Joint NCC & IIS Bull., Comp. Science, 4 (1996), 1–16

Stability of stored patterns in cellular-neural associative memory*

O.L. Bandman, S.G. Pudov

Associative neural memory of the Hopfield type, where each element (neuron) has a restricted amount of connections is considered as a cellular-automaton-like network. The goal of the investigation is to determine the degradation of memory capacity and efficiency of pattern retrieval caused by the decrease of connection number. As a result, conditions of individual stability and 1-attractivity of stored patterns are obtained being expressed in terms of neighborhood configurations of each neuron. Based on the above conditions a method of weight computation which insures stored patterns stability is introduced. Some results of cellular-neural associative memory simulation are presented.

1. Introduction

Among the scope of computation models exploring the idea of fine-grained parallelism, the most extravagant and promising nowadays is the Artificial Neural Network. It is an abstract network consisting of many bistable processing units (neurons), interacting each with all others by means of weighted connections. Data to be processed are represented by the neuron states, the result being predetermined by connection weights. The computation process is a free step-by-step transition from an initial state to the stable one. At any step each neuron calculates the threshold function of a sum-of-product of connection weights by neuron states of all neurons. Stable states determine the computation results and are characterized by the minimum of "energy function". Implicit and redundant form of storing information as connection weight values makes it possible to obtain the correct result when incomplete or distorted initial data are input. Herefrom the main application domains of the model are: pattern retrieval and classification, combinatorial optimization, image processing.

The most popular paradigm of artificial neural network is the Associative Neural Memory by Hopfield [1]. There is a great flow of investigations related both to the theory and to the technical implementation of Hopfield's memory. The theoretical investigations are designated to study the dynamics of computation process [2, 3], i.e., the stability and attractability of stored patterns. Based on these results, methods for associated neural memory

*Supported by the Russian Foundation for Basic Research under Grant 96–01–00033.

synthesis (determining connection weights) are developed [4], which provide much better efficiency than the variants proposed by Hopfield. Search for hardware implementation in the form of special-purpose computers, where the advantages of natural parallelism of neural networks is to be best manifested, lay mainly in the field of optical technology [1, 5], which is nowadays at the stage of scientific investigation. As for VLSI implementation, the main obstacle to it is the completeness of connection graph of the network, which leads to the enormous amount of links on the chip. This obstacle may be eliminated by restricting each neuron connections to certain local area in the network. Naturally, connection restriction affects the storing capacity and the restoring ability of associative neural memories. The question is how harmful is the effect of connection restriction and, as a result, what are the lower limits of the locality within which the efficiency degradation is acceptable. This question is not first stated here. There are some attempts to get some partial answers. So, in [6] the class of diluted associative neural memory is introduced, whose dynamic properties have been studied both theoretically and by simulating [7]. In [8] the previously proposed methods for associative neural memory synthesis is extended to meet the additional requirement of obtaining the network with given connection structure. Inspite of the importance of the above studies, they are far from closing the problem. Instead, they show its complexity and versatility.

In this paper we try to approach the problem from the position of cellular automaton theory [9]. Cellular automaton and associative neural memory have two fundamental features which show their similarity: 1) both are representatives of fine-grained parallelism, and 2) both are autonomous dynamic systems. These features we supplement with a few ones identical to those of cellular automaton (discrete time, synchronous mode of operation and local interactions), and some others, identical to those of neural networks (threshold cell function and weighted connections). Such a hybrid has been called *Cellular-Neural Associative Memory* (CNAM) [10]. To deal with it we use some concepts of cellular automaton generalization called Parallel Substitution Algorithm, as well as computer tools developed for cellular computations simulating [11].

The paper is organized as follows. Section 2 introduces notations, definitions, and formal representation of CNAM. In Sections 3 and 4 stability and 1-attractability conditions expressed in the terms of cell neighborhood properties, are obtained. Section 5 presents the results of CNAM learning and retrieval processes simulation.

2. Formal representation of cellular-neural associative memory

A CNAM to be investigated is defined by three notions: $\mathbf{N} = \langle \mathbf{C}, \mathbf{W}, \mathbf{\Phi} \rangle$. Here $\mathbf{C}$ is a set of rectangular arrays of equal size consisting of m rows and n columns of *cells*, represented as pairs $(c_{ij}, (i,j))$, where $c_{ij} \in \{1, -1\}$ are cell states, (i,j) are the coordinates in the array; $\mathbf{W} = \{W_{ij}\}$ is a set of *weight vectors* of the form $W_{ij} = (w_1, \ldots, w_q)$, w_k being a real number assigned to the connection between a cell c_{ij} and its k-th neighbor; $\mathbf{\Phi}$ is the rule according to which CNAM acts.

