol. 54, no. 5-6, pp. 1489–1508, 1989.

[3] F. M. C. Vieira and P. M. Bisch, "Oscillatory patterns in an autocatalytic reaction ring network simulated by a probabilistic cellular automaton," *Physica A*, vol. 199, no. 1, pp. 40–54, 1993.

[4] N. Boccara *et al.*, "A probabilistic automata network epidemic model with births and deaths exhibiting cyclic behavior," *J. Phys. A, Math. Gen.*, vol. 27, no. 5, pp. 1585–1597, 1994.

[5] H. Hinrichsen *et al.*, "Universality properties of the stationary states in the one-dimensional coagulation-diffusion model with external particle input," *J. Stat. Phys.*, vol. 86, no. 5-6, pp. 1203–1235, 1997.

[6] M. J. de Oliverira and J. E. Satulovsky, "Renormalization group of probabilistic cellular automata with one absorbing state," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat.*, vol. 55, no. 6, pp. 6377–6383, 1997.

[7] D. Hiebeler, "Stochastic spatial models: From simulations to mean field and local structure approximations," *J. Theoret. Biol.*, vol. 187, no. 3, pp. 307–319, 1997.

[8] H. A. Wan, "Modeling stock markets by probabilistic 1-D cellular automata," *Int. J. Comput. Math.*, vol. 53, no. 3-4, pp. 167–176, 1994.

[9] K. Kaneko and Y. Akutsu, "Phase transitions in two-dimensional stochastic cellular automata," *J. Phys. A, Math. Gen.*, vol. 19, no. 2, pp. L69–L75, 1986.

[10] N. Menyhard, "Inhomogeneous mean-field approximation for phase transitions in probabilistic cellular automata: An example," *J. Phys. A, Math. Gen.*, vol. 21, no. 5, pp. 1283–1292, 1988.

[11] M. Katori and H. Tsukahara, "2-neighbor stochastic cellular automata and their planar lattice duals," *J. Phys. A, Math. Gen.*, vol. 28, no. 14, pp. 3935–3957.

[12] F. C. Richards, "Extracting cellular automaton rules directly from experimental data," *Phys. D*, vol. 45, pp. 189–202, 1990.

[13] A. I. Adamatskii, "Identification of probabilistic cellular automata," *Sov. J. Comput. Syst. Sci.*, vol. 30, no. 3, pp. 118–123, 1992.

[14] Y. X. Yang and S. A. Billings, "Neighborhood detection and rule selection from cellular automata patterns," *IEEE Trans. Syst., Man, Cybern. A*, vol. 30, pp. 840–847, Nov. 2000.

[15] B. H. Voorhees, *Computational Analysis of One-Dimensional Cellular Automata, World Scientific Series on Nonlinear Science*. Singapore: World Scientific, 1996.

[16] P. Young, "Developments in time series analysis," in *Time Variable and State Dependent Modeling of Non-Stationary and Nonlinear Time Series*, T. S. Rao, Ed. Berlin, Germany: Springer Verlag, 1993.

[17] C. C. Caroe and R. Schultz, "Dual decomposition in stochastic integer programming," *Oper. Res. Lett.*, vol. 24, no. 1-2, pp. 37–45, 1999.

[18] A. Hassibi and S. Boyd, "Integer parameter estimation in linear models with applications to gps," *IEEE Trans Signal Processing*, vol. 46, pp. 2938–2952, Nov. 1998.

[19] T. J. Cole, "Scaling and rounding regression coefficients to integers," *Appl. Stat.: J. R. Stat. Soc. C*, vol. 42, no. 1, pp. 261–268, 1993.

[20] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1994.

[21] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[22] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[23] A. J. Chipperfield and P. J. Fleming, "Genetic algorithms in control systems engineering," *Contr. Comput.*, vol. 23, pp. 88–94, 1996.

Stephen A. Billings received the B.Eng. degree in electrical engineering with first class honors from the University of Liverpool, Liverpool, U.K., in 1972, the Ph.D. degree in control systems engineering from the University of Sheffield, Sheffield, U.K., in 1976, and the D.Eng. degree from the University of Liverpool in 1990.

He was appointed as Professor in the Department of Automatic Control and Systems Engineering, University of Sheffield, in 1990, and leads the Signal processing and complex systems research group. His research interests include system identification and information processing for nonlinear systems, narmax methods, model validation, prediction, spectral analysis, adaptive systems, nonlinear systems analysis and design, neural networks, wavelets, fractals, machine vision, cellular automata, spatio–temporal systems, and MRI and optical imagery of the brain.

Dr. Billings is a Chartered Engineer [C.Eng.], Chartered Mathematician [C.Math.], Fellow of the IEE [U.K.], and Fellow of the Institute of Mathematics and its Applications.

Yingxu Yang received the Ph.D. degree from the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K.

Her research interests include modeling and prediction, nonlinear optimization, genetic algorithms, neural networks, and spatio–temporal systems. She is currently with Texas Instruments, Münich, Germany.