Monograph:
Yang, Y.X. and Billings, S.A. (1998) Extracting Boolean Rules From CA Patterns. Research Report. ACSE Research Report 719 . Department of Automatic Control and Systems Engineering

Extracting Boolean Rules from CA Patterns

Y.X.Yang S.A.Billings

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

Research Report No.719

25 June 1998

University of Sheffield

Extracting Boolean Rules from CA Patterns

Y.X.Yang S.A.Billings

Dept. of Automatic Control and System Engineering

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

Abstract

A multi-objective GA algorithm is introduced to identify both the neighbourhood and the rule set in the form of a parsimonious Boolean expression for both one- and two-dimensional cellular automata. Simulation results illustrate that the new algorithm performs well even when the patterns are corrupted by static and dynamic noise.

1 Introduction

Cellular Automata (CA) are mathematical models for complex natural systems containing large numbers of simple identical components with local interactions. Since the pioneering work of John von Neumann during the 1950s [1], CA's have been largely employed as a modeling class in a wide range of applications. In physics, CA have been used to model the non-linear diffusion equation [2] and as a filter to reduce noise effects on a high energy discrete calorimeter [3]. CA have also been widely applied in image processing and pattern recognition [4], in digital circuits [5], in robotics systems [6], and many other fields.

However over the past two decades the focus on cellular automata has shifted from the potential to approximate non-linear discrete and continuous dynamical systems to the discovery of particular cellular automata with predefined properties [7] and the detailed study of these properties.

Within this overall direction, Gutowitz [7a] distinguishes two lines of research, relating to what he terms the forward problem and the inverse problem. The forward problem is to determine the characteristics and evolutionary behavior of a given set of individual cellular automata rules. This has attracted most of the attention of current research workers and many useful techniques have been developed. However the inverse problem of determining the cellular automaton which satisfies general sets of prespecified constraints has received relatively little attention. One of the most essential problems in this case is the identification of the cellular automata, that is, how to learn the underlying rule that governs the local behavior of cells from temporal slices of the global evolution of the spatio-temporal pattern.

Cellular Automata in the classical sense are autonomous systems, that is, there are no external inputs exerting an influence on the evolution. It is only possible to observe an evolution of cellular automata and fix the global states or configurations. An identification procedure can then be established based on using these fixed snapshots. In CA identification it is assumed that a given spatio-temporal pattern Ω has a dimension d ($d \geq 1$) and can be described by a cellular automaton. To identify a cellular automaton

is to obtain a minimal description of a cellular automaton Λ which simulates Ω precisely and the size of the neighbourhood must be as small as possible. It is therefore necessary to specify not only the mapping but also the structure of the neighbourhood. Ideally the identification technique should produce a concise expression of the CA rule. This ensures that the model is parsimonious, can be readily interpreted and is important for the hardware realization of the cellular automaton. Richards et al [8] proposed a method for extracting cellular automata rules from given spatio-temporal patterns using a genetic algorithm. Adamatskii [9] discussed the complexity of identification of cellular automata and presented sequential and parallel algorithms for computing the local transition table. However neither of these authors obtained a clear neighbourhood structure or parsimonious rule expression (Boolean expression).

This paper is based on Boolean expressions for one-dimensional cellular automata and the extension of these to the two-dimensional case. An evolutionary algorithm is proposed using a multi-objective genetic algorithm to extract a precise local Boolean expression of the CA rule from given spatio-temporal patterns blurred by noise. To accelerate the convergence subpopulations will be incorporated in the search process.

The remainder of the paper is organized as follows. In Section 2, the definition, characteristics and other relevant background information of a group of one- and two-dimensional cellular automata are introduced. Section 3 reformulates the Boolean expression for one-dimensional CA rules and extends these to the two-dimensional case. The GA search for Boolean expressions of CA rules is then presented with an emphasis on the construction of parsimonious forms of CA rules. Simulation results are contained in Section 4, and Section 5 discusses the efficiency of the algorithm.

2 One-dimensional and Two-dimensional Cellular Automata

Cellular Automata (CA) are mathematical idealizations of physical systems in which space and time are discrete and physical quantities take on a finite set of discrete values. A cellular automaton is based on three parts: a discrete lattice, a neighbourhood and a rule for local transition (or local rule/truth table). All cells are updated synchronically according to a rule and the value assigned to a cell at a given time step depends only on its neighbourhood.

Attention in this paper is restricted to cellular automata with cells that only take binary values (elementary CA). Although this is the simplest possible case, it has been the focus of most investigation due to the capability of generating complicated spatio-temporal patterns of global behavior and capturing essential features of many complex phenomena.

2.1 Neighbourhoods

The neighbourhood of a cell is the set of all other cells capable of directly influencing the evolution. The neighbourhood can be constructed from various combinations of cells from different temporal and spatial scales. Some neighbourhoods, shown in Figure 1, are

used often and have proper names.

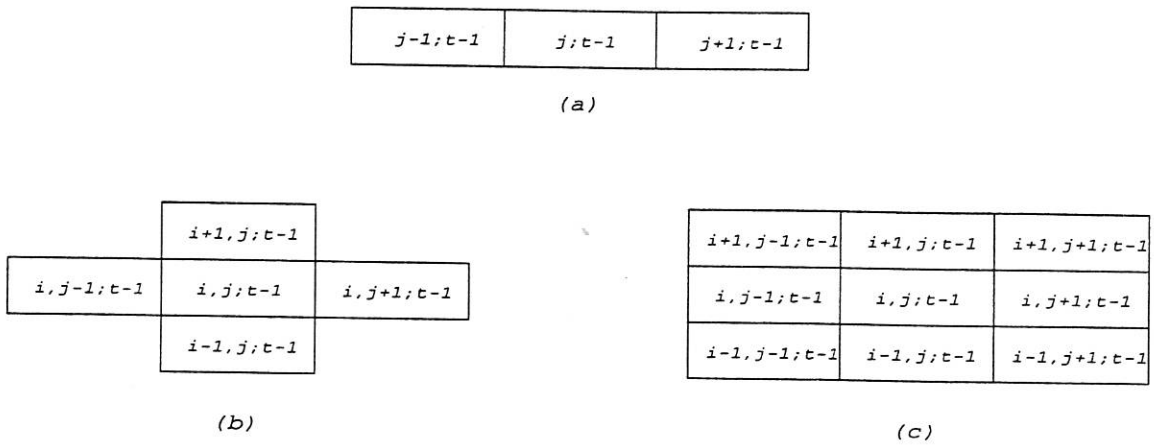


Figure 1: Examples of most frequently used 1-D and 2-D neighbourhoods (a) the 1-D von Neumann neighbourhood, (b) the 2-D von Neumann neighbourhood (c) the 2-D Moore neighbourhood