The set of coordinate pairs forms a discrete space $M = \{(i,j) : i = 0, \ldots, m;\ j = 0, \ldots, n\}$ referred to as a *naming space*. In order to avoid troubles caused by border effect the naming space is augmented by a number of margin rows and columns consisting of imaginary cells. The expanded naming space is $M' = \{(i,j) : i = -v, \ldots, -1, 0, \ldots, m, m+1, \ldots, m+v;\ j = -u, \ldots, -1, 0, \ldots, n, n+1, \ldots, n+u\}$. Cells in the added rows and columns have zero states. In the naming space M' mappings called *naming functions* $\phi(i,j) : M \Rightarrow M'$ are defined. The set of naming functions

$$T(i,j) = \{\phi_1(i,j), \ldots, \phi_q(i,j)\}, \tag{1}$$

where $\phi_k(i,j) \neq \phi_l(i,j)$ for any $(i,j) \in M$ and for all $k, l = 1, \ldots, q$ is called a *connection template*. The connection template $T(i,j)$ determines the set of cells (*the neighborhood*), the cell (i,j) communicates. The set of neighborhoods generated by a template for all $(i,j) \in M$ in a cellular array C^k is denoted as $Q(T)$.

The mapping $S : M \Rightarrow Q(T)$,

$$S(i,j) = \{(x_1, \phi_1(i,j)), \ldots, (x_q, \phi_q(i,j))\} \tag{2}$$

is called a *neighborhood function* (Figure 1). First entries in all brackets of (2) form a variable vector $X(i,j) = (x_1, \ldots, x_q)$. The value of the function $S(i,j)$, when applied to a certain C^k is obtained by substituting $c_{\phi_k(ij)}$ for x_k in (2). More often it is convenient to represent the value of $S(i,j)$ in the form of a state neighborhood vector $Q(i,j) = (c_1, \ldots, c_q)$. whose components are neighborhood states, arranged in the same order that in (2).

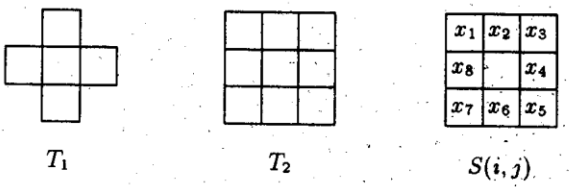

Figure 1. Spatial representation of templates T_1 and T_2 and a neighborhood function $S(i,j)$, generated by T_2

The rule of CNAM operating Φ is described by a following iterative procedure.

Procedure (*Retrieval procedure*). Let $C(t)$ be the array after the t-th iteration. Then

1) all cells in $C(t)$ compute the following function:

$$f(X(i,j),W_{ij} = \begin{cases} 1, & \text{if } X(i,j) * W_{ij} > 0, \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

where "$*$" denotes the scalar product of vectors, and the result is the cell state c_{ij} at $(t+1)$;

2) if $C(t+1) = C(t)$, then $C(t) = \Phi(C(0))$ is the result of the computation, which corresponds to a *stable state* of CNAM.

Operating in such a way CNAM performs storage and retrieval of a set of patterns $\{P^1, \ldots, P^l\}$, given as cellular arrays, and called *prototypes*. A pattern to be retrieved is input by setting the CNAM in the corresponding initial state $C(0)$. Application of the above iterative procedure to any $C(0)$

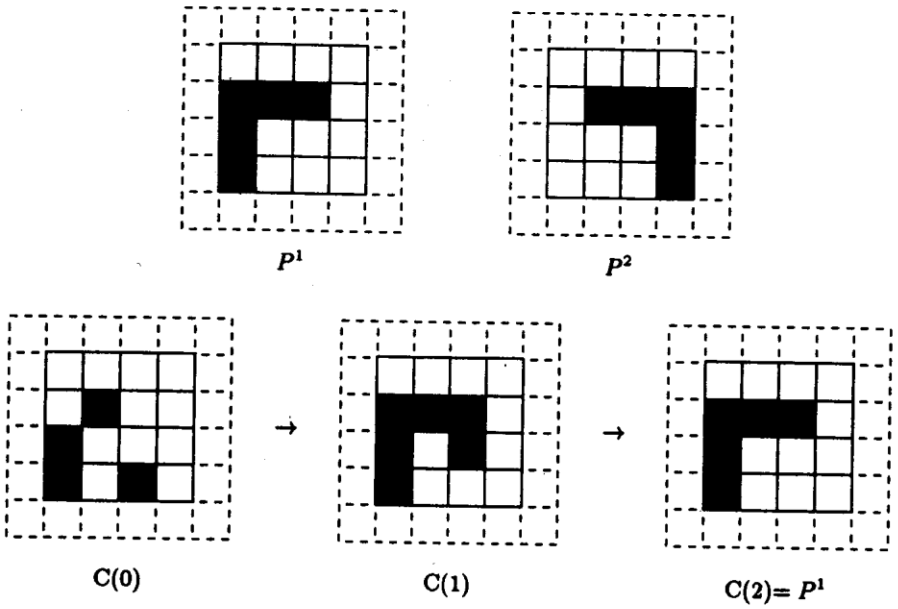

Figure 2. P_1, P_2 – stored patterns, $C(0), C(1), C(2)$ – initial, intermediate and terminal patterns in the process of retrieval of P_1, W – weight vector set with a neighborhood function generated by T_1 of Figure 1. The black color stands for 1, the white – for -1. Imaginary cells of M'/M are drawn in dashed lines and have zero states

