[78] G. K. I. Mann, B. G. Hu, and R. G. Gosine, "Analysis of direct action fuzzy PID controller structures," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, no. 3, pp. 371–388, 1999.
[79] Y. Tang, N. Zhang, and Y. Li, "Stable fuzzy adaptive control for a class of nonlinear systems," *Fuzzy Sets Syst.*, vol. 104, pp. 279–288, 1999.
[80] K. Fischle and D. Schroder, "An improved stable adaptive fuzzy control method," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 1, pp. 27–40, 1999.
[81] H. X. Li, "Approximate model reference adaptive mechanism for nominal gain design of fuzzy control system," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, no. 1, pp. 41–46, 1999.
[82] S. D. Wang and C. H. Lee, "Fuzzy system modeling using linear distance rules," *Fuzzy Sets Syst.*, vol. 108, pp. 179–191, 1999.
[83] L. X. Wang, "Automatic design of fuzzy controllers," *Automatica*, vol. 35, pp. 1471–1475, 1999.
[84] R. Ordonez and K. M. Passino, "Stable multi-inout multi-output adaptive fuzzy/neural control," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 3, pp. 345–353, 1999.
[85] S. G. Cao, N. W. Rees, and G. Feng, "Analysis and design of fuzzy control systems using dynamic fuzzy state space models," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 192–200, 1999.
[86] S. H. Zak, "Stabilizing fuzzy system models using linear controllers," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 236–240, 1999.
[87] E. Kim, H. J. Kang, and M. Park, "Numerical stability analysis of fuzzy control systems via quadratic programming and linear matrix inequalities," *IEEE Trans. Syst., Man, Cybern. A*, vol. 29, no. 4, pp. 333–346, 1999.
[88] M. Vidyasagar, *Nonlinear Systems Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[89] H. K. Lam, F. H. F. Leung, and P. K. S. Tam, "Design of stable and robust fuzzy controllers for uncertain multivariable nonlinear systems," in *Stability Issues in Fuzzy Control*, J. Aracil, Ed. New York: Springer-Verlag, pp. 125–164.
[90] T. A. Johansen, "Fuzzy model based control: Stability, robustness and performance issues," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 3, pp. 221–234, 1994.
[91] T. A. Ray, T. A. Ananda, and T. A. Majunder, "L_2 stability and the related design concept for SISO linear systems associated with fuzzy logic controllers," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-14, no. 6, pp. 932–939, 1984.

Neighborhood Detection and Rule Selection from Cellular Automata Patterns

Yingxu Yang and S. A. Billings

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

***Index Terms*—Cellular automata, genetic algorithms, identification, spatio-temporal systems.**

I. Introduction

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

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

In this paper, a new neighborhood detection technique is introduced which is capable of extracting the correct and minimal neighborhood from a large set of candidate neighborhoods without having to define

Manuscript received April 30, 1999; revised July 14, 2000. This work was supported by the University of Sheffield and UK EPSRC. This paper was recommended by Associate Editor W. Pedrycz.

The authors are with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield S1 3JD, U.K.

Publisher Item Identifier S 1083-4427(00)09030-5.

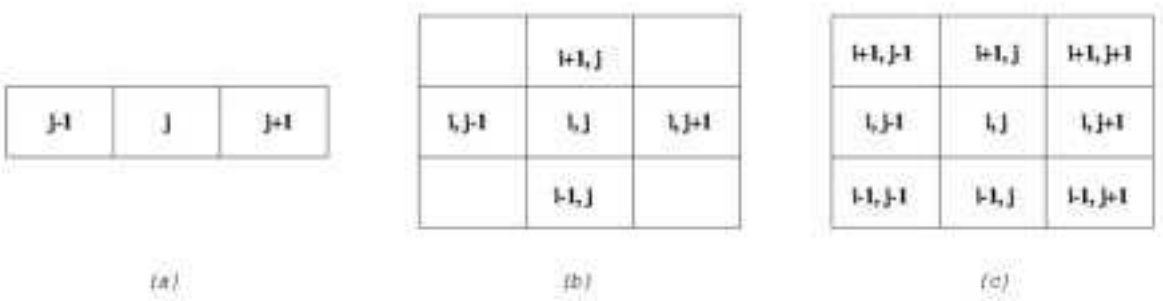

Fig. 1. Examples of some of the most frequently used 1-D and 2-D neighborhoods: (a) the 1-D von Neumann neighborhood; (b) the 2-D von Neumann neighborhood; and (c) the 2-D Moore neighborhood.

