Bull. Nov. Comp. Center, Comp. Science, 3 (1995), 1–17

Cellular–neural computations: formal model*

O.L. Bandman

A formal model for the fine-grained parallel computations is presented, which combines the connectionist approach of Artificial Neuron Networks with the cellular -like communication structure. The model is based on the concepts and formalities of Parallel Substitution Algorithm, which is considered the most theoretically advanced generalization of Cellular Automaton. Principles of cellular-neural algorithms synthesis are discussed, and three simple examples (optimization on graphs, pattern retrieval, image processing) are given.

1. Introduction

There are two exotic approaches to the fine-grained parallel computations, which take their theoretical origins from the early years of computer science. The first ascends to von-Neumann's Cellular Automaton [1]. The second – to Mc-Calloch' and Pitts' Logical Calculus [2]. Both are progressing rapidly, each in its own way of formal methods sophistication as well as in searching for proper hardware implementation.

The first approach has gained greatest advancement in the theory of Parallel Substitution Algorithm (PSA) [3], which is directed to the investigation and design of cellular algorithmically oriented architecture. The following fundamental concepts form the background of the PSA.

1. The PSA processes cellular arrays which are sets of cells, characterized by a state and a name. The set of states form the PSA alphabet. The set of names represents a discrete space, in which interaction patterns are defined by means of naming functions.

2. Operations over a cellular array are specified by a set of substitutions. All substitutions are applied in parallel (at once) at every cell of the array. The substitution is applicable to an array at a certain cell if the left-hand side containing this cell is included in it. The execution of an applicable substitution is in replacing the subarray of its left-hand side to that of its right-hand side.

3. The computation is an iterative procedure. At each step all applicable substitutions are executed, resulting in a new cellular array. The computa-

*Supported by the Russian Foundation for Basic Research under Grant 93–01–01000.

tion terminates when the array is obtained to which no single substitution is applicable.

4. Each PSA has an explicit interpretation by an automata net.

5. Due to the context in the left-hand side of substitutions, similar constructs are used to represent both processing and control. This fact underlines PSA composition and decomposition techniques as well as the methods for control synthesis.

The second approach is under intensive elaboration for creating the Artificial Neural Networks (ANN), the interest being concentrated on two paradigms: Hopfield's Associative Memory [4] and Boltzmann's Machine [5]. The ANN is a model of fine-grained parallel computation belonging to connections type. The main difference of this model from other massively parallel computations is in that the results are not computed under the program control, but are obtained when the network is settled in a stable state. The implicit and redundant form of storing data as connection weight values provides the ability of restoring the lost information. So, the main field of ANN application is as follows: pattern retrieval, pattern classification, searching for optimal solution.

The following features characterize the ANN computational model.

1. The ANN processes a data array which represents the states of bistable elements, called neurons. Each neuron is connected to all others, the connections being characterized by real non-negative numbers, referred to as connection weights.

2. The algorithm for ANN to solve a given problem is specified by the nonlinear function and the values of connection weights – the weight matrix. Constructing the weight matrix and choosing the non-linear function constitute the process of ANN programming or learning, which is the most complicated stage of problem solving by the ANN, requiring much skill and time.

3. Two modes of computation are admitted in the ANN theory. Originally, a continuous variant of computation was proposed, and its implementation as an asynchronous transition from a given initial state to a stable one, characterized by the minimum (possibly, local) of the energy function was studied [4]. Later on, the discrete time synchronous mode of ANN operation has become more popular. In this case the iterative procedure starting from the initial state at each step brings the network closer and closer to a stable state, where the computation stops, because no single neuron changes its state.

4. There are two ways of implementing the ANN. The first is widely used nowadays. It consists in simulation the ANN algorithm on a conventional computer, sometimes augmented by special–purpose processor for fast computing sum-of-products. The second way is associated with the direct implementation of neural network architecture. Much investigations hav-

ing been done to find the technologies (optics, opto-electronics, holography) which provide the effective implementation of ANN, the most difficult task being global connections realization.

5. The ANN model of computation has no special means for composing, decomposing, and making equivalent transformations of the algorithm.

In spite of the fact, that the two above models have some opposite features, we here make an attempt to cross them, cherishing the hope that the resulting hybrid would possess some merits of both. Particularly, we want it to inherit the interaction boundedness from PSA and the connectionist mode of computation from ANN. However, since parents disadvantages should also be inherited, the problem arises to study them and determine the limits of the new model capabilities. Speaking more specifically, the aim of the paper is to construct a formal representation and a set of tools for studying the computation process in cellular arrays, whose cells perform neural functions. The table bellow shows, what properties cellular-neural computations inherit from those of its parents.