Table 1. Weight vectors of CNAM storing P_1 and P_2 from Figure 2

i \ j	0	1	2	3
0	(0, 2, 0, 0)	(0, 2, −2, 0)	(0, 2, −2, 2)	(0, 0, 0, 2)
1	(0, 0, 2, 0)	(−2, 2, −2, 0)	(−2, 0, −2, 0)	(2, 0, 2, 0)
2	(2, 0, 2, 0)	(−2, 2, 2, 0)	(−2, 0, 2, 2)	(2, 0, 2, 0)
3	(2, 0, 0, 0)	(2, 2, 0, 0)	(1, 2, 0, 2)	(2, 0, 0, 0)

should terminate in a stable state equal to a prototype which has the closest resemblance to $C(0)$. In Figure 2 the process of retrieval of one out of two stored prototypes is shown, a distorted pattern being input. Weight vectors of the CNAM are given in Table 1.

The correspondence of each prototype to a stable state of the CNAM should be provided by the proper determination of the weight vector values, which constitutes the *learning process* of CNAM. The fundamental concept of most learning methods are based on, is the stability condition, which being reduced to the local case, is expressed by the minimum of the following function

$$E = -\sum_{M} c_{ij}(W_{ij} * Q(i,j)), \tag{4}$$

referred to as *Liapunov function.*

The strategy for constructing learning procedure should be grounded on the quest to provide the following dynamic properties:

(i) *individual stability*, i.e., each prototype should be a stable state;

(ii) *k-attraction* of stable states, i.e., given a number k, each input state differing from a prototype P^i not more than by k entries, must be retrieved as P^i;

(iii) *minimum of spurious patterns*, i.e., there should be as few as possible stable states differing from the prototypes.

No method in the artificial neural network theory is yet known which matches the three above requirements completely. Much less it is so for cellular neural networks. The best results [4], modified later by the same authors [8] to diluted network case, require extremely complex matrix manipulations, which are not consistent by nature with the discrete character of cellular parallelism and therefore can not be performed in the same cellular processor, if any would exist. Hebb's method [1] used by Hopfield, does not guarantee even the first requirement and provides the capability to store not more than $0.15|M|$ prototypes in the networks with full connections and, consequently $0.15|Q|$ in CNAM. So, the method for perceptron learning by Rosenblatt [12] is chosen here as a basis for studying the limits of CNAM capabilities.

3. Stability conditions

According to requirement (i) and the assumed cell function (3) the condition of individual stability is defined as follows.

Definition 1. A prototype P^k is called *strong stable* if the first iteration of Procedure result in itself, i.e.,

$$\Phi_1(P^k) = P^k \quad \text{for all} \quad k = 1, \ldots, l. \tag{5}$$

Following the Rosenblatt perceptron learning method the prototypes are regarded as vectors in $|M|$-dimensional linear space, and the concept of linear separability, similar to that from [12] but reduced to the local connection case is introduced.

Definition 2. Prototypes $P^1, \ldots, P^l$ are said to be *linear separable* (*separable* for short) at a cell (i,j) in a CNAM, if the following holds for all $P^k, k = 1, \ldots, l$,

$$c_{ij}^k (Q^k(i,j) * W_{ij}) > 0, \tag{6}$$

where the top index denotes the prototype number.

Theorem 1. *Let* $\{P^1, \ldots, P^l\}$ *be a set of prototypes,* $P^k \in \mathbf{C}(M)$, $k = 1, \ldots, l$. *Then all prototypes are strong stable, if they are separable at any cell of* **N**.

Proof. If the prototypes are separable, then by Definition 2 (5) holds for any cell in each P^k, which means that the sign of c_{ij}^k is equal to that of the scalar product $Q^k(i,j) * W_{ij}$. So, according to (3), application of retrieval procedure to any P^k causes no changes of its states. Thus, the stability condition (5) is fulfilled. Conversely, if all prototypes are strong stable, i.e., the retrieval procedure performs no changes in P^k, then the condition (6) of Definition 2 holds, which proves the linear separability. □

Corollary 1. *If* P^k *is a strong stable state in* **N**, *then so is* $\bar{P}^k$, *where* $\bar{P}$ *is the array obtained by inverting cell states in* P.

Theorem 1 allows to construct a learning algorithm which guarantees strong stability of the set of prototypes to be stored.

Algorithm 1 (*Modified perceptron learning* [12]). The prototypes $P^1, \ldots, P^l$, which are to be stored in CNAM are given. A set of weight vectors **W** should be determined, such that for any P^k, $k = 1, \ldots, l$, condition (5) is met.

Step 1. The sequence $P^1, \ldots, P^l, P^1, \ldots, P^l, P^1, \ldots$, which is the cyclic repetition of a sequence of prototypes to be stored, is formed and then relabeled resulting in an infinite sequence $\sigma = P_0, P_1, \ldots, P_t, \ldots$.