Denoting the 1-D cell at position j at time t as $cell(j;t)$ and the 2-D cell at position i, j at time t as $cell(i,j;t)$, then the one-dimensional neighbourhood $\{cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1)\}$ (Figure 1 (a)) for $cell(j;t)$ and the two-dimensional neighbourhood $\{cell(i,j-1;t-1), cell(i,j;t-1), cell(i,j+1;t-1), cell(i-1,j;t-1), cell(i+1,j;t-1)\}$ (Figure 1 (b)) for $cell(i,j;t)$ are the von Neumann neighbourhoods, while the two-dimensional neighbourhood $\{cell(i,j-1;t-1), cell(i,j;t-1), cell(i,j+1;t-1), cell(i-1,j-1;t-1), cell(i-1,j;t-1), cell(i-1,j+1;t-1), cell(i+1,j-1;t-1), cell(i+1,j;t-1), cell(i+1,j+1;t-1)\}$ (Figure 1 (c)) for $cell(i,j;t)$ defines the Moore neighbourhood.

2.2 Local rules

Local interaction rules are defined in terms of the influence that cells in the neighbourhood of a cell have on the updated value to be placed in that cell. The rules are labeled by assigning neighbourhoods in ascending numerical order and treating the listing of 0's and 1's which are obtained by specifying which neighbourhoods map to 0 and which to 1. This defines the rule/truth table. The component form of a 3-site one-dimensional rule R is shown as follows:

000	001	010	011	100	101	110	111
r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7
2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7

where $r_i, i = 0, \cdots, 7$ are the rule components taking only binary numbers and the last row shows the coefficients associated with the corresponding components. Therefore $R = (r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7)$ and the numerical label D assigned to R is given by $D(R) = \sum_{s=0}^{2^3-1} r_s 2^s$, which is simply the sum of the coefficients associated with all non-zero components.

2.3 Stochastic/probabilistic CA rules

Each cell value in all of the CA's mentioned above is updated according to a fixed deterministic rule. A particular pattern of noise (imperfection) can be added to an initially deterministic rule in two ways. Static noise is introduced by flipping a certain number of cell states according to a given probability after the CA evolution. Dynamic noise is imposed upon CA patterns by specifying one or more (not all) of the components to be 1 with a probability p, and 0 with probability $(1-p)$ while the other components remain fixed. These rules are called stochastic rules or more commonly probabilistic rules. For the CA rules corrupted by dynamic noise, a transition from one deterministic rule ($p = 0$) to another ($p = 1$) would occur if the parameter p is altered from 0 to 1. A detailed study and identification of probabilistic rules will be presented in Section 4.

3 Extracting Boolean rules using genetic algorithms

3.1 Boolean form of CA rules

In a more precise way, an elementary cellular automaton with N cells can be considered as a fully discrete dynamical system whose evolution is governed by the iteration of a global mapping F from time $t-1$ to time t:

$$F: \quad \{0,1\}_{t-1}^{N} \rightarrow \{0,1\}_{t}^{N}$$

The homogeneity and the locality of cellular automata permit a compact description of F in terms of a local transition function f,

$$f: \quad \{0,1\}_{t-1}^{n} \rightarrow \{0,1\}_{t}$$

that maps the occupancies at time $t-1$ of a neighbourhood of size n around any cell to the next state at time t of that cell.
For a one-dimensional CA, this is denoted as

$$s(j;t) = f(N(j;t-1))$$

where $s(j;t)$ is the state of the cell at position j at time step t and $N(j;t-1)$ represents the states of the cells within the neighbourhood of cell j at time step $t-1$.
For a two-dimensional CA, this is denoted as

$$s(i,j;t) = f(N(i,j;t-1))$$

where $s(i,j;t)$ is the state of the cell at position (i,j) at time step t and $N(i,j;t-1)$ represents the states of the cells within the neighbourhood of cell (i,j) at time step $t-1$.
Since in general f is equivalent to the component form of R and the rule/truth table of length 2^n, where n is the size of the neighbourhood, f can also be viewed as a logical function or Boolean function of n variables.
In [10], the Boolean form of all the 3-site 1-D CA rules with even number labels are listed using NOT, AND and XOR operators. The Boolean form of $Rule22$, for instance, is

$$\begin{aligned} s(j;t) \;=\; & s(j-1;t-1)XORs(j;t-1)XORs(j+1;t-1)XOR \\ & (s(j-1;t-1)ANDs(j;t-1)ANDs(j+1;t-1)) \end{aligned} \tag{1}$$

