Problem solving based on error minimization in cellular systems with phase fields instead of connections

Philip Van Loocke
Lab Applied Epistemology, University of Ghent
Blandijnberg 2, 9000 Ghent, Belgium
philip.vanloocke@rug.ac.be

Abstract
It is explained how a cellular automaton can grow patterns that correspond to trained networks. Since a pattern corresponds to a map between an n-dimensional and an m-dimensional space, such a pattern can be called a 'meta-pattern'. The problems solved by connectionist multi-layered networks can be solved by the automaton too. In addition, it allows for a straightforward representation of patterns with internal bindings, even if such bindings are organized at several, hierarchically related levels. Further, if a problem has symmetry, then the form corresponding to its solution usually is aesthetically attractive. This contrasts with the black-box nature attributed to the classical connectionist approach. Since problems with symmetry are often called 'beautiful problems', the present system gives beautiful solutions for beautiful problems.
Keywords: Cellular automaton, binding problem, connectionism, growth, visualization

1. Introduction

Fractals and cellular automata enjoyed a pretty large amount of scientific attention during the past two decades (Gutowitz, 1991). They showed beautiful patterns to a relatively wide audience, and the mathematical as well as numerical analysis of these patterns has been eagerly developed. In case of fractals, the relation with chaos theory resulted also in a link with physical, social, ecological, and other types of complex systems. However, in artificial intelligence, the influence of fractals and even of cellular automata has remained relatively weak. Other approaches, such as connectionism, or fuzzy rule systems, have been applied much more often. But in this paper, the following claims are put forward:

a. Growth processes in cellular automata are able to carry out complex learning tasks, comparable to the ones carried out by the famous backpropagation algorithm.

b. The solution of a complex learning task that is found by the cellular automaton is a pattern with fractal properties. Hence, a pattern plays the role of a trained network. If the task has symmetry, then the pattern is often of a special aesthetic quality.

c. This pattern, in turn, can be memorized in a connectionist way. Since information with arbitrarily complex internal structure can be mapped on a network, the automaton can memorize structured information in a content addressable way.

These points show that cellular automata and fractals do have to offer something to cognitive science, in special because connectionism is still struggling with point c.

The paper is organized as follows. In section 2, we introduce the basic concepts of the automaton. Section 3 shows the resulting forms for a familiar non-linear problem. Finally, section 4 applies the method to a problem in which a system has to learn information with complex inner structure.

2. Phase fields and growth in cellular automata

In a connectionist network, some cells are input units, some are output units, and some are none of this, the so-called hidden units. The parameters that are adapted during a learning process are the connections between the cells. In the popular backpropagation algorithm, the learning process does not recruit new cells, but the connection matrix between the cells is adapted in its entirety at each step during the learning process (Rumelhart and Mc Clelland, 1986). These properties contrast with the cellular automaton approach of the present paper. All cells of the automaton are input as well as output cells. They all receive the same input, but their outputs are usually different. A simple operation turns the outputs of the individual cells into the output of the total network. There are no weights to be adapted. Rather, every cell in the automaton corresponds to a fixed combination of weights, and a growth process determines if a cell is included in the solution of a problem. Phase fields determine the weight parameters in the cells.

A phase field is a scalar field that has a value in every cell of the automaton. Suppose that the network has to solve a learning task in which an m-dimensional space is mapped on an m-dimensional one. Then, there are n+m phase fields defined over the automaton, one for every input dimension, and one for every output dimension. The i-th phase field corresponding to an input dimension is denoted $f_i(x,y)$, where (x,y) are the coordinates of a cell. A phase field has an origin. If (x_i,y_i) is the origin of f_i, then the explicit expression for f_i is given by: $f_i(x,y)=\cos(d2\pi/c)$, where d is the Euclidean distance between (x_i,y_i) and (x,y). Similarly, the j-th output phase field is given by $g_j(x,y)=\cos(d2\pi/e)$, where d is the Euclidean distance between (x,y) and the origin of the field (x_j,y_j); c and e are system parameters. The fields f_i and g_j are called phase fields because they are not modulated by an amplitude that decays with distance.