Step 2. Initial values of all components of $W_{ij}(0)$ are chosen arbitrarily for all $(i,j) \in M$.

Step 3. Weight vectors are updated according to the following iterative procedure. At each t-th iteration for any $(i,j) \in M$:

1) the value
$$h_t(i,j) = Q_t(i,j) * W_{ij}(t), \tag{7}$$
is computed, where $Q_t(i,j)$ is the state neighborhood vector of the cell $c_{ij} \in P_t$;

2) weight vectors $W_{ij}(t)$ are changed as follows:
$$W_{ij}(t+1) = \begin{cases} W_{ij}(t), & \text{if } c_{ij}h(t) > 0, \\ W_{ij}(t) + c_{ij}Q_t(i,j), & \text{otherwise}, \end{cases} \tag{8}$$
where c_{ij} and $Q_t(i,j)$ are the state and the state neighborhood of P_t;

3) the algorithm stops, when no change of weight vectors has been performed during l iterations;

4) if the algorithm enters in a process when at some cells an oscillatory weight change is observed, then no vector matrix exists which guarantees strong stability of the given prototypes.

In [12] equivalency of learning algorithm convergence to separability of prototypes at all cells is proved. Naturally, the same is true for the case of cell· local connections, because each cell within its neighborhood is in the same condition that a neuron in a perceptron network.

Algorithm 1 has some useful properties which make it efficient when implemented as a special-purpose array processor. They are as follows.

1. It may be transformed in a cellular highly parallel version, where all cells compute their next states simultaneously.
2. It may be performed by one and the same CNAM, which is to be learned, only minimal modification being needed to make cells capable both to compute the weights and to retrieve the patterns.
3. During the learning process the cells at which (6) does not hold may be marked in order to change the corresponding prototype.

The above properties show that Algorithm 1 provides the tool for testing and diagnosing a given set of prototypes for separability, and, hence, for the existence of CNAM, which provides their individual stability. However, it would be extremely useful to test prototype set before or during the learning

process, which requires separability conditions to be expressed basing on the prototypes properties. To determine some of them the following new concept is needed.

Definition 3. Let P^g and P^h be two prototypes,

$$S^g(i,j) = (c_1^g, \ldots, c_q^g) \quad \text{and} \quad S^h(i,j) = (c_1^h, \ldots, c_q^h),$$

be the neighborhoods of cells (c_{ij}^g and c_{ij}^h in the prototypes P^g and P^h, respectively. Then the set of neighbor numbers

$$Id^{gh}(i,j) = \{n : c_n^g c_n^h = c_{ij}^g c_{ij}^h\} \tag{9}$$

is called an *identification set*, and the neighborhood cells with the names from $Id^{gh}(i,j)$ are referred to as *identification cells* (Figure 3).

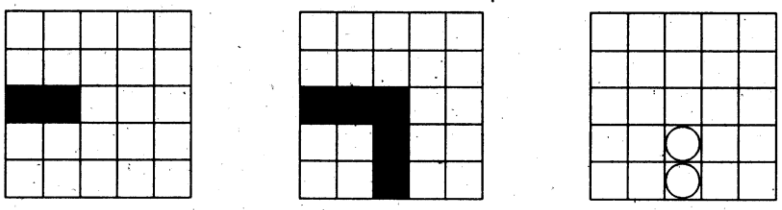

Figure 3. A pair of neighborhoods of identically named cells of two prototypes and two identification cells (marked with circles)

The concept of identification cells allows to formulate separability condition in the form of the following theorem.

Theorem 2. *Two prototypes P^g and P^h both from* **C** *are linear separable, if at any cell named $(i,j) \in M$ the identification set cardinality is not less than 1, i.e.,*

$$|Id^{gh}(i,j)| \geq 1 \quad \textit{for all} \quad (i,j) \in M. \tag{10}$$

Proof. If P^g and P^h are separable, then for both of them condition (6) holds at each cell. Let us partition the neighborhoods

$$S^g(i,j) = \{(c_1^g, n_1), \ldots, (c_q^g, n_q)\} \quad \text{and} \quad S^h(i,j) = \{(c_1^h, n_1), \ldots, (c_q^h, n_q)\}$$

into two parts: the first part, containing cells with equal states, i.e., such that $c_k^g c_k^h = 1$, and the second one, containing those cells, for which $c_k^g c_k^h = -1$. The neighbor numbers of the first part form a set $Q'(i,j)$, and those of the second one – a set Q''. The left-hand sides of (6) may be then rewritten as follows:

$$c_{ij}^g\Big(\sum_{k\in Q'} c_k^g w_k + \sum_{k\in Q''} c_g^g w_k =\Big) > 0, \quad c_{ij}^h\Big(\sum_{k\in Q'} c_k^h w_k - \sum_{k\in Q''} c_k^h w_k\Big) > 0, \quad (11)$$

minus in the second inequality resulting from the difference in state signs in the neighbors of Q''. The above inequalities show that, if $c_{ij}^g = c_{ij}^h$, then $Q' > Q''$ and, if $a_{ij}^k = -a_{ij}^l$, then $Q' < Q''$, which may be possible, if only there is $Q' > 1$, which corresponds condition (6) and proves the necessity. Conversely, the condition of the theorem (10) determines the above inequalities, which in its turn, is equivalent to the condition of separability, (6), which proves the sufficiency. □