Note that equation 1 is actually the XOR combination of all the terms connected by AND. It is not easy to see how to identify equation 1 directly because it involves the logical combination of terms. Linear in the parameter regression methods are therefore inappropriate and alternative approaches have to be considered.
However it can be observed that every one-dimensional 3-site elementary cellular automaton can be represented by a Boolean function with NOT, AND and XOR operators. Furthermore, note that

$$NOT(a) = 1XORa \quad 0XORa = a$$

Hence all the one-dimensional 3-site elementary cellular automata can be represented by a Boolean function with only AND and XOR operators of the form

$$\begin{aligned} s(j;t) &= a_0 XOR a_1 s(j;t-1) XOR \cdots XOR \\ &\quad a_7(s(j-1;t-1) AND s(j;t-1) AND s(j+1;t-1)) \end{aligned} \tag{2}$$

where a_i $(i = 0, \cdots, 7)$ are binary numbers and $a_i = 1$ indicates that the following term is included in the Boolean function while $a_i = 0$ indicates that the following term is not included.
Note that the number of possible expressions in 2 is $2^8 = 256$ which is exactly the number of all 3-site one-dimensional rules. This implies that the representation in 2 is unique, one set of $\{a_i, i = 0, \cdots, 7\}$ corresponds to one and only one CA rule.
This can be extended to multi-dimensional CA's. For instance any two-dimensional CA with a 5-site neighbourhood $\{cell(i-1,j;t-1), cell(i,j-1;t-1), cell(i,j;t-1), cell(i,j+1;t-1), cell(i+1,j;t-1)\}$ can be represented by a Boolean expression

$$\begin{aligned} s(i,j;t) &= a_0 XOR a_1 s(i-1,j;t-1) XOR \cdots XOR \\ &\quad a_{31}(s(i-1,j;t-1) AND \cdots AND s(i+1,j;t-1)) \end{aligned} \tag{3}$$

Extending this further, every CA with an n site neighbourhood $\{cell(x_1;t-1), \cdots, cell(x_n;t-1)\}$ may be written as

$$\begin{aligned} s(x_j;t) &= a_0 XOR a_1 s(x_1;t-1) XOR \cdots XOR \\ &\quad a_N(s(x_1;t-1) AND \cdots AND s(x_n;t-1)) \end{aligned} \tag{4}$$

where $N = 2^n - 1$ and x_j is the cell to be updated.

3.2 Selecting Boolean rules using a genetic algorithm

Expression 4 considerably simplifies the CA identification problem. Assume that the only a priori knowledge is the dimension of the CA, which can be obtained from examining the spatio-temporal patterns. Then the emerging difficulty lies in how to determine which terms should be included in the Boolean expression and which should be discarded. The problem is very similar to the term selection problem in structure detection for nonlinear system identification. However in CA identification all the terms are combined by the XOR operator and are therefore nonlinear in the parameters, whereas in nonlinear system

identification all the items are combined by the ordinary addition operator and can often be configured to be linear in the parameters. This difference induces increased difficulty in CA term selection. GA's [12] therefore appear to be a natural choice to search for appropriate terms through the space of logical models constructed upon AND and XOR operators due to the ability to globally optimize nonlinear and noncontinuous functions through updating of a whole population of possible solutions at each iteration. The GA search will be implemented as follows.

3.2.1 Definition and initialization of the population

In the current application the problem to be solved can be simplified to selecting certain terms from a given term set consisting of all the possible combinations of states of cells within the assumed neighbourhood. A chromosome will be defined as a $1 \times N$ binary vector c_i, where $N = 2^n - 1$ and n is the size of the largest possible neighbourhood assumed $\{cell(x_1; t-1), \cdots, cell(x_n; t-1)\}$. Each entry in the chromosome corresponds to a term in the set:

$$c_i(1) \rightarrow 1, c_i(2) \rightarrow s(x_1; t-1), c_i(3) \rightarrow s(x_2; t-1),$$
$$\cdots, c_i(N) \rightarrow s(x_1; t-1)AND \cdots ANDs(x_n; t-1)$$

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

$$itemf = [1, s(x_1; t-1), \cdots, s(x_1; t-1)AND \cdots ANDs(x_n; t-1)]$$
$$C = [c_1; c_2; \cdots; c_m].$$

where m is the population size. The whole population C is initialized by assigning each chromosome as a randomly generated binary vector with N bits.

3.2.2 Fitness function

The fitness function is used to evaluate how well a certain rule structure performs in re-generating the behavior of observed spatio-temporal evolution. Firstly an error function is defined as $Error(i) = \sum_j^{SET} |o(i,j) - \hat{o}(i,j)|$ where $o(i,j)$ is the original measured state at data point j for chromosome i and $\hat{o}(i,j) = c_i XORItemf_j$, where

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} XOR \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix} = (a_1 ANDb_1) XOR (a_2 ANDb_2) XOR \cdots XOR (a_n ANDb_n)$$

$$\begin{bmatrix} a_1 & 0 & a_3 \end{bmatrix} XOR \begin{bmatrix} b_1 \\ b_2 \\ b_n \end{bmatrix} = (a_1 ANDb_1) XOR (a_3 ANDb_3).$$

The fitness function

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

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

3.2.3 Reproduction, crossover and mutation

The reproduction process selects a new population by extracting individuals after several repetitions of the old population. The selected population is then used for genetic operations in the breeding process. There are two classic genetic operators for producing new chromosomes during the breeding process. Crossover produces new chromosomes which have some segments of both parents' genetic structure. Mutation is a random process which alters one or more bits in a chromosome with a probability equal to the mutation rate. For details see [13] and [14].

3.3 Multi-objective genetic algorithms

Searching for a Boolean expression to represent CA rules using the GA algorithm can produce excellent results with 0 error (a fitness of 1). However it is not always true that the identified neighbourhood correctly matches the true neighbourhood. The initial assumed neighbourhood will almost always encompass more cells than are actually in the neighbourhood. A one-dimensional 3-site rule *Rule*22, for instance, should be

$$\begin{aligned} s(j;t) &= s(j-1;t-1)XORs(j;t-1)XORs(j+1;t-1)XOR \\ &\quad (s(j-1;t-1)ANDs(j;t-1)ANDs(j+1;t-1)) \end{aligned} \tag{5}$$

while during the GA search the rules are often searched for over a larger neighbourhood. For instance, if a CA rule is considered as operating on a 5-site neighbourhood $\{cell(j-2;t-1), cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1), cell(j+2;t-1)\}$ then a rule of the form

$$\begin{aligned} s(j;t) &= a_0 XOR a_1 s(j-2;t-1)XOR \cdots XOR \\ &\quad a_{31}(s(j-2;t-1)AND \cdots ANDs(j+2;t-1)) \end{aligned} \tag{6}$$