	PSA	ANN	Cellular-neural computation
Process	discrete	continuous	discrete
Mode	synchronous	asynchronous	synchronous
Connections	bounded unweighted	each with all weighted	bounded weighted
Cell function	finite automaton	sigmoid	sigmoid

The quest for connection restriction in neural networks has stimulated a number of investigations of Hopfield's type ANN with local connections. There are three lines of inquiries in these studies. The first is associated with the ANN implementation in a form of asynchronous cellular system with amplifiers [6]. The main theoretical result is stability conditions expressed in terms of circuit parameters. The applications are confined to pattern classification problems. The second line [7] is concerned to the investigation of the capability of Hopfield's ANN, in which cell connections outside the given neighborhood are cut off, the loss of connections being paid by the decrease of memory capacity. The third direction of investigations seems to be the most fruitful [8]. It introduces the method of construction a sparse weight matrix with the predetermined locations of zeroes, which provides the correspondence of stable states to the given set of stored patterns.

Here the cellular-neural algorithm is regarded as a PSA with sigmoid (neuron-like) cell functions. There are two reasons for that. Firstly, we hope that the wide expressive ability of PSA (naming functions, context, operation on subsets of cells, equivalent transformations, etc.) would compensate the constraints imposed on connection structure. Secondly, PSA is

a model born and bred in our staff, and we have in our possession all the facilities (methods, techniques, computer simulating tools) to deal with.

The paper consists of five sections. The first section is the introduction. The second section contains formal concepts and definitions. In the third section the general principles of cellular-neural algorithm synthesis are stated and general principles of its solution are discussed. The fourth section deals with examples, illustrating the expressive capability of the proposed formalism.

2. Formal representation of a cellular-neural algorithm

When constructing the formal representation of cellular-neural algorithm the priority is given to "cellular" notation, which is as close as possible to that of PSA. Sometimes, if necessary, the ANN concepts are also used.

Definition 1. A triple (a, m, v), where $a \in A$, A – a finite *alphabet*, $m \in M$, M – *a naming set*, and $v \in V$, V –*a weight vector set*, is called *a neuron.* A pair (a, m) is referred to as *a cell*, a being interpreted as *a state*, m – as *a name.*

Definition 2. A set of neural cells $N = \{(a_i, m_i, v_i) : i = 1, \ldots, q\}$ with no pair of elements in N having equal names, is called *a cellular-neural array.* The set of cells, obtained by omitting the vectors v_i from the triples in N

$$C(N) = \{(a_i, m_i) : \ (a_i, m_i, v_i) \in N\} \tag{1}$$

is called *a cellular array*, generated by N and denoted as $C(N)$.

The alphabet A consists of three parts: $A = A_0 \cup A_x \cup A_c$, where $A_0 \subseteq$ R is a state alphabet, $A_x = \{x_1, x_2, \ldots, x_n\}$ is an alphabet of variables with the domain in A_0, $A_c = \{\alpha, \beta, \ldots\}$ is a control alphabet. The naming set M may consist of any kind of symbols or tuples of symbols. The set V of weight vectors has the same cardinality as that of the naming set, the length of vectors being not more than this value, the component being from R.

The set of all finite cellular-neural arrays, generated by an alphabet A, a naming set M, and a set of weight vectors V isomorphic to M, is denoted as $K(A, M, V)$. The set of cellular arrays, generated by A and M is referred to as $K(A, M)$.

Definition 3. A function $\phi(m)$ determined on a naming set M is called *a naming function.* A finite set of naming functions

$$T(m) = \{m, \phi_1(m), \ldots, \phi_n(m)\}, \tag{2}$$

in which for any $m \in M$ $\phi_i(m) \neq \phi_j(m)$, $i, j \in \{1, \ldots, n\}$, $i \neq j$, is called *a template.* A set of names $T(m_i) = \{m_i, \phi_1(m_i), \ldots, \phi_n(m_i)\}$ is referred to as *a template element* for the cell named m_i.

Each template element $T(m_i)$ is characterized by a *weight vector* $v_i \in V$,

$$v_i = \{w(m_i, \phi_1(m_i)), \ldots, w(m_i, \phi_n(m_i)\}, \tag{3}$$

whose components are interpreted as weights of the connections between the cells named m_i and $\phi_j(m) \in T(m)$.

Definition 4. A mapping $S(m) \Rightarrow K(A, M)$, expressed by the equation

$$S(m) = \{(a_0, m)(a_1, \phi_1(m)), \ldots, (a_n, \phi_n(m))\}, \tag{4}$$

in which the set of naming functions constitute a template, is called *a configuration*, the vector $S_A(m) = (a_0, a_1, \ldots, a_n)$ being referred to as *a state configuration vector.*

A configuration is called *k-bounded* if each name m_i appears not more than in k elements of it. A bounded configuration is called *local* if a metric is given on a set of names, and within this metric a sphere of a finite radius may be constructed such, that for each $m_i \in M$ all cell names appearing in $S(m_i)$ are inside this sphere. The configuration is *global* if there exists a cell $m' \in M$ such that the amount of cells with this name depends on the size of the array.