Corollary 2. *Let P^1, P^2, P^3 from $\mathbf{C}$, be the prototypes in which $c_{ij}^2 = c_{ij}^3$ and $Q^1 = Q^2 = \bar{Q}^3$. Then if P^1 and P^2 are separable at a cell c_{ij}, then P^1 and P^3 are not.*

Theorem 2 gives the possibility to test any pair of prototypes for separability. However, pairwise separability does not guarantee strong stability for the whole set containing more than three prototypes.

Theorem 3. *For any prototype P^0 there exist three others P^1, P^2, P^3, such that all four are pairwise separable, but for the whole set the separability condition* (6) *does not hold.*

Proof. Let us take a cell named (i,j) in P^0, such that $c_{ij} = 1$, and represent $Q^0(i,j) = (c_1, \ldots, c_q)$ as a concatenation of three vectors $Q^0(i,j) = (Q_1 \cdot Q_2 \cdot Q_3)$, where

$$Q_1 = (c_1, \ldots, c_b), \quad Q_2 = (c_{b+1}, \ldots, c_d), \quad Q_3 = (c_{d+1}, \ldots, c_q).$$

Now let us construct three prototypes P^1, P^2, and P^3 with the following neighborhoods of the cell named (i,j), assuming that its state is $c_{ij} = 1$ in all arrays.

$$\begin{aligned} Q^1(i,j) &= (Q_1 \cdot Q_2 \cdot \bar{Q}_3), \\ Q^2(i,j) &= (Q_1 \cdot \bar{Q}_2 \cdot Q_3), \\ Q^3(i,j) &= (\bar{Q}_1 \cdot Q_2 \cdot Q_3), \end{aligned}$$

where $\bar{Q}$ is componentwise inverse of Q. It is clear, that for the above three state neighbor vectors a weight vector W_{ij} may be found, for which (6) holds. It follows from the fact that (6) holds for any pair of subvectors Q_k and W_k, $(k = 1,2,3)$, the latter being obtained by partitioning W_{ij} in the same manner that Q^0. The separability condition (6) also holds for $Q^0(i,j)$, because $Q^0(i,j) = Q^1(i,j) + Q^2(i,j) + Q^3(i,j)$ is the convex sum. So, the set of prototypes $\{P^0, P^1, P^2, P^3\}$ is separable at the cell named

(i,j), and consequently any pair of them is separable too. According to Corollary 1 the same is true for their inverses $\bar{P}^k, k = 0,1,2,3$. Now let us consider the set of prototypes $\bar{P}^1$, $\bar{P}^2$, $\bar{P}^3$, P^0. According to the supposition that in the initially given prototypes $c_{ij} = 1$, in their inverses the states $c^1_{ij} = c^2_{ij} = c^3_{ij} = -1$, whereas $c^0_{ij} = 1$. According to Corollary 1 the new set is separable at c_{ij} as well. However, since $Q^0(i,j) = -\sum_{k=1}^{3} \bar{Q}^k(i,j)$, then according to Corollary 2, P^0 is not separable at the cell named (i,j) to any of $\bar{P}^k$, $k = 1,2,3$. In consequence of this contradiction the set of prototypes $\{P^0, \bar{P}^1, \bar{P}^2, \bar{P}^3\}$ being pairwise separable is not separable as a whole. Since all above is true with the suggestion that $c_{ij} = -1$, the theorem is proved. □

4. Attractivity conditions

In order to provide capability to retrieve a prototype when a distorted pattern has been input, learning procedure should rely upon the criterion of maximalizing the degree of distortion which is admissible for the recognition of a prototype. To formalize this criterion the Hamming distance $H(C^1, C^2)$ (the number of cells with different states), is used, and the following concepts are introduced.

Definition 4. A set of cellular arrays $C^1, \ldots, C^r$ such that the application of Procedure 1 to any one of them results in a stable state P^k, is called a *domain of attraction* of P^k.

Definition 5. A cellular array C is said to be a *k-attractor*, if the application of Procedure 1 to any initial array C^j, such that $H(C, C^j) \leq k$ results in C. If the result is achieved after one iteration, then N is a *strong k-attractor.*

Remark. From Definition 4 it follows that if a cellular array is a k-attractor, then it is a $(k-1)$-attractor as well, and any stable state is a strong 0-attractor.