will be assumed and it is highly likely that the GA will select more than one expression including 5 from 6 to match the patterns. The solution is therefore not necessarily unique and this often leads to a false extension of the neighbourhood.
Note that a rule with a larger neighbourhood cannot be represented by a rule with a small neighbourhood while a rule with a smaller neighbourhood can be expressed by a rule with a larger neighbourhood. Therefore the true model is always the smallest model amongst all the possible models chosen. This is the principle of parsinomy. Thus another search

goal is required to direct the GA evolution to a parsimonious logical model.
It is known that if there are two objectives to be optimized, it might be possible to find a solution which is best with respect to the first objective and another solution which is the best with respect to the second. The final result may therefore be a balance between these two best solutions.
Alternatively, a multi-objective GA search method based on ranking according to the concept of Pareto optimality would guarantee equal probability of reproduction to all non-dominant chromosomes and consequently generate a solution nearest to the optimal. In this problem, the two search objectives are to minimize the error function and to minimize the number of terms in each chromosome with the same error. For the current population with size m, each chromosome is ranked according to its error. The chromosome with the least error occupies the first position , the chromosome with the second least error occupies the second, etc. Chromosomes with the same error will share the same rank. Thus,

$$\begin{array}{cccccccc} 1 & \cdots & i & & i & & \cdots & m \\ Error(1) & \cdots & Error(i) & Error(i+1) & & Error(i+2) & \cdots & Error(m) \end{array}$$

with

$$Error(1) < \cdots < Error(i) = Error(i+1) = Error(i+2) < \cdots < Error(m).$$

Define the structure function $su(i) = \sum_j c_i(j)$ and then resort the orders of chromosomes sharing the same rank in proportion to the associated $su(i)$ with the ranks of the rest unchanged. Thus,

$$\begin{array}{cccccccc} 1 & \cdots & i & i+1 & & i+1 & \cdots & m \\ Error(1) & \cdots & Error(i) & Error(i+1) & & Error(i+2) & \cdots & Error(m) \\ su(1) & \cdots & su(i) & su(i+1) & & su(i+2) & \cdots & su(m) \end{array}$$

with $su(1) < \cdots < su(i) < su(i+1) = su(i+2) < \cdots < su(p)$. Consequently, the fitness function is defined as

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

This ranking technique will result in a search with a preference towards the first objective *Error*. The structure function *su* will not have any impact on the first few steps of the search since all the chromosomes are likely to hold various ranks at that initial stage and the fitness of each chromosome is determined exclusively by the error function *Error*. Only after certain chromosomes have converged to a similar *Error*, is it possible to re-arrange the ranking at that error according to the associated *su*. This search process will always select individuals with the smallest structure within the span of the lowest error. Hence chromosomes with a parsimonious logical expression and zero error will remain in the latest population to produce the final solution.

3.4 Multi-objective GA with subpopulations

In a multi-objective fitness landscape, local optima offer the GA more than one opportunity for evolution. Although populations are potentially able to search many local optima,

a finite population tends to settle on a single good optimum, even if other equivalent optima exist. This phenomenon is known as genetic drift and could cause the GA search to become trapped at a local optima. A practical solution is to use a parallel implementation of the iterative method where several subpopulations evolve independently at the same time.
In the search for a parsimonious logical model, two subpopulations are initialized and motivated to evolve separately under different governing objectives. One is to minimize the error function *Error*, the other to minimize the structure function *su*. Together these should produce new candidates for the main population through crossover and mutation. The main population would 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 subpopulations and is subject to evaluation by the ranking technique.
The new multi-objective GA search process with two subpopulations can be summarized as:

1. Initialize the subpopulations and the main population.
2. Evaluate the three populations according to *Error*, *su* and *Error* combined with *su* respectively using the ranking technique.
3. Apply the sampling technique to the two subpopulations.
4. Employ crossover and mutation to the two subpopulations separately.
5. Employ crossover and mutation to the two subpopulations combinedly to produce new candidates for the main population.
6. Repeat (ii) and insert new populations to the three old populations respectively.
7. If all chromosomes in the new main population converge to a single individual then stop, otherwise return to (iii) and repeat.

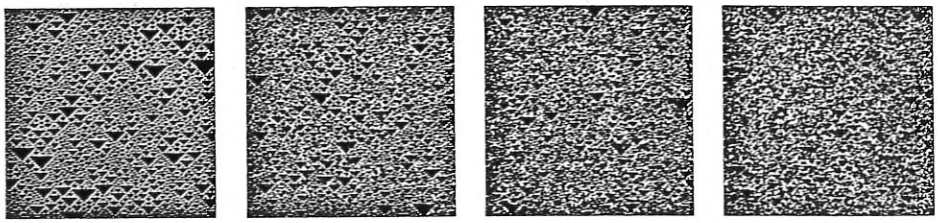
4 Simulation analysis

4.1 The effects of noise

4.1.1 Static noise

In system identification noise is usually classified as either white noise with zero mean or more commonly coloured noise. In cellular automata noise is a form of imperfection which at a critical magnitude can induce an essential phase transition which can suddenly change the behavior of the CA. Static noise can be added to a spatio-temporal pattern by first evolving a deterministic CA rule and then randomly flipping a limited number of binary values according to a specified probability p, where $p = \frac{q_1}{q_2}$ and q_1 is the number of cells to be flipped and q_2 is the total number of cells in the spatio-temporal pattern. This will be referred to as static noise because it is added after the CA evolution. Figure

2 shows the original noise free pattern ($p = 0$) for the one-dimensional von Neumann 3-site *Rule*22 and the same pattern corrupted by noise with probabilities of switching of $p = 0.05$, $p = 0.1$, and $p = 0.2$ respectively. All these were developed on a 200×200 lattice with time evolution from top to bottom and a periodic boundary condition. That is the lattice is taken as a circle, so the first and last sites are identified as if they lay on a circle of finite radius.