Definition 5. The expression of the form

$$\theta : S'(m) \Rightarrow S''(m), \tag{5}$$

where S' and S'' are configurations, is called *a parallel substitution* (*substitution* for short).

$S'(m)$ contains two parts separated by the sign $*$, $S'(m) = S'_1(m) * S'_2(m)$. The first part

$$S'_1(m) = \{(a_0, m), (a_1, \phi_1(m)), \ldots, (a_n, \phi_n(m))\}, \quad a_i \in A_0 \cup A_x,$$

is referred to as *a base*, and the second part

$$S'_2(m) = \{(b_1, \psi_1(m)), \ldots, (b_l, \psi_l(m))\}, \quad a_i \in A_0 \cup A_c$$

– as *a context.* In the right-hand side of (5)

$$S''(m) = \{(f_0, m), (f_1, \phi_1(m)), \ldots, (f_n, \phi_n(m))\},$$

f_j, $j = 1, \ldots, n$, are nonlinear (sigmoid or threshold) functions of the inner product of the state configuration vector by the weight vector. The most frequently used is the function

$$f_j = \begin{cases} a, & \text{if } \sum_{j=1}^{n} a_j w_j \geq 0, \\ b, & \text{otherwise,} \end{cases} \tag{6}$$

where a_j, w_j are components of $S_A(m_i)$ and v_i respectively, $a, b \in A_0$. It is worth to pay attention that we use here *stationary substitutions* which have configurations S_1' and S'' generated by the same template.

A substitution is considered to be *applicable* to a cellular-neural array $N \in K(A, M, V)$, if there exists $M' \subset M$ such, that for any $m_i \in M'$ the following holds:

$$S_1'(m_i) \cup S_2'(m_i) \subseteq C(N). \tag{7}$$

If $S_1'(m)$ contains variable symbols, then the substitution θ is applicable at any $m_i \in M$, where all components of $S_A(m_i)$ are in A_0. An application of θ to an array N results in substituting the states a_i in the cells of $S_1' \subseteq C(N)$ for the values of f_i in the equally named cells of the right-hand side of (5). This is done simultaneously all over the array.

The set of substitutions $\Phi = \{\theta_1, \ldots, \theta_k\}$ constitute a *parallel substitution system or a PSS*. A PSS is applied to a cellular-neuron array in accordance with the following procedure.

Procedure 1. Let $N \in K(A, M, V)$ be a cellular neural array to be processed by Φ, $C(N)_t$ – an array at the t-th step of computation, $C(N)_0 = C(N)$. Then

1) if there exists a nonempty subset of substitutions $\Phi' \subset \Phi$, applicable to $C(N)_t$, then all of them are applied simultaneously, the obtained array being $C(N)_{t+1}$;
2) if the application of Φ does not change the array or no substitution is applicable to $C(t)$, then $C(N)_t$ is the result of the computation, denoted as $\Phi(N)$.

The determinacy of computations generated by a given PSS Φ is provided by its noncontradictoriness [3]. This property guarantees that the application of Φ to any $N \in K(A, M, V)$ results in a cellular-neuron array, i.e., such one which contains no equally named neurons.

Definition 6. A noncontradictory substitution set Φ, applied to $N \in K(A, M, V)$ in accordance with Procedure 1 is called *a cellular-neuron algorithm* which is represented by a pair $\langle \Phi, K(A, M, V) \rangle$.

A cellular-neural algorithm has a direct interpretation by an artificial neural network. This network consists of a set of processing elements, $P = p_1, \ldots, p_q, q = |M|$, which corresponds to the set of neurons. Each processing element should be capable to perform the following:

1) to be in any state from a state alphabet,
2) to perform the non-linear functions indicated in the right-hand sides of all substitutions, and
3) to store the connection weight vector.

The connections between processing elements are predetermined by the substitution system Φ. If Φ consists of a single substitution, then the weight vector contains the names from $T'(m_i)$, T' being the template generating the configuration of the left-hand side of the substitution. In the general case the connection vector of p_i corresponds to the union of template elements generating the left-hand sides $S'(m)$ from (5) of all substitutions $\theta \in \Phi$.

Definition 7. Let $T'_j(m)$ be the template, generating the substitution $S'_j(m)$, which constitutes the left-hand side of $\theta \in \Phi$. Then the set of names

$$Q'(m_i) = \bigcup_{j=1}^{k} T'_j(m_i), \quad k = |\Phi|, \tag{8}$$

is called *a neighborhood* of a cell named $m_i \in M$.