In the ideal case each prototype $P^h \in \mathbf{C}$ to be stored should be a k_h-attractor, so that any $C^k \in \mathbf{C}$ belongs to a domain of attraction of one of the prototypes. In that case all three requirements to the learning process (see Section 2) would be fulfilled. Seeking the ways to approach this ideal case, a method for providing the given prototypes to be strong 1-attractors, was proposed in [13]. The idea of this method is in augmenting the set of patterns to be stored, by adding to each prototype P^h as much as $|M|$ patterns P^h_{ij}, $(i,j) \in M$, differing from P^h by a state c_{ij}, called (i,j)*-distortions.* The learning process should make these distortions to be retrieved as the true prototypes, so the added patterns should meet the following condition:

$$\Phi(P_{ij}^h) = P^h, \tag{12}$$

for all $(i,j) \in M$ and for all $h = 1, \ldots, l$.

To follow the above idea when constructing the similar algorithm for the local connection case and, moreover, for taking advantage of the parallelism induced by the connection locality, some new notions are to be introduced.

Let us partition the cellular array N into $u = m/\alpha \cdot n/\beta$ (α and β being the dimensions of the template) rectangular subarrays referred to as *macrocells*. Each macrocell contains a central cell and its neighborhood, whose cells being numbered according to their order in the neighborhood function $n_0, n_1, \ldots, n_q$, where n_0 is the number of the central cell. Due to the local character of interactions the prototype distortions in different macrocells influence the learning process independently. Hence, the iterative updating of weight vectors by applying (8) sequentially to a prototype, say P^h, distorted at cells belonging to different macrocells, has the same result that if a single iteration were applied to the array $P_h(n_i)$, obtained by inverting the n_i-th neighbor states in all makrocells of P^h, $P_h(n_i)$ being referred to as n_i-th 1-distortion of P_h.

Algorithm 2. The cellular arrays $P^1, \ldots, P^l$ which are to be stored in CNAM are given. A weight matrix $\mathbf{W}$ should be determined, such that for any P^h, $h = 1, \ldots, l$ and $P^h(i,j)$, $(i,j) \in M$, conditions (6) and (12) are met, respectively.

Step 1. The sequence $P^1, \ldots, P^l, P^1(n_1), \ldots, P_1(n_q), P^l(n_1), \ldots, P^l(n_q)$, $P^1, \ldots, P^l, P_1(n_1), \ldots$, which is the cyclic repetition of a sequence of given prototypes and their n_i-th 1-distortions is formed, and then relabeled resulting in an infinite sequence $\sigma = P_0^1, P_1^2, \ldots, P_t^g, \ldots$, a top index indicating the number of corresponding prototype in the initially given set, a bottom one – its number in the sequence.

Step 2. Initial values of all components of $W_{ij}(0)$ are chosen arbitrarily.

Step 3. Weight vectors are updated according to the following iterative procedure. At each t-th iteration for any $(i,j) \in M$:

1) the value

$$h_t(i,j) = Q_t(i,j) * W_{ij}(t), \tag{13}$$

is computed, where $Q_t(i,j)$ is the state neighborhood vector of the cell named (i,j) in P_t^g;

2) weight vectors $W_{i,j}(t)$ are changed as follows:

$$W_{ij}(t) = \begin{cases} W_{ij}(t), & \text{if } c_{ij}^g h_t > 0, \\ W_{ij}(t) + c_{ij}^g * Q^g(i,j), & \text{otherwise,} \end{cases} \tag{14}$$

where c_{ij}^g and $Q^g(i,j)$ are the state and the neighborhood of the cell, named (i,j) in P^g, respectively;

3) the algorithm stops, when no change of weight vectors has been performed during lu iterations;

4) if the algorithm enters a process when at some cells an oscillatory weight change is observed, then no vector matrix exists which guarantees 1-attraction of the given prototype set.

Like in the case of strong stability, it is preferable to have the conditions of strong k-attraction expressed in terms of prototype properties. For that the concept of identification cells (Definition 3) is also used.

Theorem 4. *If all prototypes* $\{P^1, \ldots, P^l\}$, $P_g \in \mathbf{C}$, *are strong 1-attractors, then any pair of them* (P^g, P^h) *has the identification set cardinality not more than 3 at any cell* $(i,j) \in M$, *i.e.,*

$$Id^{gh}(i,j) \geq 3 \quad \textit{for all} \quad (i,j) \in M. \tag{15}$$

Proof. If the prototypes are strong 1-attractors, then according to Definition 5, they are separable, and, hence, pairwise separable too. Thus, by Theorem 2, for any pair P^g, P^h, $|Id^{gh}(i,j)| \geq 1$ at all $(i,j) \in M$. Since 1-distortions of P^g or P^h may result in inverting the state of the identification cell, then there should be two more identification cells, which provide the fulfillment of separability condition even if both distortions coincide with the identification cells. So, 3 identification cells should exist to assure the separability of any 1-distortion of P^g to any one of P^h. Since all above is true for any $(i,j) \in M$ and any pair given prototypes, the theorem is proved. □

Corollary 3. *If all prototypes* $\{P^1, \ldots, P^l\}$, $P_g \in \mathbf{C}$, *are strong k-attractors, then any pair of them* (P^g, P^h) *has the identification set cardinality not more than* $(2k+1)$ *at any cell named* $(i,j) \in M$.