$(a)p = 0$ $(b)p = 0.05$ $(c)p = 0.1$ $(d)p = 0.2$

Figure 2: Noise free pattern and static noise contaminated patterns produced by 1-D *Rule*22

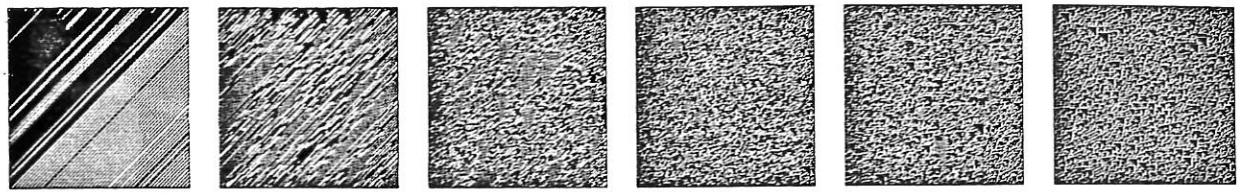
4.1.2 Dynamic noise

Unlike static noise which is added to the CA patterns after the evolution, dynamic noise is directly involved in the development of the CA patterns by specifying one or more (not all) of the components of the rule be 1 with a probability p and 0 with probability $(1-p)$, where $p = \frac{w_1}{w_2}$ and w_1 is the number of the prespecified component to be filled in by 1 and w_2 is the number of the prespecified component. The contaminated or stochastic CA may exhibit a rule transition according to the noise probability. Figure 3 shows the transition from a simple one-dimensional 3-site *Rule*184 to a complicated chaotic one-dimensional 3-site *Rule*60 under the stochastic rule in Table 1. Note that in cases like this the maximum noise density for the transition rule is 50 percent. For instance when $p = 0.6$, the rule transition behaves more like *Rule*60 than *Rule*184 and therefore the noise density for the transition should be considered as $1 - p = 0.4$ for *Rule*60 rather than 0.6 for *Rule*184.

Table 1. Transition of truth table from *Rule*184 to *Rule*60

Components	000	001	010	011	100	101	110	111
Rule184	0	0	0	1	1	1	0	1
Transitiion	0	0	p	1	1	1	0	$1-p$
Rule60	0	0	1	1	1	1	0	0

where $0 < p < 1$.



$(a)p = 0$ $(b)p = 0.1$ $(c)p = 0.5$ $(d)p = 0.65$ $(e)p = 0.8$ $(f)p = 1.0$

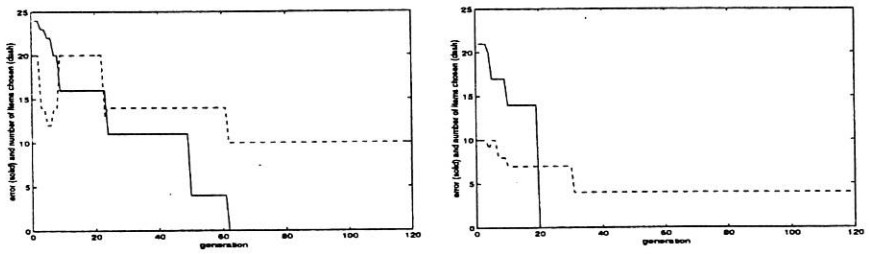
Figure 3: Transition from *Rule*184 ($p = 0$) to *Rule*60 ($p = 1$) with p varying to indicate different noise densities

4.2 Identification of the Boolean expression of 1-D CA rules with noise

4.2.1 Patterns corrupted by static noise

Assume the neighbourhood structure of a class of one-dimensional CA's is defined by $\{cell(j-2;t-1), cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1), cell(j+2;t-1)\}$. Given the spatio-temporal patterns in Figure 2 corrupted by various levels of static noise, the GA identification technique of Section 3 was used to produce the results in Figure 4 and 5.

1. The noise free case $p = 0$

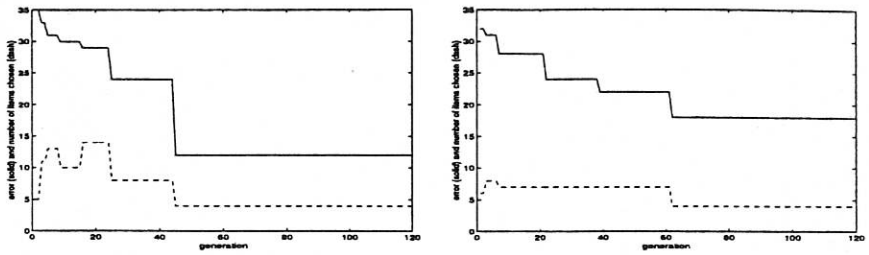


(a) (b)

The dash line is the evolution of the number of items of the chromosome with the best fitness, solid line is the evolution of the error of the chromosome with the best fitness.

Figure 4: Different GA evolutions of noise free *Rule*22 (a) Ordinary unmodified GA evolution of noise free *Rule*22, (b) Multi-objective GA evolution with subpopulations of noise free *Rule*22

2. The noisy cases $p = 0.1$ and $p = 0.2$



(a) (b)

The dash line is the evolution of the number of items of the chromosome with the best fitness, the solid line is the evolution of the error of the chromosome with the best fitness.

Figure 5: Multi-objective GA evolution with subpopulations (a) static noise, level $p = 0.1$ (b) static noise, level $p = 0.2$