the rule at the same time. The algorithm is simple and easy to implement and forms the first stage in a CA identification procedure. A simple GA is then employed, starting with the obtained neighborhood, to search for the best matching local rule. Separating the neighborhood detection from the rule construction provides a new approach to CA identification that overcomes many of the limitations of earlier methods which were often restricted to low-dimensional systems.

II. Neighbourhood Detection

The notation and background to CA neighborhoods and rules is briefly introduced below.

A. Cellular Automata

A cellular automaton is composed of three parts: a discrete lattice, a neighborhood and a rule for local transitions. All cells in the lattice are updated synchronously according to the local rule. The neighborhood of a cell is the group of the cells which are able to directly affect the evolution. Some of the most frequently used neighborhoods are illustrated in Fig. 1. For simplicity, this paper only considers neighborhoods composed of cells from time step $t-1$, but the results are not restricted to this case. There are various representations for a CA rule. In this paper, Boolean expressions will be considered. From [9], every CA rule with an n site neighborhood $\{cell(x_1), \cdots, cell(x_n)\}$ can be written as

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

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

B. Neighborhood Detection

Define the vector XOR operator $\oplus$ as

$$[b_1 \quad b_2 \quad \cdots \quad b_n] \oplus \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_n \end{bmatrix} = (b_1 * c_1) \oplus (b_2 * c_2) \oplus \cdots \oplus (b_n * c_n)$$

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

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

where

$$\mathbf{a} = [a_1 \quad a_2 \quad \cdots \quad a_P]^T \quad \text{and}$$
$$\mathbf{s} = [1 \quad s(x_1) \quad \cdots \quad s(x_n) \quad s(x_1) \times s(x_2) \quad \cdots \quad s(x_1) \times \cdots \times s(x_n)].$$

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

$$s_{new}(x_j) = \mathbf{s} \times \overline{\mathbf{a}} \quad (3)$$

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

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

Equation (3) can be written as

$$\mathbf{s}_{new} = \mathbf{S} \times \overline{\mathbf{a}} \quad (4)$$

where

$$\mathbf{s}_{new} = [s_{new}(x_j(1)) \quad s_{new}(x_j(2)) \quad \cdots \quad s_{new}(x_j(N))]^T$$
$$\mathbf{S} = [\mathbf{s}^T(1) \quad \mathbf{s}^T(2) \quad \cdots \quad \mathbf{s}^T(N)]^T = [\mathbf{s}_1 \quad \cdots \quad \mathbf{s}_P]$$
$$\mathbf{s}^T(t) = [1 \quad s(x_1(t)) \quad \cdots \quad s(x_n(t)) \quad s(x_1(t)) \times s(x_2(t)) \quad \cdots \quad s(x_1(t)) \times \cdots \times s(x_n(t))]$$

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

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

is an orthogonal matrix

$$\mathbf{E}^T \times \mathbf{E} = \text{Diag}\,[\mathbf{e}_1^T \times \mathbf{e}_1 \quad \cdots \quad \mathbf{e}_P^T \times \mathbf{e}_P]$$