The neighborhood is represented, sometimes, as a vector

$$Q(m_i) = (m_i, m'_1, \ldots, m'_r),$$

where the neighbors are ordered in an arbitrary way, but, once chosen, the order is kept when forming the *neighborhood weight vector*,

$$v_i = (w(m_i, m_i), w(m_i, m'_1), \ldots, w(m_i, m'r)) = (w_0.w_1, \ldots, w_r),$$

and *neighborhood state vector*

$$S'(m_i) = (a_0, a_1, \ldots, a_r),$$

which characterize the neuron named m_i.

Example. Let $N \in K(A, M, V)$ be given such, that $A = \{-1, 1\} \cup \{x, y, z, u, v\} \cup \{\alpha, \beta\}$. $M = \{\langle i, j\rangle : i = 0, 1, 2, 3;\ j = 0, 1, 2, 3, 4\}$. The initial cellular array is $C(N)_0 = \{(1, \langle 1, 1\rangle)(1, \langle 0, 2\rangle)(1, \langle 0, 3\rangle)(\alpha, \langle 0, 4\rangle)(\alpha, \langle 1, 4\rangle)(\alpha, \langle 2, 4\rangle)(\beta, \langle 3, 4\rangle)\}$, the state of other cells being equal to -1 (Figure 1a).

The following templates are used:

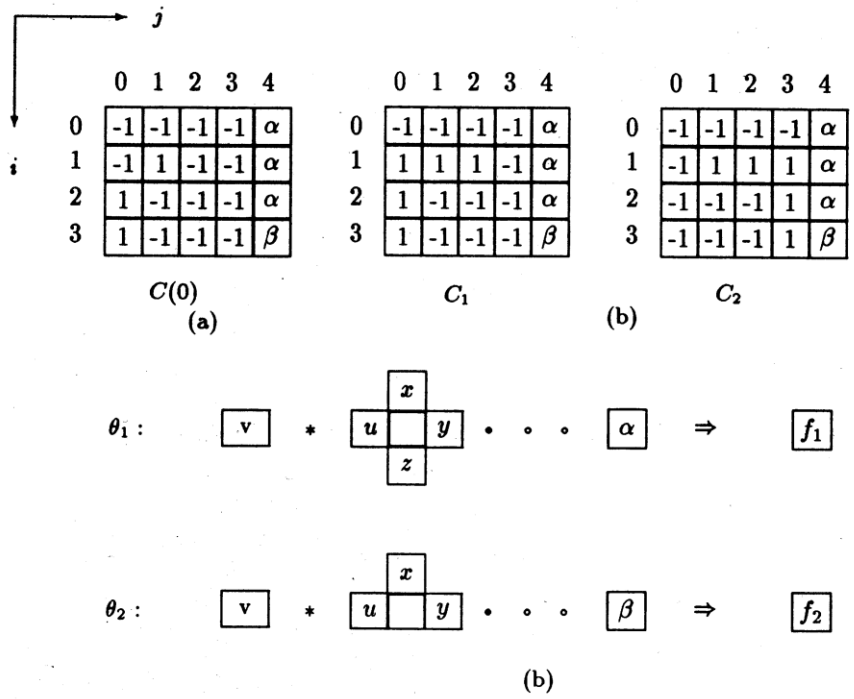

Figure 1

$$T_1 = \{\langle i,j\rangle\},$$
$$T_2 = \{\langle i-1,j\rangle, \langle i,j+1\rangle, \langle i+1,j\rangle, \langle i,j-1\rangle\, \langle i,4\rangle\},$$
$$T_3 = \{\langle i-1,j\rangle, \langle i,j+1\rangle, \langle i,j-1\rangle, \langle i,4\rangle\},$$

The neighborhood of cells whose names have $i = 0, 1, 2$ corresponds to the template $T_1 \cup T_2$, the neighborhood of cells whose names have $i = 3$ – to the template $T_2 \cup T_3$. Neighbor numbering corresponds to that of naming functions in the template. The weight vectors $w(\langle i,j\rangle) = (w_1, \ldots, w_5) : \langle i,j\rangle \in M\}$ are computed according to the formula: $w_k = a_{ij}a^k_{ij} + b_{ij}b^k_{ij}$, where a^k_{ij} and b^k_{ij} are the states of k-th neighbors of the cells $(a_{ij}, \langle i,j\rangle)$ and $(b_{ij}, \langle ij\rangle)$ in the two given stored patterns, respectively (Figure 1b). The weight vectors are given in the table below.
The set of substitutions $\Phi = \{\theta_1, \theta_2\}$:

$$\theta_1 = \{(v, \langle i,j\rangle)\} * \{(x, \langle i-1,j\rangle)(y, \langle i,j+1\rangle)(z, \langle i+1,j\rangle)(u, \langle i,j-1\rangle)$$
$$(\alpha, \langle i,4\rangle)\} \Rightarrow \{(f_1, \langle i,j\rangle)\},$$
$$\theta_2 = \{(v, \langle i,j\rangle)\} * \{(x, \langle i-1,j\rangle)(y, \langle i,j+1\rangle)(u, \langle i,j-1\rangle)\}$$
$$(\beta, \langle i,j\rangle)\} \Rightarrow \{(f_2, \langle i,j\rangle)\},$$

where

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

$$f_1 = \begin{cases} 1 & \text{if } w_1v + w_2x + w_3y + w_4z + w_5u \geq 0, \\ -1 & \text{otherwise,} \end{cases}$$

$$f_2 = \begin{cases} 1 & \text{if } w_1v + w_2x + w_3y + w_4u \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

The application of Φ to the cellular-neuron $\langle C, W \rangle$ after one iteration results in the stored pattern $C(N)_1 = C_1$.