Remark. The condition of Theorem 4 is invariant of the size and the structure of the neighborhood function.

Extrapolating the result of simulation of the method [13] to the cellular case, it is possible to conclude that strong 1-attractability provides the

Figure 4. A pair of patterns with the $|Id| \geq 3$ for a neighborhood function $S(i,j)$ of Figure 1

retrieval of hardly distorted prototypes. This advantage is paid by a large complexity of the learning process, which, moreover, frequently does not converge. For example, if the neighborhood is limited by the size 3×3, then it is very difficult to construct a pair of patterns, having not less than 3 identification cells at each cell (Figure 4). However, if the neighborhood contains more than 20 cells, it is quite possible to store a few strong 1-attractor, which provides high retrieval capability.

5. Some results in learning and simulating

Some computer experiments have been performed in order to obtain quantitative assessment of learning and retrieval capability of CNAM. A computer simulating system, called ALT (Animating Language Tools) [11], which is oriented to cellular computations investigation was used. This system combines textual and graphical tools for representing cellular algorithms and displaying computation process. CNAM is presented in the system as in the form of a multiplanar array, so, that the naming set is $M''' = \{(k,i,j) : k = 0,1,\ldots,q; (i,j) \in M'\}$, k being the number of the plane. The 0-th plane of the array (*pattern plane*) is equal to the current cellular array, state values being represented by cell colors (1 by black, -1 by white). Each k-th plane, $k = 1,\ldots,q$, *weight plane* contains the k-th components of the weight vector W_{ij}, so that the subset of cells named $\{(w_1,(1,i,j)),\ldots,(w_q,(q,i,j))\}$ forms a register storing the weight vector $W(i,j)$ (Figure 5).

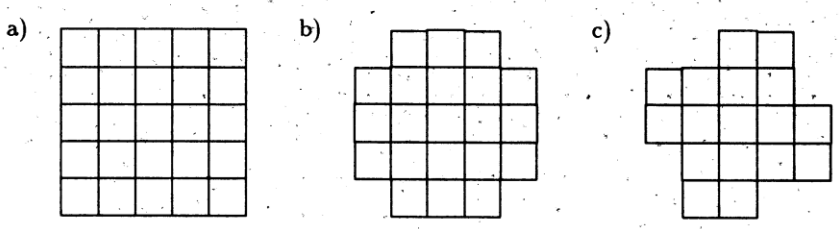

Figure 5. Examples of the templates of CNAMs, which were investigated by simulating; a) $q = 24$, b) $q = 20$, c) $q = 16$

The algorithm (either a learning one or a retrieval procedure) is written in ALT-language (a modification of C) in a program window. The program is executed in a quasiparallel form, the results of each iteration being displayed in the cellular array.

A CNAM of the size 20×20 cells has been investigated; two templates shown in Figure 1 determining the neighborhoods. The prototype sets contained imaged of symbols from English and Russian alphabets and arabian numbers.

Learning process has been performed according to Algorithm 1 as follows.

A multiplanar cellular array representing the CNAM has been drawn with the help of graphical tools of ALT, as well as the planar arrays representing the prototypes and the neighborhood functions. The learning ALT-program has been written in a special window in ALT-language. The main part of the program constitutes the block which contains two following instructions: (a) inputting next prototype states into the pattern plane of the CNAM array, (b) updating states of cells in weight planes according to (8). Both instruction are parallel operations, i.e., they are executed by all cells indicated in the instruction. Some additional instructions are introduced into the program, which test the linear separability of the prototypes at each cell, marking those of them, which do not meet the condition (6). Such a test makes it possible to do some corrections of "bad" prototypes or, perhaps, remove them out of the stored set.

The learning program, as well as the above modification, have been applied to a number of prototypes in order to assess storing capability and learning time. The latter is measured in iteration number needed for learning algorithm convergence. Some representative results are shown in Table 2.

Remark. It is worth to notice, that the CNAM with $q = 24$ was the first to be learned. During this learning process 7 corrections of the prototypes have been done, which, none the less, have not altered the symbol semantic. This fact explains the decrease of learning time for the case $q = 20$ relative to the case $q = 24$. However, the separability for the whole set containing 50 prototypes was not obtained, one of them had to be removed, which is marked by "*".

Table 2. Learning time of CNAMs with different neighborhood cardinality

Number of stored prototypes	15	26	40	50
q=24	12	14	19	25
q=20	12	17	25	38*
q=16	17	25	32	—

In the retrieval process the states of weight planes of a multiplanar array are weight values, obtained after the learning process. The retrieval program is written in the ALT language according to Procedure 1. It contains two parallel instructions: 1) the input of a pattern to be retrieved into the pattern plane, and 2) the iterative operation of computing the cell function (3). When no change of the states is observed during a prescribed number of iterations, the algorithm stops. The retrieval process in CNAM, storing 50 prototypes with $q = 24$ was simulated. Since learning process provided strong stability, but not attractability, the retrieval of distorted patterns are not guaranteed. So, the aim of simulation was to determine how many distorted patterns may be, nevertheless, recognized. The parallelism induced