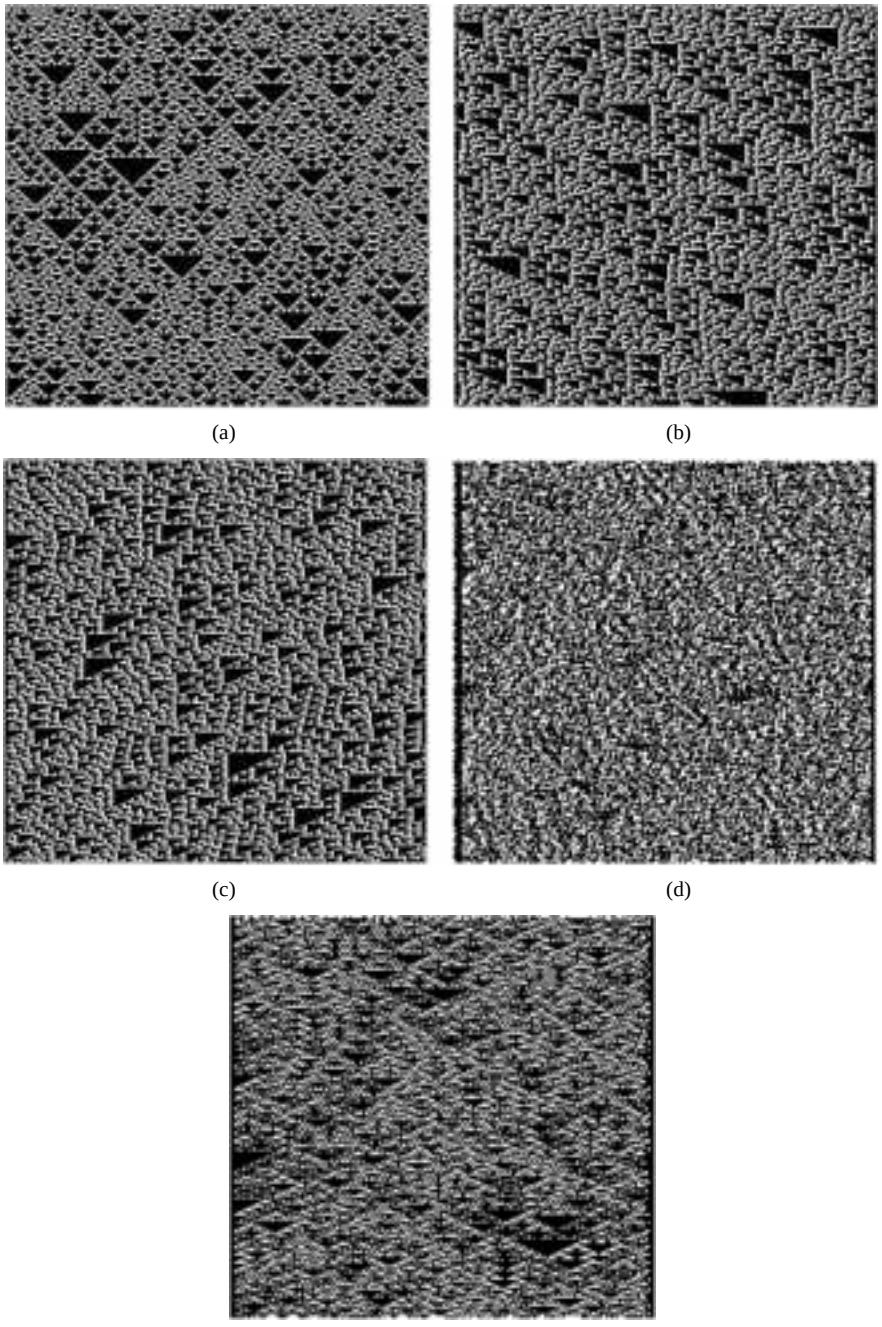

Fig. 2. Evolution of 1-D CA $Rule22$ on various 3-site neighborhoods.