Suppose that a problem has a single-dimensional output space (so that m=1), and suppose that an input vector is denoted $(a_1,a_2,\ldots,a_n)$. This input vector is presented to all units of the automaton. In every unit, an output value is determined in accordance with the expressions:

$$net = \Sigma f_i.a_i \quad (1)$$

and

$$o = -1+2/(1+\exp(-\alpha.net)) \quad (2)$$

These expressions are familiar from connectionism. They mean that every unit performs a linear operation on the input, followed by a strongly non-linear squashing operation that draws the output between –1 and +1. A cell can be active or not active. In both cases, the output is determined by eq. 2, but only active cells contribute to the output of the entire automaton. In order to obtain this output, the outputs of the cells that are active are multiplied with a constant number w, and subsequently summed. The collection of active cells is called a form 'F'. Hence, the output of the entire automaton is:

$$o_F=\Sigma_{\{C \text{ in } F\}}w.o_C \quad (3)$$

If the output space has m dimensions, with m>1, then the quantity o in a unit is multiplied with all fields g_j, so that m outputs become associated with an active cell. The j-th output value associated with a form F is given by

$$o^j_F=\Sigma_{\{C \text{ in } F\}}w.o.g_j \quad (4)$$

These elementary concepts are sufficient to explain how the automaton carries out its learning processes. Consider, like in connectionism, a supervised learning task. Suppose that a training set has N input-output pairs, and that there is a test set with M such couples. The learning process in the automaton corresponds to the growth of a form. The form starts in a singe cell. This cell is included in the form, and the effective outputs produced for the training couples determine the initial value of the error function associated with the form. Like in connectionism, the error function can be determined as the summed squared differences between effective and target outputs for all training couples and for all output dimensions. Then, the square outside boundary of the starting cell is considered. The cells of this outside boundary are examined one after another. If the inclusion of a cell in the form leads to a lower value for the error function, the cell is effectively included in F. Subsequently, the outward boundary of the square with side 3 is considered, and so on. Gradually, the square with inspected cells grows, and so does F. Like in connectionism, the growth is stopped when further inclusion of cells in F leads to an increase or stagnation of the value for the error on the test set.

3. Application of the method to a concrete problem

Consider the 10-dimensional parity problem. This problem defines a mapping from a 10-dimensional space to a 1-dimensional space. The input-vectors considered are confined to vectors with binary components with values 0 and 1. The output generated by an input vector is equal to 1 if it has an odd number of 1's; else the output is 0. The problem is highly non-linear, since vectors that differ by a single value only are mapped on opposite outputs. For this reason, it is a tedious affair to get a neural net trained for parity problems with high-dimensional input spaces.

The first step in the application of the method of section 2 is to put the origins of the phase fields on the plane in which the form is grown. Suppose that the origins of the phase fields are put on the edges of a pentagon with a side of 750 cells, and that the following system parameters are chosen: α=10, c=45 and w=0.0002. Then, after 701 single-cell increases of the side of the square of inspected cells, the form of Figure 1 results. This form is a perfect solution of the parity problem. The error function decays from an initial value of 512 to a value of 9, and all inputs are mapped on the correct outputs.

The procedure of algorithmic growth can be subject to a number of variations. For instance, instead of inspecting new cells only, old cells can be reexamined at any stage in order to inspect if a change of their activation would lead to a decrease in the error function. This way, one usually obtains smaller forms that solve a problem. For instance, suppose that, every time the square has extended its side with a size of 20 cells, the older cells are reexamined, and that all other parameters are the same as in case of Figure 1. Then, one obtains the series of forms of Figure 2 after the square has a side of 101, 201 and 401 cells. The latter form is a perfect solution of the problem.

It is not indispensable to make recourse to a growth process in order to solve a problem with a form. One can also select a single cell randomly at every time step, and decide to include this cell if it leads to a decrease of the error function. A form that solves the 9-parity problem and that is generated in this way is given in Figure 3. Grown forms, however, have an important advantage in cognitive contexts. Suppose that one wants to save the patterns obtained for a series of problems in a connectionist network. This is easily realized by using, for instance, the Hebbian learning rule. In case of a growth procedure, forms have two relevant properties: i) they are deterministic and ii) related problems are solved by forms with large overlap. The latter property is important, since it allows that typical connectionist properties like prototype formation and content addressability can be put at work. Suppose that a set of problems is solved by forms with overlap. Suppose that a related problem is given to the network. Then, it can grow a form on basis of the set of input-output pairs corresponding to the problem, but it can also constrain the form corresponding to the problem by making recourse to forms that have been memorized. Connectionist prototype effects entail in this situation that common cores of forms will have an inclination to be present in solutions of related

problems. This may be of help, for instance, when a particular problem has to be solved for which the number of input-output pairs is relatively low (For more examples, for different field definitions, as well as applications of the method of section 2 to function approximation problems, I refer to Van Loocke, 1999a,b,c,d,e).