Figure 4 shows the results of identifying a Boolean rule from the pattern generated by $Rule22$ without noise. Figure 4 (a) illustrates the convergence of the error and the number of items in the Boolean expression obtained through an ordinary unmodified GA search. The error converges to 0 after 62 generations but the number of items shows no sign of decreasing further after settling at 10, which from inspection of equation 5 is obviously a wrong result implying an incorrectly extended neighbourhood. The multi-objective evolution in Figure 4 (b) produces a more promising result with the structure diminishing after the error has settled to zero. Furthermore, in Figure 4 (b) the error convergence, the error settles to zero after 20 generations, is considerably faster than in Figure 4 (a) where the error converges to zero after 62 generations because subpopulations are incorporated to accelerate the convergence. The Boolean rules produced by Figure 4 (a) and (b) after 120 generations are shown below.
The rule from Figure 4 (a) was

$$\begin{aligned} s(j;t) \;=\; & s(j-1;t-1)XORs(j;t-1)XORs(j+1;t-1)XOR(s(j-1;t-1)AND \\ & s(j+1;t-1))XOR(s(j-1;t-1)ANDs(j;t-1)ANDs(j+2;t-1))XOR \\ & (s(j-1;t-1)ANDs(j+1;t-1)ANDs(j-2;t-1))XOR(s(j-1;t-1)AND \\ & s(j+1;t-1)ANDs(j+2;t-1))XOR(s(j-1;t-1)ANDs(j;t-1)AND \\ & s(j+1;t-1)ANDs(j-2;t-1))XOR(s(j-1;t-1)ANDs(j;t-1)AND \\ & s(j-2;t-1)ANDs(j+2;t-1))XOR(s(j-1;t-1)ANDs(j+1;t-1)AND \\ & s(j-2;t-1)ANDs(j+2;t-1)) \end{aligned}$$

and the rule from Figure 4 (b) was

$$\begin{aligned} s(j;t) \;=\; & s(j-1;t-1)XORs(j;t-1)XORs(j+1;t-1)XOR(s(j-1;t-1)AND \\ & s(j;t-1)ANDs(j+1;t-1)) \end{aligned}$$

Although the result from Figure 4 (a) produces the same truth table as the result from Figure 4 (b), the neighbourhood structure of the former is wrong since the identified Boolean rule covers the 5-site neighbourhood $\{cell(j-2;t-1), cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1), cell(j+2;t-1)\}$ while the actual neighbourhood should be $\{cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1)\}$. The reason lies in the unmodified GA search which selects the rule with the minimum error without considering the structure of the neighbourhood, the second objective. The result from Figure 4 (b) was obtained using a GA search which took into account the second objective, and this produced a rule structure exactly the same as listed in [10] with a correct neighbourhood and a parsimonious Boolean expression.
Figures 5 shows the GA evolution obtained by searching for $Rule22$ with static noise density $p = 0.1$ and $p = 0.2$ respectively. Although the error in both cases does not converge to zero, due to the randomly flipping of some cell values, correct and parsimonious Boolean expressions were still obtained as

$$\begin{aligned} s(j;t) &= s(j-1;t-1)XORs(j;t-1)XORs(j+1;t-1)XOR(s(j-1;t-1)AND \\ &\quad s(j;t-1)ANDs(j+1;t-1)) \end{aligned}$$

for both $p = 0.1$ and $p = 0.2$. These results suggest that the GA search is not sensitive to static noise when the noise density is within a certain amplitude, in this case, $p \leq 0.2$.

4.2.2 Patterns corrupted by dynamic noise

While static noise is added to the CA patterns after the evolution, dynamic noise is immediately involved in the development and tends to induce much more complicated behavior changes. Ideally, an identification procedure should be designed to remain insensitive to these disturbances and to recover the underlying rule. Figure 6 shows the results of a GA search for the appropriate CA rule using data generated in the transition from $Rule184$ to $Rule60$ as shown in Figure 3.
In Figure 6 (a) for $p = 0$ the search result produces a parsimonious Boolean expression of $Rule184$:

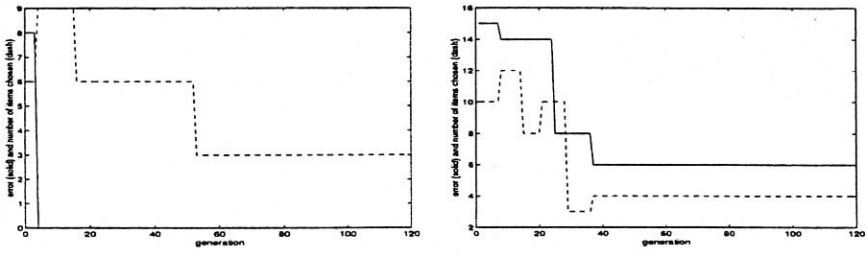
$$\begin{aligned} s(j;t) &= s(j-1;t-1)XOR(s(j-1;t-1)ANDs(j;t-1)) \\ &\quad XOR(s(j;t-1)ANDs(j+1;t-1)). \end{aligned}$$

In Figure 6 (b) $p = 0.1$ the result is

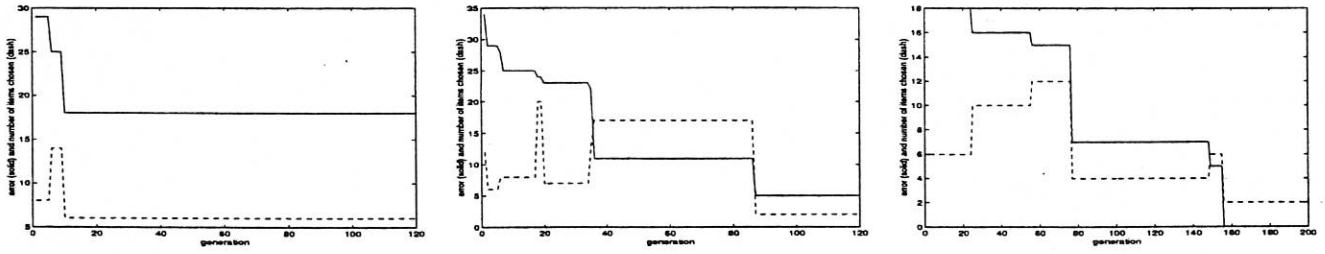
$$\begin{aligned} s(j;t) &= s(j-1;t-1)XOR(s(j-1;t-1)ANDs(j;t-1))XOR(s(j;t-1) \\ &\quad ANDs(j+1;t-1)XOR(s(j;t-1)ANDs(j;t-1)ANDs(j+1;t-1)) \end{aligned}$$

which generates the rule components (0001110). Notice that the last bit in the rule components (00011101) is incorrect. The reason for this appears to lie in the nature of the rule itself. Rules which produce simple patterns of self-repetition or shifting imply that a certain number of combinations of the states of the cells within the neighbourhood will probably never appear in the existing data set. Hence the GA is unlikely to learn

that particular behavior and most probably indiscriminately selects a rule from a range of rules only satisfying other combinations. For *Rule*184 even though the evolution is slightly complicated and the data set sampled from the noise free pattern is rich enough to produce the correct rule component (00011101), the combination of the states of the cells within the neighbourhood (111) which corresponds to the last component appears infrequently and the noise simplifies the data set by eliminating (111). Consequently the GA fails to learn the behavior of (111) and hence produces a wrong result. A similar phenomena applies to other simple 1-D rules such as *Rule*46, *Rule*116, *Rule*72 and *Rule*172. The GA search may repeatedly produce incorrect rules even after the data size has been enlarged.