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

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

Equation (4) can then be represented as

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

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

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

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

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

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

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

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

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

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

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

$$[ct]_2^{(i)} = \frac{(\tilde{a}_2^{(i)})^2 \times (\mathbf{e}_2^{(i)})^T \times \mathbf{e}_2^{(i)}}{\mathbf{s}_{new}^T \times \mathbf{s}_{new.}}$$

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

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

III. Rule Selection

The correct and minimal neighborhood can be detected from the set of terms selected using the procedure in Section II-B (details are presented in Section IV-A). Denote the neighborhood obtained as $\{cell(x_{n_1}), \cdots, cell(x_{n_2})\}$, then the Boolean form of the rule to be identified can be written as

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

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

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

A. The Population

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

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

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

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

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

B. The Evaluation Function

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

The evaluation function

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

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

C. The Reproduction Process

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

When the GA is run, the population, evaluation function and reproduction process work in combination to affect the evolution in the search for a matching CA rule. After initializing the population, each chromosome is evaluated by the evaluation function and a series of cycles of replacing the current population by a new population begins. In each cycle, the evaluation, the parent selection, the genetic operation and the insertion of the new population to replace the old population are performed sequentially. The search process terminates when all chromosomes in the new population converge to a single individual.

Compared to the algorithm in [9] where a multi-objective GA with subpopulations was employed, the GA in this paper is much simpler and easier to implement. The pre-determined minimal neighborhood which is obtained in the neighborhood detection procedure enables the second search objective to minimize the neighborhood structure in the GA to be discarded. It is also possible to eliminate the subpopulations designed for the multi-objective approach of the earlier method. These simplications will therefore considerably accelerate the rule identification process.

IV. Simulation Studies

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

A. Neighborhood Detection

1) Spatio-temporal patterns produced by 1-D CA $Rule22$ on various 3-site neighborhoods: The spatio-temporal patterns produced by 1-D CA $Rule22$ on various 3-site neighborhoods are shown in Fig. 2. All of these were developed on a 200×200 lattice with time evolution from top to bottom and a periodic boundary condition. That is the lattice is taken as a circle in the horizontal dimension, so the first and last sites are identified as if they lay on a circle of finite radius. The evolution started from an initial condition of a randomly generated binary vector. The neighborhoods of $cell(j)$ for $(a) - (e)$ are $\{cell(j-1), cell(j), cell(j+1)\}$, $\{cell(j-2), cell(j-1), cell(j)\}$, $\{cell(j), cell(j+1), cell(j+2)\}$, $\{cell(j-4), cell(j-1), cell(j+3)\}$, and $\{cell(j-2), cell(j), cell(j+2)\}$, respectively. Although the patterns were all produced under the same rule, Fig. 2 clearly shows the diversity induced by the different neighborhoods associated with the rule. Fig. 2(a) was produced by the symmetric von Neumann neighborhood and is therefore composed of inverted symmetric triangles of varying sizes. However, the triangle structures in Figs. 2(b) and (c) only represent the left and right half of the triangles

in Fig. 2(a). This is due to the left- and right-shift nature of the corresponding neighborhoods $\{cell(j-2), cell(j-1), cell(j)\}$ and $\{cell(j), cell(j+1), cell(j+2)\}$. The irregularity of the neighborhood $\{cell(j-4), cell(j-1), cell(j+3)\}$ produced the blurred and twisted triangles in Fig. 2(d). The increase of the distance between neighboring cells in the neighborhood $\{cell(j-2), cell(j), cell(j+2)\}$ (compared to $\{cell(j-1), cell(j), cell(j+1)\}$) is clearly illustrated in Fig. 2(e) where the triangles are flattened.

2) Neighborhood Detection of 1-D CA $Rule22$: Assume initially that the largest possible neighborhood is a 9-site neighborhood defined by $\{cell(j), cell(j-4), cell(j-3), cell(j-2), cell(j-1), cell(j+1), cell(j+2), cell(j+3), cell(j+4)\}$. Define the neighborhood vector $\mathbf{nei}$ as

$$\mathbf{nei} = [cell(j) \quad cell(j-4) \quad cell(j-3) \quad cell(j-2) \quad cell(j-1) \quad cell(j+1) \quad cell(j+2) \quad cell(j+3) \quad cell(j+4)]^T.$$