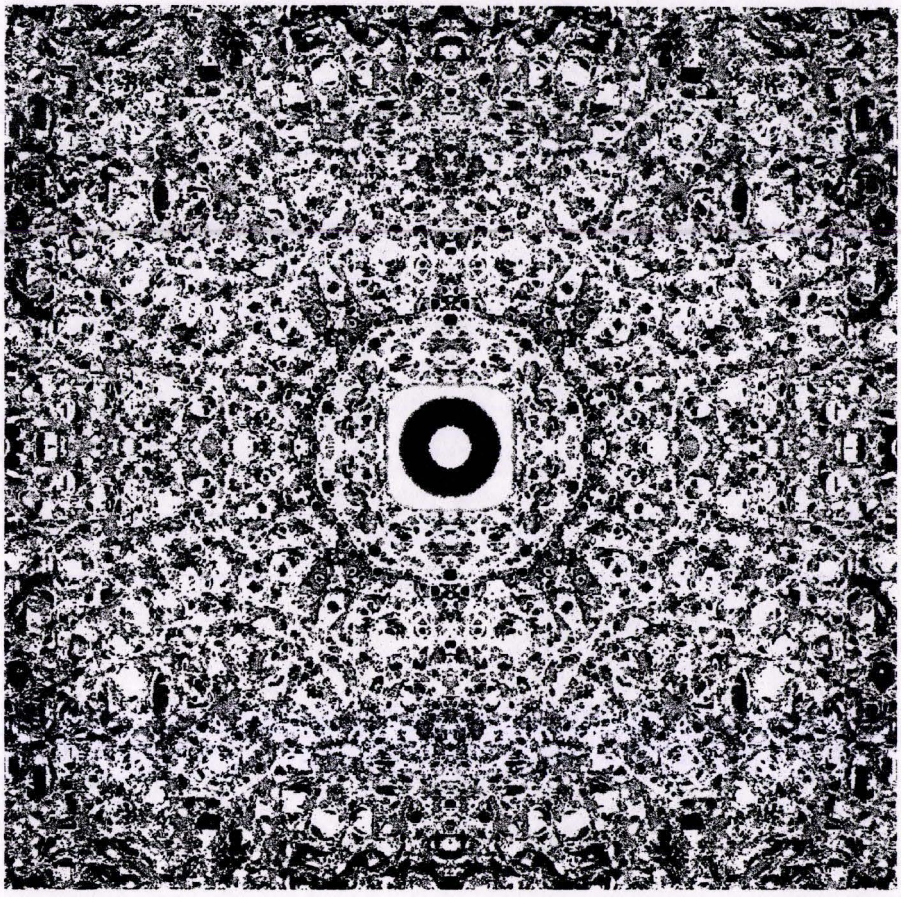

Fig. 1: Gray-scale values indicate the reduce in error by inclusion of an active cell in the form, or increase in error if a non-active cell would be included in the form

4. On the representation of information with complex inner structure

Consider a domain in which patterns have internal structure, such as grammar, or hierarchically organized visual systems. Whatever the inner structure of a pattern, it can always be mapped on a function. If the form corresponding to the function is grown,

then this form can be taken as the pattern that corresponds to the hierarchical structure, and it can be stored in the usual connectionist way in a network.