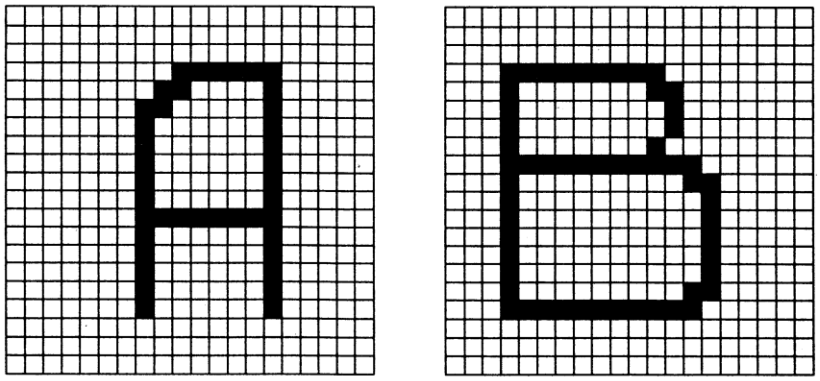

Figure 6. Examples of prototypes stored in CNAMs under investigation

by connection locality was explored, so the set of 1-distorted prototypes (one per a macrocell) was input in the form of $P(n_i)$. 5 prototypes representing the symbols of English and Russian alphabets in the form shown in Figure 6 have been stored. The results are given in Table 3.

Table 3. Number of 1-distorted prototypes $P^k(i,j)$, which have not been recognized in CNAM with 400 cells and 24 cell template

Prototype	A	B	C	D	F	I
$\lvert P^k(i,j)\rvert$	145	148	125	116	145	128

Table 3 shows, that a CNAM being learned to store as many prototypes as it can guarantee to be strong stable, is capable to recognize 30–40% of the set of 1-distortions. For obtaining a better quality of retrieval Algorithm 2 should be used.

6. Conclusion

In this paper, the following results on CMAM investigation are presented.

1. Algorithms for CNAM learning and retrieval of distorted patterns are proposed. All of them take most advantage of the fine-grained parallelism induced both by cell operational independence and by connection locality. A very important property is also the fact that learning and retrieval processes may be performed in the same cellular array. This allows to consider such an array as the CNAM architecture for VLSI implementation.
2. Some necessary and sufficient conditions of strong stability and k-attractability are obtained, which are expressed in terms of cell neighborhood relations of a pair of prototype. These conditions allow to assess

the compatibility of a pair of prototypes for being stored together in a CNAM.

3. Simulation of learning and retrieval processes in a CMAM storing the symbols drawn in thin lines (see Figure 6) showed the following: 1) for this class of patterns it is possible to provide strong stability approximately for $2|Q|$ prototypes, and 2) when learned according to Algorithm 1, which provides strong stability, only 60–70%, of 1-distortions can be restored.

References

[1] J.J. Hopfield, D.W. Tank, *Computing with Neural Circuits: a Model*, Science, **233**, 1986.

[2] M. Cottrell, *Stability and attractability in associative memory networks*, Biological Cybernetics, **58**, 1988, 129–139.

[3] L. Perzonas, I. Guyon, G. Dreyfus, *Collective computational properties of neural networks: new learning mechanism*, Physical Review, A, **34**, November, 1986, 4217–4228.

[4] A.M. Michel, J.A. Farell, H. Sun, *Analysis amd Synthesis Techniques for Hopfield Type Synchronous Discrete Time Neural Networks with Application to Associative Memory*, IEEE Transactions, **CS-37**, No. 11, 1990.

[5] J. Zhang, I. Zhang, D. Yan, A. He, L. Liu, *Local interconnection neural network and its optical implementation*, Optics Communication, **102**, 1993.

[6] B. Derrida, E. Gardner, A. Zippelius, *A exactly solvable asymmetric Network Model*, Europhysics Letters, **4**, No. 2, 1987, 167–173.

[7] J.J. Arenzon, N. Lemke, *Simulating highly diluted neural networks*, Journal of Physics, A, Math. Gen., **27**, 1994, 5161–5165.

[8] D. Liu, A. Michel, *Sparsely interconnected artificial neuron networks for associative memories*, Lecture Notes in Comp. Sci., **606**, p. 155.

[9] J. von-Neumann, *Theory of Self-Reproducing Automata*, ed. A.W.Burks, University of Illinois Press, Urbana and London, 1966.

[10] O.L. Bandman, *Cellular-neural computations, formal model and possible applications*, Lecture Notes in Computer Science, **964**, 1995, 21–35.

[11] S. Achasova, O. Bandman, V. Markova, S. Piskunov, *Parallel Substitution Algorithm. Theory and Application*, World Scientific, Singapore, 1995.

[12] F. Rosenblatt, *Principles of Neurodynamics*, Washington, Spartan, 1959.

[13] X. Zhuang, Y. Huang, F.A. Yu. *Design of Hopfield content-addressable memories*, IEEE Transactions on Signal Processing, **42**, No. 2, 1834–1837.