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

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

where 1, 2, 3, 4, 5, 6, 7, 8, and 9 denote the elements in $\mathbf{nei}$. For instance, entry 5 represents the fifth element in $\mathbf{nei}$ and is therefore associated with $cell(j-1)$, and so on. The full SET consists of $P = 2^9 - 1 = 511$ terms/rows. Each row in SET represents a candidate term which corresponds to an $\mathbf{s}_i, i = 1, \cdots, P$ in matrix $\mathbf{S}$ in (4) in Section II-B. For instance, the first row (1 0 0 0 0 0 0 0 0) represents $s(j)$ only while the last row (1 2 3 4 5 6 7 8 9) corresponds to a product of nine states $s(j) \times s(j-4) \times s(j-3) \times s(j-2) \times s(j-1) \times s(j+1) \times s(j+2) \times s(j+3) \times s(j+4)$.

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

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

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

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

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

Data extracted from the spatio-temporal patterns in Figs. 2(a)–(e) were used in the neighborhood detection. For each pattern, 1000 data points were used and the tolerance value C_{off} was set as 0. Applying the neighborhood detection technique in Section II-B to each pattern produced matrices (a)-(e). The last term on each row in each matrix represents the contribution $[ct]_i, (i = 1, \cdots, 7)$ the term on the same row makes to $s_{new}(j)$. The maximum contribution is 1.0 so multiplying the last term in each row by 100 would give the percentage contribution of the term on the same row.

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

Other 1-D CA rules with neighborhood sizes larger than three were also tested using the same procedure and again these confirm the validity of the algorithm and the results are all correct. For simplicity the results are not presented in the paper.

3) Neighborhood detection of a 2-D CA Rule: Data extracted from the spatio-temporal patterns produced by the evolution of the two-dimensional (2-D) $Rule$(01 011 111 10 100 011 01 001 001 01 000 000) on the 2-D von Neumann neighborhood $\{cell(i-1,j), cell(i,j-1), cell(i,j), cell(i,j+1), cell(i+1,j)\}$ will be used to illustrate the neighborhood detection procedure for the 2-D case. The initial neighborhood was assumed to be a 2-D 9-site Moore Neighborhood defined by $\{cell(i,j), cell(i+1,j-1), cell(i+1,j), cell(i+1,j+1), cell(i,j-1), cell(i,j+1), cell(i-1,j-1), cell(i-1,j), cell(i-1,j+1)\}$, and the neighborhood vector $\mathbf{nei}$ was defined as

$$\mathbf{nei} = [cell(i,j) \quad cell(i+1,j-1) \quad cell(i+1,j) \quad cell(i+1,j+1) \quad cell(i,j-1) \quad cell(i,j+1) \quad cell(i-1,j-1) \quad cell(i-1,j) \quad cell(i-1,j+1)]^T.$$

TABLE I
SUMMARY OF RESULTS OBTAINED IN EVOLVING SOME 1-D CA RULES WITH VARIOUS SIZES OF NEIGHBOURHOODS USING GA

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

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

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

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

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

B. Identification of the CA Rules

1) Selection of 1-D CA rules: The genetic algorithm described in Section III will be used to identify the CA rules based on the neighborhood structure which was obtained from the neighborhood detection algorithm. The GA search was tested over a large set of 1-D CA rules with neighborhoods of various sizes which were identified in Section IV-A2. Some of the results are shown in Table I. For each rule, 100 trials were conducted with different initial populations. The search was terminated when 400 generations had been reached. For simplicity only the average and standard deviation (*std.dev.*) values are listed in Table I.

For $Rule22$ the data points for the GA search can be extracted from any of the five spatio-temporal patterns in Fig. 2. This is possible because all the patterns were produced under $Rule22$, although over different neighborhoods. Now the neighborhoods have been determined it will not make any difference which pattern is used in the GA search. This also applies to the other rules. Assume the neighborhood for $cell(x_2)$ is $\{cell(x_1), cell(x_2), cell(x_3)\}$, the Boolean form of $Rule22$ is then identified as $s_{new}(x_2) = s(x_1) \oplus s(x_2) \oplus s(x_3) \oplus (s(x_1) * s(x_2) * s(x_3))$. In Table I, only rules with small neighborhoods are enumerated. This is due to the fact that the numerical label and the truth table form of the rules can be very cumbersome when the neighborhood size is larger than 4. Each identified rule in Table I produces a correct truth table which, together with the pre-detected minimal neighborhood, defines a minimal and correct Boolean rule for the corresponding spatio-temporal pattern.