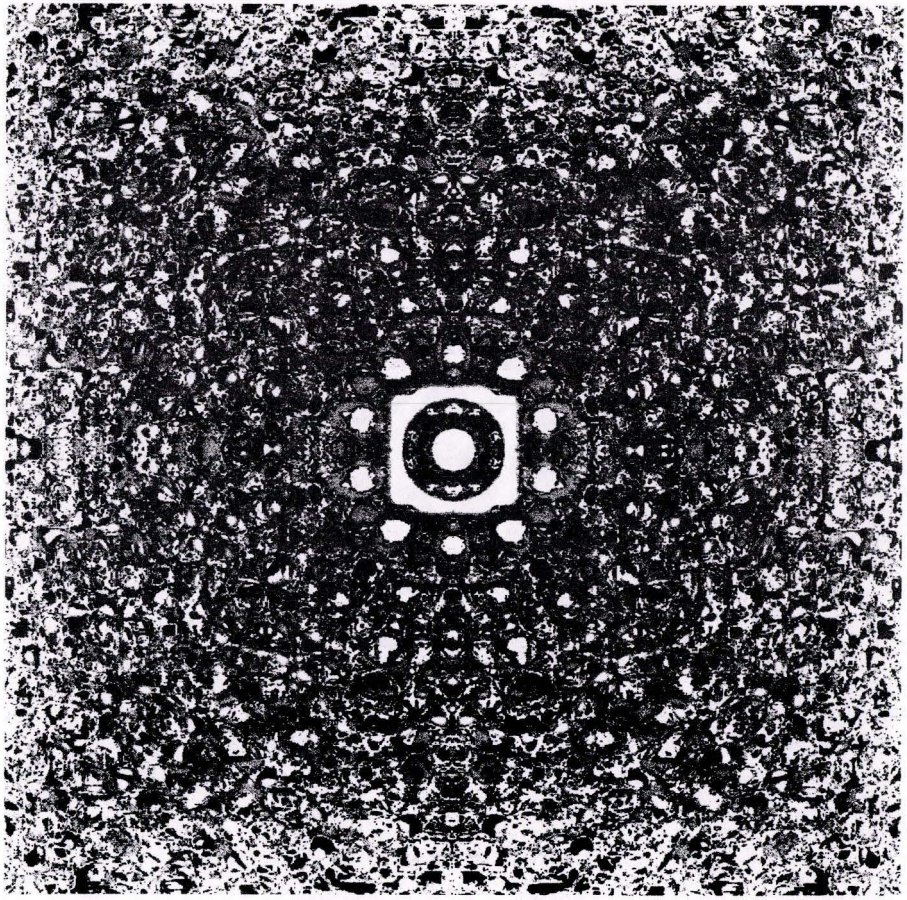

Fig. 2: Same conditions as in Fig. 1, but for reexamination of older parts

Suppose, for instance, that patterns have a tree-like hierarchical inner structure, and that this hierarchy is organized on k layers. For simplicity, it is assumed that every layer in