3. Methods for cellular-neural algorithm synthesis

Synthesis procedure of a cellular-neural algorithm for solving a given problem consists in the following: 1) choice of the alphabet, 2) choice of the naming set, 3) computation of weight vectors, and 4) writing the substitutions.

The alphabet is determined by data representation in the problem to be solved. The state alphabet A_0 is a one-to-one mapping of data to be processed. For example, in the problems of pattern retrieval $A_0 = \{0, 1\}$ or $A_0 = \{1, -1\}$ for bistable patterns and $A_0 = \mathbf{R}$, if the patterns are in multigraduated grey or colored. The variable alphabet A_x contains as many symbols as is the cardinality of a neuron neighborhood. If the computation process requires controlling actions, then a set $A_c = \{\alpha, \beta, \ldots\}$ should be added.

The neural naming set is chosen according to data representation of the problem to be solved, and the way of mapping the given parameters to the set of neurons. Thus, in the optimization problems the neuron naming set is in correspondence with the set of parameters forming the domain, where the objective function is determined. When the optimization problems on graphs are to be solved, the naming set may be put in one-to-one correspondence with the set of vertices, circuits, edges or other parameters. If some kind of activity is to be optimized, the naming set is to be isomorphic to the set of actors or subjects the actors are dealing with. The names in these algorithms are symbols or digits, the naming functions are constant or in the form of shifts $\phi_i(m) = m + d$, where $d \in \mathbf{N}$. When the initial data are

represented as a picture in the space (image processing, pattern retrieval), the most frequently used is the set of coordinate in the Euclidean space,

$$M = \{\langle i, j, k\rangle : i = 0, \ldots, n_i, j = 0, \ldots, n_j, k = 0, \ldots, n_k\}.$$

The templates may contain any kinds of naming functions defined on M, but the most usable are also shifts and constants. The neighborhood or at least its admissible size also should be chosen at this stage of synthesis. It is done according to the requirements imposed by the implementation conditions. The naming set may consist of several subsets, $M = \{M_i : i = 1, \ldots, q\}$, each M_i being peculiar to its own cellular-neuron algorithm Φ_i. The interactions of Φ_i with any other Φ_j is represented by means of context configurations in substitutions of Φ_i containing naming functions defined on M_j.

Weight vector set determination is the most important and the most labor consuming stage in cellular-neural algorithm synthesis. This part of synthesis constitutes the "learning process" and should exploit the same approaches, than those of the ANN theory adapted to cellular-neural arrays. Though the methods of weight determining strongly depend on the problem to be solved, there is a fundamental concept forming the basis for all of them, which in its turn is based on the dynamic properties of cellular-neural algorithm operation. The point is that the cellular-neural array is an autonomous system, such that its dynamic characteristics depend on the weight vectors on the one hand, and on the other hand the computation results are in correspondence with its stable states. Hence, the weights may be determined starting from the dynamic stability condition of the array, which is characterized by the extremum of the following function.

$$E = -\sum_{m_i \in M} a_i \sum_{m_j \in Q(m_i)} w_j a_j, \tag{9}$$

where w_j and a_j are the equally indexed neighborhood weight vector and state neighborhood vector components, respectively. In the ANN theory the expression (9) is called *Liapunov's function* or *energy function* [4]. Its inverse $B = -E$ is used in the Boltzmann machine theory being called there as the *consensus function* [5]. The minimal value of E (the maximum of B) indicates that the cellular-neural array is in a stable state, i.e., being left to perform the cellular-neural functions without the outer intervention, it does not change neuron states. From above it follows that the strategy for weight vector synthesis should be grounded on the following requirements.

1. The result of computation should be represented by a stable state of the array. Formally it is expressed as follows. Let the result of applying a cellular-neural algorithm to an array $N \in K(A, M, V)$ be $\overline{N}$ with the cellular array $C(\overline{N})$, corresponding to the result of the computation. Then for the resulting array the following condition is to be met:

$$\Phi(\overline{N}) = \overline{N}. \tag{10}$$

2. Starting at any initial cellular array $C(N) \in K(A, M)$ the algorithm should reach a stable state, i.e., no oscillation should occur in the array.
3. There should be no spurious stable states, which do not represent any result in the problem.