(a) (b)



(c) (d) (e)

The dash line is the evolution of the number of items of the chromosome with the best fitness, the solid line is the evolution of the error of the chromosome with the best fitness.

Figure 6: GA search of CA rules in transition induced by dynamic noise (*a*) GA evolution of the rule transition from *Rule*184 to *Rule*60 with dynamic noise $p = 0$ (*b*) GA evolution of the rule transition from *Rule*184 to *Rule*60 with dynamic noise $p = 0.1$ (*c*) GA evolution of the rule transition from *Rule*184 to *Rule*60 with dynamic noise $p = 0.65$ (*d*) GA evolution of the rule transition from *Rule*184 to *Rule*60 with dynamic noise $p = 0.8$ (*e*) GA evolution of the rule transition from *Rule*184 to *Rule*60 with dynamic noise $p = 1$

However for Figure 6 (d) $p = 0.8$ where the noise density is $1 - 0.8 = 0.2$ for the rule transition, the search result is

$$s(j;t) \;=\; s(j-1;t-1)XOR(s(j;t-1),$$

which is correct and parsimonious even though the noise level is 10 percent higher than in (b), where the noise level is 0.1 for the rule transition. This appears to be due to the chaotic nature of *Rule*60 which is able to generate patterns complicated enough so that even after the patterns are contaminated by a relatively high density of noise the data set still contains sufficient information to correctly characterize the behavior. When $p = 1$, Figure 6 (e), the result is simply a Boolean expression of *Rule*60

$$s(j;t) = s(j-1;t-1)XORs(j;t-1).$$

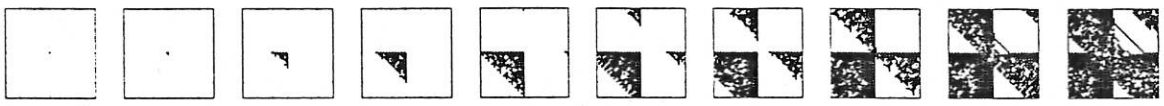
Notice that for Figure 6 (c) $p = 0.65$, the GA search also converged to the correct result

$$s(j;t) = s(j-1;t-1)XOR(s(j;t-1)$$

despite the high noise density of 35 percent.

4.3 Extracting Boolean rules from 2-D CA patterns

Figure 7 shows patterns formed by the evolution of a two-dimensional cellular automaton with a 5-site von Neumann neighbourhood from a simple seed. The patterns are formed on a 60×60 lattice with a periodic boundary. Each frame shows the two-dimensional configuration generated by the evolution of the cellular automaton after the indicated number of time steps. The seed consists of a single nonzero site as illustrated in Figure 7 (a). The growth of cellular automata from such initial conditions should provide models for a variety of physical and other phenomena. One example is crystal growth [11].

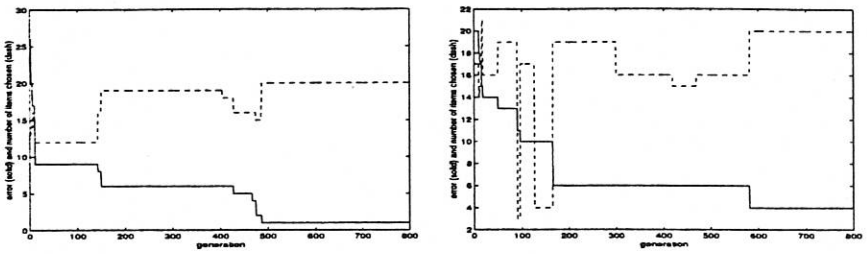


(a) t=0 (b) t=1 (c) t=10 (d) t=20 (e) t=30 (f) t=40 (g) t=50 (h) t=60 (i) t=70 (j) t=80

Figure 7: Evolution of a 2-D CA rule with 5-site von Neumann neighbourhood

4.3.1 Identification of a noise free 2-D CA

Assume the neighbourhood structure of the two-dimensional CA which produced the spatio-temporal patterns in Figure 7 is $\{cell(i,j-1;t-1), cell(i,j;t-1), cell(i,j+1;t-1), cell(i-1,j-1;t-1), cell(i-1,j;t-1), cell(i-1,j+1;t-1), cell(i+1,j-1;t-1), cell(i+1,j;t-1), cell(i+1,j+1;t-1)\}$, the Moore neighbourhood. The multi-objective GA identification technique of Section 3 was then used to produce the results illustrated in Figure 8 (a).



(a) (b)

The dash line is the evolution of the number of items of the chromosome with the best fitness, the solid line is the evolution of the error of the chromosome with the best fitness.

