Available online at www.sciencedirect.com
ScienceDirect
Physica D 237 (2008) 1226–1234
PHYSICA D
www.elsevier.com/locate/physd

# Statistical physics on cellular neural network computers

M. Ercsey-Ravasz[a,b,*], T. Roska[a], Z. Néda[b]

[a] *Pázmány Péter Catholic University, Department of Information Technology, HU-1083 Budapest, Hungary*
[b] *Babeş-Bolyai University, Department of Physics, RO-400084 Cluj, Romania*

Available online 25 March 2008

**Abstract**

The computational paradigm represented by Cellular Neural/nonlinear Networks (CNN) and the CNN Universal Machine (CNN-UM) as a Cellular Wave Computer, gives new perspectives also for computational statistical physics. Thousands of locally interconnected cells working in parallel, analog signals giving the possibility of generating truly random numbers, continuity in time and the optical sensors included on the chip are just a few important advantages of such computers. Although CNN computers are mainly used and designed for image processing, here we argue that they are also suitable for solving complex problems in computational statistical physics. This study presents two examples of stochastic simulations on CNN: the site-percolation problem and the two-dimensional Ising model. Promising results are obtained using an ACE16K chip with $128 \times 128$ cells. In the second part of the work we discuss the possibility of using the CNN architecture in studying problems related to spin-glasses. A CNN with locally variant parameters is used for developing an optimization algorithm on spin-glass type models. Speed of the algorithms and further trends in developing the CNN chips are discussed.


*PACS:* 89.20.Ff; 07.05.Tp; 05.10.Ln

*Keywords:* Cellular neural network; CNN universal machine; Unconventional computing; Cellular wave computers

## 1. Introduction

Progress in computation is always driven and deeply influenced by the available technology. For example the circuit implementation of the building blocks (e.g. the circuits for memory) played a significant role when John Von Neumann invented the digital stored programmable computer [1]. For a long period after this breakthrough, computation was approached by using discrete variables and the instructions were defined via arithmetic and Boolean logic.

In the light of the presently emerging quantitative neuroscience, it became possible to understand the signal representation and processing in some parts of our nervous system. Parallel with this, a new and revolutionary way of computing is emerging. Today the technology has developed so much that it is possible to imitate some basic principles of our nervous system. Several thousands of microprocessors (cells, neurons) can be placed on a single chip locally interacting with each other, similar to a layer of neurons. The new cellular visual microprocessor Q-Eye [2], for example, has 25000 processors, each one hosting 4 optical sensors. One suggested prototype architecture for an unconventional computation is the Cellular Wave Computer [3,4], a special case of it being the Cellular Neural Network Universal Machine (CNN-UM) [5].

The history of CNN computing starts in 1988 when the theory of cellular neural/nonlinear networks (CNN) was presented [6]. Few years later a detailed plan for a computer using cellular neural networks was developed. This is called CNN Universal Machine (CNN-UM) [5] and is an analogic (analog+logic) computer which has on its main processor several thousands of interconnected computational units (cells), working in parallel. Since then many experimental hardware have been developed and tested. These chips can be easily connected to digital computers and programmed with special languages. Although the CNN computer is proved to be a universal Turing machine [7] its structure and properties make

* Corresponding author at: Pázmány Péter Catholic University, Department of Information Technology, HU-1083 Budapest, Hungary. Tel.: +36 1 886 4760.
*E-mail addresses:* ravasz@digitus.itk.ppke.hu (M. Ercsey-Ravasz), roska@itk.ppke.hu (T. Roska), zneda@phys.ubbcluj.ro (Z. Néda).

0167-2789/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.physd.2008.03.028

it suitable only for some special complex problems and it is complementing and not replacing the digital computers.

CNN chips are generally used and developed for fast image processing applications. The reason for this is that the cells can be used as sensors (visual or tactile) as well. CNN computers can work thus as a fast and "smart" camera, on which the capturing of the image is followed in real time by analysis [8]. Several applications related to physics and mathematics were also considered: solving in an elegant manner partial differential equations [9,10] or studying cellular automata models [11,12]. These applications result straightforwardly from the appropriate structure of the CNN chips.