It follows from above that the methods of weight vector synthesis are based on solving the set of equations of the form (10), or comparing the given parameters of the problem with the condition (10). It is not always possible to perform this precisely, therefore some approximate and iterative methods are developed and studied. Sometimes, the expression for energy function increment caused by a state change in a single neuron

$$\Delta E_i = -2a_i \sum_{m_j \in Q(m_i)} w(m_i, m_j) a_j \tag{11}$$

may be used as well. Till now in the ANN theory there are no methods of determining the connection weights, which provide all three above requirements to be met completely. Though some approaches are developed which succeeded to come close to the aim [9]. As for weight vector determining in cellular-neural synthesis, the problem is only touched [8]. We hope that the capabilities of celullar-neural model would help to develop proper synthesis techniques.

Moreover, the learning algorithms are essentially cellular, because the connection weights are computed as the functions of neighbor neuron states of the prototypes (stored patterns) the computations being independent for each neuron and, hence, may be executed in parallel.

The substitution system is formed according to the rules of PSA theory [3]. The main peculiarity is as follows. The cellular-neural algorithm is represented by a PSS containing functional substitutions performing sigmoid function of the inner product of two vectors. These functional substitutions are structurally similar to the class of Neumann substitutions (the cardinality of the base is equal to 1).

The PSS should contain also substitutions which makes the algorithm to stop and to generate a signal of termination when the result is obtained. Some general techniques for that are described in [3], but in any particular algorithm this is to be done taking into account the peculiarities of the problem.

The PSS for weight vector set determining is formed according to the methods of PSS theory. This "learning" algorithm belongs to a class of functional PSA's, operating on the same cellular-neural array, the weight vector components being considered as cell states.

4. Examples of cellular-neural algorithm application

The expressive capability of cellular-neural algorithm is displayed here by three simple examples chosen as representatives of three following classes of problems: 1) optimization on graphs, 2) retrieval of patterns, and 3) identification of figures. Some considerations based on the experience of study and computer simulation are suggested.

4.1. Optimization on graphs

The synthesis of cellular-neural algorithm for solving these problems consists of direct mapping the graph representation onto the cellular-neural array, the weight vector determination being reduced to the direct comparison of the objective function expression to that of the energy function (9).

Let a weighted graph $G = \langle V, E \rangle$, where $V = \{v_1, \ldots, v_{10}\}$ is the set of vertices, $E = \{e_{ij} : i, j = 1, \ldots, 10\}$ is a set of integers, interpreting the weights of edges, be given (Figure 2). A path between two given vertices v_1

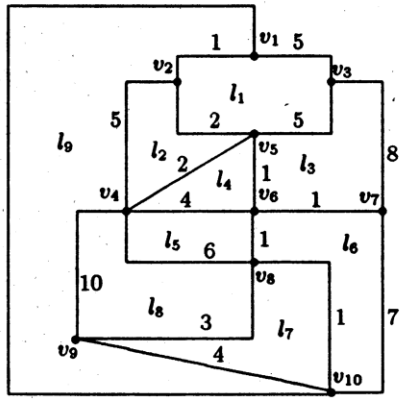

Figure 2

and v_{10} is to be found such, that the total weight of its edges is minimal. The problem is reduced to the problem of searching a circuit of minimal weight. The circuit is formed by adding an edge between v_1 and v_{10} with $e_{1,10} = 0$. The solution is sought as a stable state of a cellular-neural array corresponding to the minimum of the inverse of the energy function $B = -E$ (9). So, the set of neural cells is to be isomorphic to the set of simple circuits, representing the graph, the set of neurons in the state equal to 1 corresponding to the set of circuits forming the sought circuit. The algorithm is based on the calculation of the increment ΔB_i caused by the change of the state of the neuron named m_i (11), and then changing the states of

those neurons whose increments are minimal. The result is obtained when for all neurons the increments are positive, indicating that B has reached its minimum. The synthesis procedure according to Section 3, is as follows:

1. The alphabet $A = A_0 \cup A_x \cup A_c$, $A_0 = \{0, 1\}$, $A_x = \{x_0, x_1, x_2, x_3, k\}$, $A_c \subset \mathbf{N}$.