It can be seen from Table I that the average run time depends largely on the size of the neighborhood. For each rule, the average run time in Table I is considerably smaller than in [9, Table VI], where without using the neighborhood detection algorithm the solutions had to be selected from a substantially larger set of possible rules. For example

TABLE II
THE TABULAR FORM OF THE IDENTIFIED 2-D BOOLEAN RULE

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

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

TABLE III
SUMMARY OF RESULTS OBTAINED IN EVOLVING THE 2-D CA RULE WITH GA

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

for the 3-site 1-D rules the rule set for the GA search without the initial neighborhood detection algorithm would comprise a massive $2^{2^9} = 1.3408e + 154$ rules. In comparison, the rule set for the GA search based on the algorithm introduced in this study and which generated the results in Table I consisted of a substantially reduced search over just $2^{2^3} = 256$ rules. In addition since the pre-defined neighborhood is minimal the GA used above evolves with consideration of only one objective, to minimize the matching errors, while the GA without neighborhood detection involves a second search objective, to minimize the structure of the neighborhood.

2) Selection of 2-D CA rules: The neighborhood obtained in Section IV-A3 will be used for the GA search of the 2-D CA rule in Section IV-A3. A total of 100 trials were tested with different initial assignments. For each trial the search was terminated after 400 generations.

The tabular form of the identified Boolean rule is shown in Table II. The search results are shown in Table III. The Boolean rule is the $\oplus$ combination of the selected 12 terms from B to M. As can be seen from the second column in Table II, which matches the rule definition (01 011 111 10 100 011 01 001 001 01 000 000) exactly, this truth table is correct. Therefore, together with the pre-identified minimal neighborhood, the $\oplus$ combination of B to M defines the desired correct and minimal Boolean rule. The average run time in Table III is very similar to the the run time for 5-site 1-D rules in Table I and is considerably shorter than the time in [9, Table IV]. Again this is because the neighborhood detection algorithm was used to predetermine the neighborhood for a full GA search. The main element that determines the search time is the size of the neighborhood rather than the position occupied by each cell in the neighborhood. This means that the dimensionality of the CA does not have a crucial impact on the search. A GA search over a large set of 2-D rules was conducted using the results from the neighborhood detection routine. Because of the insensitivity to

the dimensionality of CA, the results were very similar to the 1-D case in Table I and are not listed in the paper.

V. Conclusions

A new CA identification technique has been introduced which breaks the identification of CA rules from given patterns of data into two problems. First a new neighborhood detection procedure is used to establish the correct neighborhood. Then using this identified neighborhood a GA search is conducted to determine the minimal CA rule.

The new approach can yield significant improvements in efficiency. For example a full GA search for a 3-site 1-D rule would comprise a search over a huge $2^{2^9} = 1.3408e + 154$ possible rules. But by using the neighborhood detection procedure to prune the GA search this can be reduced to a search over just 2^{2^3} rules. Simulation results for both 1-D and 2-D CAs clearly demonstrate the performance of the new algorithm.

Acknowledgment

The work of Y. X. Yang was supported by a scholarship from the University of Sheffield. The work of S. A. Billings was supported in part by EPSRC.