The aim of this article is to show how the CNN-UM can be effectively used for studying complex problems in computational statistical physics: stochastic cellular automata, Monte Carlo type simulations, optimization problems etc. We will present results obtained on an experimental version of CNN-UM: the ACE16K chip [13] which has $128 \times 128$ cells. After discussing in a critical manner the obtained results and the developing trend of CNN computers, we argue that CNN-UM could represent a useful computational alternative in the near future.

The paper is organized as follows: in the next (second) section we present a short description of the basic principles and architectures of the CNN computers, giving also a short review of the most important applications. In the third section after discussing the basics (random number generation) for doing stochastic simulations on the CNN-UM, two classical problems of statistical physics are considered as examples: the site-percolation problem and the two dimensional Ising model. Both of them offer an opening to a broad class of problems and, as such, the presented algorithms can be easily generalized for other closely related models as well. Next we discuss the relation between CNN and spin-glass type models. We will show that using a locally variant CNN, fast and simple optimization algorithms can be developed.

## 2. The CNN universal machine

Cellular wave computers [4,3] are universal machines on flows, meaning that the type of data to be processed are topographic flows of cell array signals. The instructions are defined in space and time, typically as a spatial-temporal wave acting on the flow of data. One special case of these cellular wave computers is the CNN Universal Machine (CNN-UM) [5] in which for this spatial-temporal dynamics a standard cellular neural network (CNN) [6] is used. The physical implementations of these computers are numerous and widely different: mixed-mode CMOS, emulated digital CMOS, FPGA, and also optical. In the following we will usually refer to mixed-mode implementations.

The standard CNN [6] is composed of $L \times L$ cells placed on a square lattice and interconnected through their 8 neighbors (Fig. 1). Each cell is characterized by a state value, $x_{i,j}(t)$. This usually describes a voltage in the circuit of the cell, but the particular physical correspondent of this quantity might vary with the different physical implementations of the chip. The cell

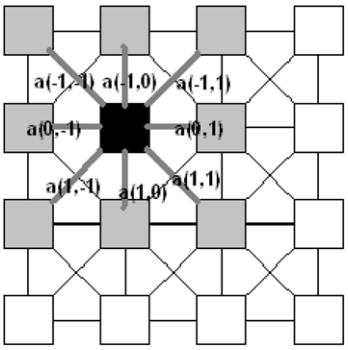


Fig. 1. The lattice structure of the standard CNN. Each cell is connected to its 8 neighbors.

has also an input value (voltage) $u_{i,j}$, which is constant in time and can be defined at the beginning of an operation. The third characteristic quantity of the cell is the output value $y_{i,j}(t)$. This is equivalent to the $x_{i,j}$ state value in a given range. More specifically it is a piece-wise linear function, bounded between $-1$ (called as white) and 1 (black):

$$y = f(x) \equiv \frac{1}{2}(|x+1| - |x-1|). \tag{1}$$

The wiring between neighbor cells assures that the state value of each cell can be influenced by the input and output values of its neighbors. In the original model as well as on CNN chips developed, wiring exists with 8 neighbors (the 4 nearest and the 4 next-nearest neighbors). The equation governing the dynamics of the CNN cells results from the time-evolution of the equivalent circuits and has the following form [6]:

$$\frac{\mathrm{d}x_{i,j}(t)}{\mathrm{d}t} = -x_{i,j}(t) + \sum_{k=-1}^{1}\sum_{l=-1}^{1} A_{k,l}\, y_{i+k,j+l}(t) + \sum_{k=-1}^{1}\sum_{l=-1}^{1} B_{k,l}\, u_{i+k,j+l} + z_{i,j}, \tag{2}$$