2. The naming set is the union of two subsets: $M \cup M'$, $M = \{m_1, \ldots, m_9\}$, $M' = \{m'_1, \ldots, m'_9\}$. Both are isomorphic to the set of simple circuits $L = \{l_1, \ldots, l_9\}$ of the graph. The naming functions used are constants. The templates are defined for each neuron separately, so that $T(m_i)$ contains m_i together with the names corresponding to the circuits adjacent to l_i. For example,

$$T(m_1) = (m_1, m_2, m_3, m_9), \quad T(m_2) = (m_2, m_1, m_4, m_9).$$

3. The weight vectors are computed according to the following assumptions:

a) The weight of any circuit l is the sum of weights of its edges. The weight of a simple circuit l_i is $e(l_i) = e_i^1 + \ldots + e_i^r$, where e_i^j is the weight of edge shared by l_i and l_j.

b) The weight of a circuit l composed of a subset $L' = \{l_i^1, \ldots, l_i^r\}$ of the simple circuits is as follows:

$$e(l) = \sum_{l_i \in L'} e(l_i) - 2 \sum_{l_i, l_j \in L'} e_i^j. \tag{12}$$

Comparing (12) with (9) it is easy to obtain the expression for computing the weight vectors.

$$w(m_i, m_i) = e(l_i), \quad w(m_i, m_j) = -2e_i^j, \quad m_i, m_j \in T(m_i).$$

For example,

$$v_1 = (13, -4, -10, -2), \quad v_2 = (9, -4, -4, -10).$$

4. The substitution set is represented here by two substitutions: θ_1, which performs the cellular-neural computation, and θ_2, which performs a controlling function, changing the threshold at each step of iteration.

$$\theta_1 : \{(x_0, m_i)\} * \{(x_i^1, m_i^1), \ldots, (x_i^4, m_i^4)(k, m'_i)\} \Rightarrow \{(f_i, m_i)\},$$
$$\theta_2 : \{(k, m'_i)\} \Rightarrow \{(f, m'_i)\},$$

where

$$f_i = \begin{cases} \overline{x} & \text{if } \sum_{j=1}^{4} x_i^j v_i^j \geq k, \\ x & \text{otherwise,} \end{cases}$$

$$f = \min_{i=1} \sum_{j=1}^{4} x_i^j v_i^j.$$

The initial cellular state is chosen arbitrarily, in our case it is $C(N)_0 = \{(1, m_1), (0, m_2), (1, m_3), (0, m_4), (1, m_5), (0, m_6), (1, m_7)(0, m_8)(1, m_9)\}$. The results of computations are given in the table bellow.

t	neuron states	energy decrements	k	B
0	1,0,1,0,1,0,1,0,1	−1, −5, −5, −3, −12, 3, 0, −19, 10	−19	46
1	1,0,1,0,1,0,1,1,1	−1, −5, −5, −3, 0, 3, 6, 19, 10	−5	29
2	1,1,0,0,1,0,1,1,1	−7, 5, 5, −5, 0, 5, 6, 19, 20	−7	19
3	0,1,0,0,1,0,1,1,1	7, 1, 15, −5, 0, 5, 6, 19, 20	−5	12
4	0,1,0,1,1,0,1,1,1	7, 5, 13, 5, 8, 5, 6, 19, 20	0	7

As it is seen from the table the result is obtained after the fourth iteration. The sought circuit is the composition of simple ones from the subset $L = \{l_2, l_4, l_5, l_7, l_8, l_9\}$, which yields the minimal weight path from v_1 to v_{10} going via the vertices: v_2, v_5, v_6, v_8, and having the weight equal to 7.

4.2. Pattern retrieval

The cellular-neuron algorithm for pattern retrieval descends from Hopfield's neural associative memory [4]. The problem is as follows. A number of patterns (prototypes) is given to be stored in the array, weight vector values providing the correspondence of each prototype to a stable state. So, if the initial cellular array is set to represent an arbitrary pattern, the algorithm should start the computations and terminate in a stable state corresponding to a prototype with closest resemblance to the initial pattern. This class of problems presents a good illustration of synthesis procedure, where a cellular algorithm (PSA) is used for weight vector determining, which gives the possibility to combine the learning and the processing stages.

Let a set of patterns $P = \{p_1, \ldots, p_s\}$, represented by two-dimensional white-black pictures, be stored in the array. Then the synthesis procedure is as follows.

1. Following Hopfield's model, A_0 is taken as $\{1, -1\} \cup \mathbf{N}$, $A_x = \{x, x_0, X_1, \ldots, x_r, v_0, \ldots, v_s\}$, $r + 1$ is the cardinality of the neighborhood, s – the number of prototypes.

2. The naming set consists of two subsets M and M', both being taken as sets of coordinates in a 3D Eucledian space. The first is the naming set for cellular-neuron array, where the main computations are performed,

$M = \{\langle i, j, k\rangle\}$. The ranges of i and j are defined according to the size of the patterns, $k = 0, 1, \ldots, r+1$. The plane $\{\langle i, j, r+1\rangle\}$ is dedicated for pattern representation, the cells named $\{\langle i, j, 0\rangle, \ldots, \langle i, j, r\rangle\}$ are for storing the weight vector $v_{ij} = (w_{ij}^0, \ldots, w_{ij}^r)$. The second subset $M' = \{\langle i', j', h\rangle\}$, i', j' ranging as i, j in M, $h = 1, \ldots, s$ is dedicated for storing the prototypes.