References

[1] O. Lafe, "Data compression and encryption using cellular automata," *Eng. Applicat. Artif. Intell.*, vol. 10, no. 6, pp. 581–591, 1997.
[2] G. Hernandez and H. J. Herrann, "Cellular automata for elementary image enhancement," *Graph. Models Image Process.*, vol. 58, no. 1, pp. 82–89, 1996.
[3] R. Cafiero *et al.*, "Disordered one-dimensional contact process," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 57, no. 5, Pt. A, pp. 5060–5068, 1998.
[4] R. B. Pandey, "Stochastic cellular automata approach to cellular dynamics for HIV: Effect of viral mutation," *Theory in Biosciences*, vol. 117, no. 1, pp. 32–41, 1998.
[5] K. Lindgren *et al.*, "Complexity of two-dimensional patterns," *J. Stat. Phys.*, vol. 91, no. 5–6, pp. 909–951, 1998.
[6] E. C. Monteiro *et al.*, "A cellular automaton computer model for the study of magnetic detection of cardiac tissue activation during atrial flutter," *IEEE Trans. Magn.*, vol. 34, no. 5, Pt. 1, pp. 3451–3454, 1998.
[7] A. I. Adamatzky, *Identification of Cellular Automata*. New York: Taylor and Francis, 1994.
[8] F. C. Richards, "Extracting cellular automaton rules directly from experimental data," *Physica D*, vol. 45, pp. 189–202, 1990.
[9] 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.
[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1994.
[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[12] A. J. Chipperfield and P. J. Fleming, "Genetic algorithms in control systems engineering," *Control Comput.*, vol. 23, pp. 88–94, 1996.

Texture Classification Using Rotated Wavelet Filters

Nam-Deuk Kim and Satish Udpa

***Abstract*—In this paper, we propose a novel approach to the texture classification problem using a new set of two-dimensional (2-D) wavelet filters that are nonseparable and oriented for improved characterization of diagonally oriented textures. Channel energies are estimated at the output of both the new filter bank and a standard discrete wavelet frames (DWF) filter bank. Classification results obtained using each individual method and in combination are presented. The results show that the oriented filter set results in finer discrimination providing complementary texture information to the DWF by making use of its orientation selectivity. As a result, a combination of the features from the output of two filter banks improved the classification accuracy significantly with a smaller number of features.**

***Index Terms*—Discrete wavelet frames, rotated wavelet filters, texture classification.**

I. Introduction

The past few decades have witnessed a substantial interest in the analysis of textured images. The interest has been motivated in large part by the huge number of applications in such diverse areas as remote sensing, robot vision, medical image analysis, and quality inspection. While considerable research has been carried out in the texture analysis domain, the problems related to texture processing have still only partially been solved and active research is continuing.

Among the many texture analysis algorithms proposed in the literature, statistical approaches [1] and model-based approaches [2], [3] are some of the more commonly used methods. Recently, multiscale filtering methods have shown significant potential for texture classification and segmentation [4]–[9], where advantage is taken of the spatial-frequency concept to maximize the simultaneous localization of energy in both spatial and frequency domains. The notion of spatial-frequency analysis is also supported by experimental research on human and mammalian vision [10]. Multiscale filtering approaches were originally motivated by Gabor-filter models of neural architecture in the visual cortex [4]. The approach to texture analysis is intuitively appealing because it allows us to exploit differences in dominant sizes and orientations of different textures.

In this paper, we propose a novel approach to the texture classification problem using a new set of two–dimensional (2-D) wavelet filters that are nonseparable and oriented for improved characterization of diagonally oriented textures. Channel energies are estimated at the output of both the new filter bank and a standard discrete wavelet frame (DWF) filter bank. Twenty-eight textures are used in the classification experiment using a Bayes classifier. The results show that the information provided by the oriented filterbank can be used to complement information generated by a standard DWF filterbank. The discrimination performance obtained with the combined filterbank is superior relative to the performance obtained using the DWF filterbank alone.

In the next section, we briefly review the wavelet theory and summarize prior work on the application of wavelet-based algorithms to texture analysis applications. Then, we introduce the newly designed rotated wavelet filter set in Section III. Section IV describes the experimental design and classification algorithm. In Section V, we present the

Manuscript received August 11, 1999; revised July 17, 2000. This paper was recommended by Associate Editor M. S. Obaidat.

The authors are with the Material Assessment Research Group, Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA.

Publisher Item Identifier S 1083-4427(00)09029-9.