Figure 8: (a) GA search of the Boolean rule for a noise free 2-D CA (b) GA evolution of the rule transition from rule1 to rule2 with $p = 0.8$

The search result after 800 generations contained 20 items in the Boolean expression of the form

$$
\begin{aligned}
s(i,j;t) \;=\; & s(i,j-1;t-1)XORs(i,j;t-1)XORs(i,j+1;t-1)XORs(i-1,j;t-1) \\
& XOR(s(i+1,j;t-1)ANDs(i,j+1;t-1))XOR(s(i,j-1;t-1) \\
& ANDs(i,j;t-1))XOR(s(i,j-1;t-1)ANDs(i,j+1;t-1)) \\
& XOR(s(i,j-1;t-1)ANDs(i+1,j;t-1))XOR(s(i,j;t-1) \\
& ANDs(i,j+1;t-1))XOR(s(i+1,j;t-1)ANDs(i,j-1;t-1) \\
& ANDs(i,j;t-1))XOR(s(i+1,j;t-1)ANDs(i,j-1;t-1) \\
& ANDs(i,j+1;t-1))XOR(s(i+1,j;t-1)ANDs(i,j;t-1) \\
& ANDs(i,j+1;t-1))XOR(s(i+1,j;t-1)ANDs(i,j;t-1) \\
& ANDs(i-1,j;t-1))XOR(s(i+1,j;t-1)ANDs(i,j+1;t-1) \\
& ANDs(i-1,j;t-1))XOR(s(i,j-1;t-1)ANDs(i,j;t-1) \\
& ANDs(i,j+1;t-1))XOR(s(i,j;t-1)ANDs(i,j+1;t-1) \\
& ANDs(i-1,j;t-1))XOR(s(i+1,j;t-1)ANDs(i,j-1;t-1) \\
& ANDs(i,j;t-1))ANDs(i,j+1;t-1))XOR(s(i+1,j;t-1) \\
& ANDs(i,j-1;t-1)ANDs(i,j+1;t-1)ANDs(i-1,j;t-1)) \\
& XOR(s(i+1,j;t-1)ANDs(i,j;t-1)ANDs(i,j+1;t-1) \\
& ANDs(i-1,j;t-1))XOR(s(i+1,j;t-1)ANDs(i,j-1;t-1) \\
& ANDs(i,j;t-1))ANDs(i,j+1;t-1)ANDs(i-1,j;t-1))
\end{aligned}
$$

It can be seen that this two-dimensional rule covers a 5-site von Neumann neighbourhood $\{cell(i,j-1;t-1), cell(i,j;t-1), cell(i,j+1;t-1), cell(i-1,j;t-1), cell(i+1,j;t-1)\}$, which has been extracted from the assumed 9-site Moore neighbourhood. The rule table produced by the identified Boolean function is also correct. However as Figure 8 (a) shows the convergence of the error is much slower than in the one-dimensional noise free

case, 100 compared to 20 generations. This is because the growth of the size of the neighbourhood will inevitably induce a considerable increase in the possible items which can be included in the Boolean expression and therefore an increase in the number of bits in each chromosome.

4.3.2 Identification of a 2-D CA corrupted by dynamic noise

To demonstrate the impact of noise on the GA search, data obtained from patterns in Figure 9 was used for the identification. Figure 9 shows the spatio-temporal patterns generated by the transition from a 2-D 5-site rule1, shown in Figure 7, to another 2-D 5-site rule2 when $p = 0.8$. For simplicity, the transition table in not illustrated in the paper. In comparison with the patterns produced by the evolution of rule1 in Figure 7 where $p = 1$, it can be observed that the dynamic noise has had substantial impact on the growing patterns. The growth of patterns in Figure 9 is 100 percent faster than the growth in Figure 7 due to the noise.

(a) t=0 (b) t=1 (c) t=10 (d) t=20 (e) t=30 (f) t=40 (g) t=50 (h) t=60 (i) t=70 (j) t=80

Figure 9: Transition from 2-D CA rule1 to rule2 with $p = 0.8$

The search result is shown in Figure 8 (b). Although the data are contaminated by a dynamic noise with density $1 - 0.8 = 0.2$ for the rule transition and the error does not converge to zero, the Boolean rule obtained is still the same as extracted in the noise free case. This suggests that the GA search method is robust in the presence of dynamic noise even for the two-dimensional case.

5 Conclusions

A solution to the inverse problem in cellular automata has been proposed using a multi-objective evolutionary algorithm. Both one- and two-dimensional CA's have been investigated and it has been shown that the CA rule, in the form of a parsimonious Boolean expression, can be identified by carefully formulating the GA search procedure. The simulation results illustrate the efficiency of the new algorithm and demonstrate that the correct neighbourhood and CA rule can be determined in the presence of both static and dynamic noise.

6 Acknowledgment

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

References

[1] J.von Neumann, "The general logical theory of automata ", in *Cerebral Mechanisms in Behavior – The Hixon Symposium*, New York: Wiley, 1951.

[2] L.Brieger and E.Bonomi, "A Stochastic Cellular Automaton Simulation of the Non-linear Diffusion Equation ", *Physica D*, vol.47, no.1-2, pp.159-168, 1992.

[3] M.Casolino, and P.Picozza, "A Cellular Automaton to Filter Events in A High-energy Physics ", *Nuclear Instruments and methods in Physics Research Section A – Accelerations Spectrometers Detectors and Associated Equipment*, vol.364, no.3, pp.516-523, 1995.

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

[5] S.Bhattachariee, S.Sinha and etc. "Cellular Automata Based Scheme for Solution of Boolean Equations ", *IEE Proceedings: Computers and Digital Techniques*, vol.143, no.3, pp.174-180, 1996.

[6] S.Surka and K.P.Valavanis, "A Cellular Automata Model for Edge Relaxation ", *Journal of Intelligent and Robotic Systems*, vol.4, no.4, pp.379-391, 1991.

[7] H.Gutowitz, "Cellular Automata and the Sciences of complexity (Part I and II) ", *Complexity*, vol.1, no.5, pp.16-22; 29-35, 1996.

[7a] H.Gutowitz, *Cellular Automata: Theory and Experiment*. MIT Press, 1991.

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

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

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

[11] N.Packard, "Cellular automaton models for dendritic crystal growth ", Institute for Advanced Study, preprint 1985.

[12] Z.Michalewicz, *Genetic Algorithms*, Springer – Verlag, 1994.

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

[14] A.J.Chipperfield and P.J.Fleming, "Parallel genetic algorithms: a survey ", Research Report No.518, Department of Automatic Control and Systems Engineering, University of Sheffield, 1994.