3. The method used for weight vector synthesis is based on a well-known iterative algorithm of perceptron learning [10]. Here it is presented in the form of a PSA. The initial array is composed of N and N', where $N \in K(A, M, V)$ all cell states being equal to 0, $N' \in K(A, M')$ whose planes represent the prototypes. The substitution set is $\Phi = \{\theta_1, \theta_2\}$, θ_1 performs the calculation of weights, θ_2 transfers the prototypes from N to N'. The geometrical representation of the configuration S' with $(r = 9)$, used in θ_1 is shown in Figure 3.

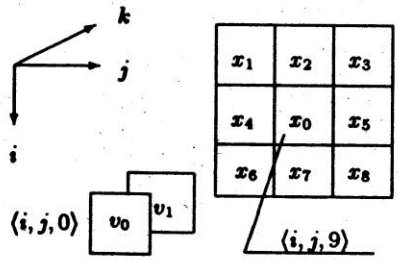

Figure 3

$$\theta_1: \{(v_k, \langle i, j, k\rangle)\} * \{(x_0\langle i, j, 9\rangle), (x_1, \langle i-1, j-1, 9\rangle), \ldots, (x_8, \langle i+1, j4+1, 9\rangle)(v_0, \langle i, j, 0\rangle), \ldots, (v_8, \langle i, j, 8\rangle)\} \Rightarrow \{(f, \langle i, j, k\rangle)\},$$

$$\theta_2: \{(v, \langle i', j', h\rangle)\} \Rightarrow \{(v, \langle i, j, 9\rangle)\},$$

where

$$f = \begin{cases} v & \text{if } v\sum_{j=1}^{r} x_j v_j > 0, \\ v + x_i x_j & \text{otherwise.} \end{cases}$$

These two substitutions show only the main operations of the algorithm. Some substitutions should be added to make them be applied alternatively, and to transfer the prototypes in turn until the weight become stable.

4. The main algorithm for the retrieval of a stored pattern consists of a single substitution, in which the same configuration S is used.

$$\theta: \{(x, \langle i, j, 9\rangle)\} * \{(x_1\langle i-1, j-1, 9\rangle) \ldots, (x_8, \langle i+1, j+1, 9\rangle) (v_0, \langle i, j, 0\rangle), \ldots, (v_8, \langle i, j, 8\rangle)\} \Rightarrow \{(f, \langle i, j, 9\rangle)\},$$

where

$$f = \begin{cases} 1, & \text{if } \sum_{j=0}^{j=8} x_j v_j \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

The simulation of the algorithm shows that all prototypes correspond to stable states and more than r patterns may be stored.

4.3. Image processing

Here the problem of making some transformation on images is briefly considered. The image is given in a colored or in a multigraduated gray form. The most typical problems are the following: to classify the cells or the areas by colors, to determine the locations of some given symbols or figures, to clean the image from the noise, to distinguish contours, edges, angles, etc. The peculiarity of cellular-neural algorithms for these problems is in that the weights are associated with the configurations (cloning templates in [6]) of the substitution rather than with the cellular-neural array. In the neural-cellular algorithm synthesis the alphabet is the set of codes of the colors, representing the image. The naming set is a set of coordinates of a 2D plane, the size being determined by the size of the image. The templates are usually taken in the form of squares or crosses. The main difficulty is to determine the weight vector. There is no regular methods for doing this, except the intuitive choice with subsequent simulation. The substitution set consists of functional substitutions, the function being in the form of (6).

5. Conclusion

A formal model for fine-grained parallel computations is presented. The model combines the features of Parallel Substitution System and Artificial Neural Nets. It is shown that such combination provides new useful possibilities in representation, synthesis, and simulation of neural-net algorithms with bounded number of each neuron connections. The future investigations are to be directed to develop method of cellular-neuron algorithms synthesis which provide satisfactory quality of problem solution and to define the field of model application.

References

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

[2] W.S. Mc-Culloch, W. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bull. of Math. Biophysics, Vol. 5, 1943, p. 115.

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

[4] J.J. Hopfield, D.W. Tank, *Computing with Neural Circuits: a Model*, Science, Vol. 233, 1986, p. 625.

[5] J.H.M. Korst, E.H.L. Aarts, *Combinatorial optimization on Boltzmann machine*, Journ. of Parallel and Distributed Computing, Vol. 6, 1989, p. 331.

[6] L.O.Chua, I.Yang, *Cellular Neural Networks: Theory amd Application*, IEEE Transactions, Vol. CS-35, No. 10, 1988, p. 1257.

[7] J. Zhang, I. Zhang, D. Yan, A. He, L. Liu, *Local interconnection neural network and its optical implementation*, Optics Communication, Vol. 102, 1993, p. 13.

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

[9] 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, Vol. CS-37, No. 11, 1990, p. 1356.

[10] F. Rosenblatt, *Principles of Neurodumamics*, Washington, Spartan, 1959.