where $i, j$ denotes the coordinates of the cell and the neighbors are identified with the help of $k$ and $l$ numbers ($k, l = \{-1, 0, 1\}$). Self-interaction ($k = 0, l = 0$) is also possible. The coupling between neighbors is controlled with matrices $A$ and $B$ which are a $3 \times 3$ matrix identical for all cells ($A = \{a_{-1,-1}, a_{-1,0}, a_{-1,1}; a_{0,-1}, a_{0,0}, a_{0,1}; a_{1,-1}, a_{1,0}, a_{1,1}\}$ and $B = \{b_{-1,-1}, b_{-1,0}, b_{-1,1}; b_{0,-1}, b_{0,0}, b_{0,1}; b_{1,-1}, b_{1,0}, b_{1,1}\}$). We emphasize however that in Section 4, we will also consider locally variant CNN, in which the coupling parameters ($A$, $B$ matrices) can be different for each cell. Parameters $z_{i,j}$ are constant values and can also vary from cell to cell. The set of parameters $\{A, B, z\}$ is called a template. An operation is performed by giving the initial states of the cells, the input image (the input values of all cells) and by defining a template. The states of all cells will vary in parallel and the result of the operation will be the final steady state of the CNN. If the state values ($x_{i,j}$) of all cells remain bounded in the [-1, 1] region (i.e. $y_{i,j} = x_{i,j}$ holds for each cell at any time $t$), then each operation is equivalent with solving a differential equation defined by the template itself [14,9,10]. When $x_{i,j}$ does not

remain between 0 and 1 then the piece-wise linear function described at the definition of the output value $y_{i,j}$ Eq. (1) takes an important role. The final steady state will not be simply the solution of the differential equation, and this can be used for defining other useful operations as well [14].

The CNN-UM [5] is a programmable analogic cellular wave computer. Beside the analog circuit described by Eq. (2) each cell also contains a logic unit, local analog and logic memories and a local communication and control unit. The logic unit and logic memories are included to complement the analog computation, this way very basic logic operations can be performed without needing to define complicated templates. In the local logic memories one can save a binary value (1 and 0 respectively), and in the local analog memories it is possible to save real values between $-1$ and 1. Since the whole CNN array is used mainly for image processing and acquisition, the binary values are often referred to as black and white and the real values bounded between $-1$ and 1 are mapped in a grayscale scheme. Beside these local units, the CNN-UM has also a global analog programming unit which controls the whole system, making it a programmable computer. It can be easily connected to PC type computers and programmed with special languages, for example the analogic macro code (AMC).

Many applications ideal for the analogic architecture of the CNN-UM were already developed and tested. Studies dealing with partial differential equations (PDE) [9,10,15,16] or cellular automata (CA) models [11,12] prove this. Solving partial differential equations is relatively easy and offers the advantage of continuity in time [9]. Deterministic cellular automata [11] with simple nearest-neighbor rules are also straightforward to implement in the CNN architecture.

For practical purposes the most promising applications are for image processing, robotics or sensory computing purposes [17], so the main practical drive in the mixed-mode implementations was to build a visual microprocessor [4]. The cells of the computer are additionally equipped with sensors (usually visual sensors, but in special applications these can be tactile or of other type). By this way optical sensing and computing are inherently topographically coupled and implemented on the surface of a silicon chip. The first self-contained camera–computer, containing a CNN-UM type visual microprocessor (ACE16K), was built recently, as the Bi-i camera–computer [8]. Its input flow speed achieves 20000 frames/s including sensing and processing.

On Table 1. we can see the size of some CNN chips realized in the last decade. It is important to mention that besides increasing the lattice size of the chips engineers are focusing now on developing multi-layered, 3 dimensional chips as well. This trend is opening again new and fascinating application possibilities.

Table 1
Evolution of the CNN-UM chip, different physical realizations

| Name | Year | Size |
|---|---|---|
| – | 1993 | $12 \times 12$ |
| ACE440 | 1995 | $20 \times 22$ |
| POS48 | 1997 | $48 \times 48$ |
| ACE4k | 1998 | $64 \times 64$ |
| CACE1K | 2001 | $32 \times 32 \times 2$ |
| ACE16K | 2002 | $128 \times 128$ |
| XENON | 2004 | $128 \times 96 \times 2$ |
| Q-EYE | 2006 | $176 \times 144$ |

From these chips only the ACE16K [13] and the Q-Eye [2] are commercially available, mass production began with the Q-Eye at the end of 2006.