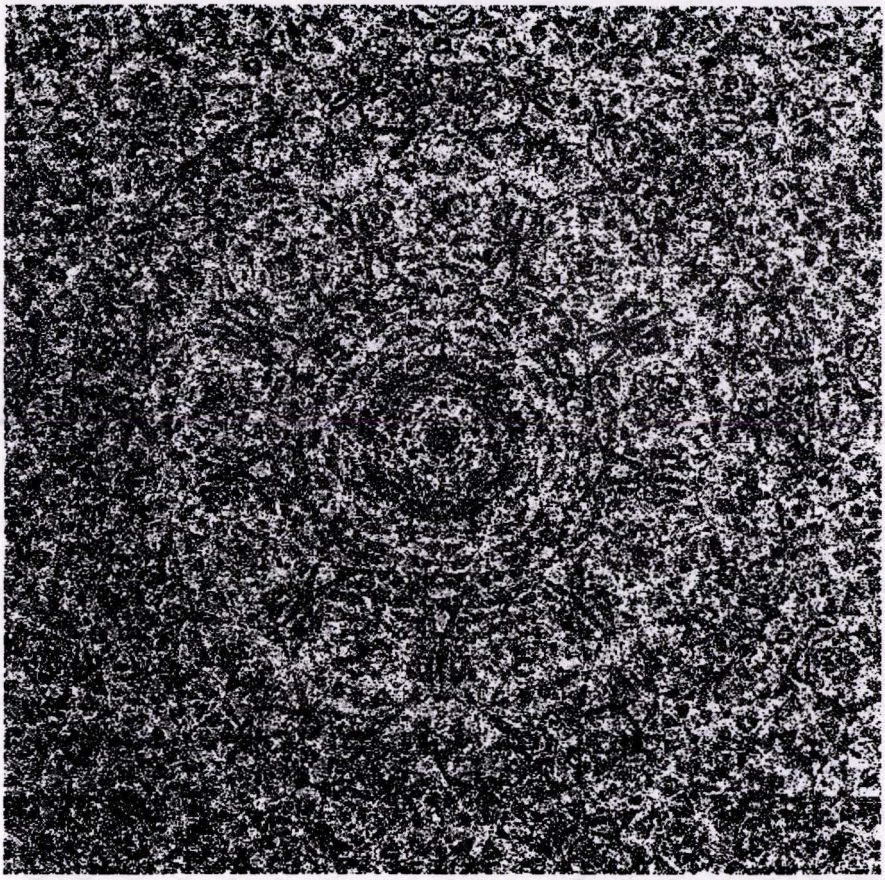

Fig. 3: Solution of 9-parity problem for random growth condition. Gray-scale values indicate if the cell is in the form or not

this hierarchy has q nodes that can have representational function. Then, a structure that is represented by a set of trees defined over these layers can always be mapped on a function with q.(k-1) couples. The first element of such a couple specifies a node; all nodes, except the ones of the top layer, correspond to a couple. The second element specifies the nodes of the next layer that receive a branch from the former node. More specifically, the first element of a couple is a vector with q+(k-1) components, and the second element is a q-dimensional vector. If p<=q, then the first vector has a component equal to 1 on the p-th and on the q+1-th dimension; all other components are equal to zero. If 2q>=p>q, then the first vector has value one on the dimensions p-q and q+2, and so on. The second vector is zero, except at the places that correspond to the nodes of the next layer that receive a branch from the unit identified by the first vector. Once the function has been identified, the method of the previous section can be

used in order to grow a form that approximates the function. Then, this form can be stored in the automaton in the usual connectionist way.

Consider, for example, the tree in Figure 4a. It has 5 layers of 9 elements, so that it can be mapped on a function with 36.(5-1)=36 couples. The first element of such a couple has 13 components ; the second element is a vector with 9 components. Hence, for this function, the input space has 13, and the output space with 9 dimensions. In the illustration that led to the form of Figure 5a, the phase fields corresponding to the first 9 input dimensions were put on a horizontal line 100 cells above the starting cell. The next four input dimensions were put at 50 and 150 cells above the starting cell, and 50 and 150 cells below the starting cell, respectively. The origins of the nine phase fields corresponding to the output dimensions were also put on a horizontal line. This time, the line was 100 cells below the starting cell. After the form had grown until it covered a square of 479 cells, the error had decreased form 38 to 2.1. Similarly, Figure 5b shows the form corresponding to the hierarchical structure in Figure 4b. We notice that these forms have high overlap (more exactly, their have an overlap of 0.77, notwithstanding their relatively low density). Hence, we observe that related hierarchical structures are mapped on forms with large overlap. Since, in connectionism, overlap between patterns is the basis of prototype effects, as well as of the formation of abstract cores or of conceptual taxonomies (Van Loocke, 1994; Rumelhart and Smolensky, 1986; Bechtel and Abrahamson, 1984), we see that the present method allows such effects to be created for patterns with complex hierarchically organized internal structure. This may mean, for instance, that sub-trees that are often present in a particular context have especially strong tendency to influence the flow of activation that relates to the inner structure of new patterns.

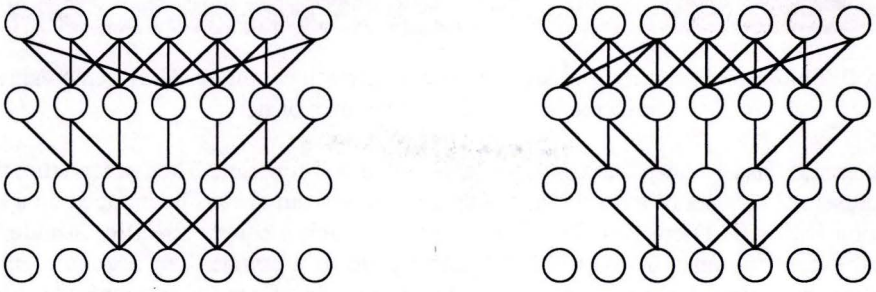

Fig. 4: Two trees to be transformed in activation patterns

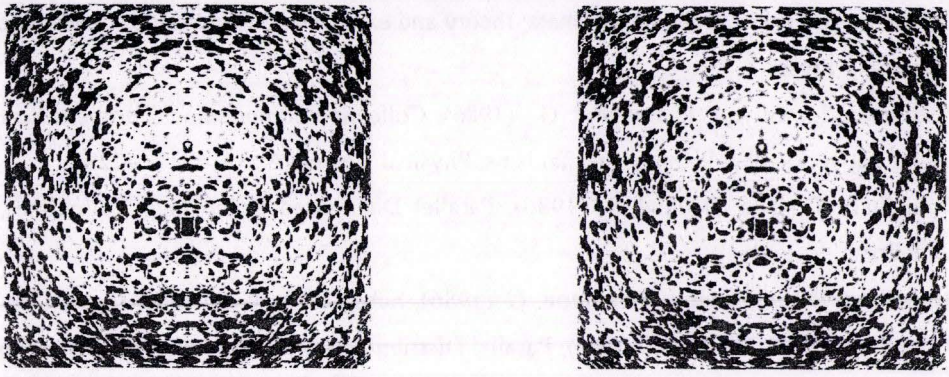

Fig. 5: Patterns of activation corresponding to the tree structures of Fig. 4

5. Discussion

The method explained shows that growth in cellular automata does have cognitive relevance. Not only can such growth solve problems that are traditionally presented to multi-layered neural networks, one of its striking features of is that a pattern of activation plays the role of a connection matrix in a connectionist context. Such a pattern of activation, in turn, can be stored. In this sense, a pattern of activation grown by the automaton is a meta-pattern : it is a pattern that defines a mapping between an n-dimensional and an m-dimensional space. Since information with arbitrarily complex inner structure can be represented as a mapping, the present method allows a network to store such information. Hence, the familiar connectionist properties of associative reasoning, prototype formation, pattern completion, and so on, can be realized for patterns with complex inner structure.

In addition, the present method gives aesthetic solutions for problems that have symmetry. Such problems are often called "beautiful" problems. This means that beautiful problems have beautiful solutions. The link between concrete art and problem solving has been explored also for different field definitions, such as fields based on iterated function systems, and fields based on Mandelbrot transformations (Van Loocke, 1999b,e). The method described is open to other variations, each of which gives raise to its own aesthetics. For instance, phase fields may be defined in three dimensions, so-called teleological variations of the algorithm can be designed, and so on (Van Loocke 1999d).

References

Bechtel, W. Abrahamsen, A. (1994), Connectionism and the mind, Oxford: Blackwell

Gutowitz, H. (1991), Cellular automata: theory and experiment, Cambridge: Cambridge University Press

Personnaz, L., Guyon, I., Dreyfus, G., (1986), Collective computational properties of neural networks: new learning mechanisms, Physical Review A, 34, 4217- 4228

Rumelhart, D., Mc Clelland, J. (1986), Parallel Distributed Processing, 2 Volumes, MIT Press

Rumelhart, D. Smolensky, P., Hinton, G. (1986), Schemata and sequential thought, in: Rumelhart, D. Mc Clelland, J. (eds), Parallel Distributed Processing: explorations in the microstructure of cognition, Volume 2: Psychological and biological models, Cambridge: MIT Press, pp. 423-461

Van Loocke, Ph. (1994), The dynamics of concepts, Berlin: Springer

Van Loocke, Ph. (1999a), Properties of conscious systems and teleology: a cellular automaton perspective, Journal of Intelligent Systems, (accepted, to appear)

Van Loocke, Ph. (1999b), The art of growth and the solution of cognitive problems, in C. Soddu, proceedings of the 2-nd International Generative Art Congress, Milan, to appear

Van Loocke, Ph. (1999c), Function approximation in cellular automata. Fields and meta-patterns as methods for the representation of structural information, submitted

Van Loocke, Ph. (1999d), Problem solving in cellular automata and the problems of consciousness, in: Van Loocke, Ph. (ed.) The nature of consciousness, Amsterdam, Jon Benjamins, to appear

Van Loocke, Ph. (1999e), Fractals generated in cellular automata as solutions for benchmark problems in connectionism , Fractals (accepted)