## 3. Stochastic simulations on the CNN universal machine

Working with analog values leads to the presence of noise and one would naturally think of applications in which this noise (usually thermal and Nyquist type noise) can be useful. In computational statistical physics many of the interesting problems deal with stochastic processes or Monte Carlo type simulations. Thus, generating random numbers is a key issue for stochastic cellular automaton, random initial conditions or other MC simulations on lattices (lattice spin problems, population dynamics models, lattice gas models, percolation etc...). Developing and proving the efficiency of stochastic simulation techniques on the CNN-UM - using its stored (or algorithmic) programmability - would thus be an important step toward its success. For implementing this kind of simulations on the CNN-UM one first needs a good and trustworthy random-number generator. The natural noise of the chip can not be used directly for generating random numbers since it is highly correlated spatially and in time as well. There are, however, other simple possibilities to generate uncorrelated but still realistic (not pseudo-) random numbers in the CNN architecture.

In [18] a realistic random number generator (RNG) which uses also the natural noise of the CNN-UM chip was presented. The method is based on a chaotic cellular automaton (CA) which uses simple logic functions and is already accepted as a good pseudo-random number generator (called PNP2D [12,19]). It generates binary values (0 and 1) with the same $1/2$ probability independently of the starting condition. This pseudo-random number generator perturbed periodically by the natural noise of the chip can lead to a realistic random number generator.

In the algorithm proposed by us, the chaotic CA is perturbed in each time step with a noisy binary picture (array). This picture is obtained by performing a cut on a grayscale picture. The initial grayscale picture is obtained by fixing a value $a$ for the state of each cell. Due to the analog nature of the CNN-UM chip one will always get a few randomly located cells with state values greater than $a + z$ ($z \ll 1$). Selecting these cells (i.e. making the cut) will result in a random binary picture with few black pixels (black will be the selected cells and white the other ones). These noisy binary images might be strongly correlated and will fluctuate in time. The time-like fluctuations are caused by real stochastic processes in the transistor circuits of the chip and can not be thus controlled. Because the used noisy images contain only very few black pixels the perturbation — which consists of a simple exclusive-or operation — will just slightly sidetrack the chaotic CA from the original deterministic path and all the good properties ($1/2$ density of black pixels and small correlations) of the pseudo-

random number generator will be preserved. A method for generating pictures with any $p$ probability of the black pixels was also developed. For more details see [18]. This RNG and the described algorithms were tested and are properly working on an ACE16K chip which is an experimental version of the CNN-UM with $128 \times 128$ cells. It is found that the RNG with $p = 0.5$ is already almost 5 times faster on the ACE16K than on modern PC type digital computers (see [18]).

Once a properly working RNG is available, Monte Carlo type simulations on two-dimensional lattice-type models are possible. Generating random initial conditions for cellular automata models is straightforward and many simple stochastic lattice models can be relatively easily solved. Here we consider two well-known problems of statistical physics: the site-percolation problem and the two dimensional Ising model.

### 3.1. The site-percolation problem on the CNN-UM

Percolation type problems are very common in many areas of sciences like physics, biology, sociology and chemistry (for a review see e.g. [20]). Different variants of the problem (site percolation, bond percolation, directed percolation, continuum percolation etc.) are used for modeling various natural phenomena [21]. As an example, the well-known site percolation problem is widely used for studying the conductivity or mechanical properties of composite materials, the magnetization of dilute magnets at low temperatures, fluid passing through porous materials, forest fires or propagation of diseases in plantations etc. The site-percolation model exhibits a second order geometrical phase transition and it is important also as a model system for studying critical behavior [22].

Implementing the site-percolation problem on the CNN-UM the array of cells will represent the square lattice on which percolation is studied. In the following the state of the lattice will be referred to as a binary picture, black pixels pointing for activated and white pixels for non-activated sites. In the classical site-percolation problem one is interested whether it is possible or not to go from one side of the picture to the opposite side through activated and neighboring pixels. If there is a path that satisfies this condition, it is considered that the black (activated) pixels percolate through the lattice.

Finding the complicated connected path through neighboring black pixels takes only a single operation on the CNN chip. We are using a template (the parameters in Eq. (2)), called *figure reconstruction* with parameters:

$$\mathbf{A} = \begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad z = 3.$$

On CNN chips the wiring allows us to consider 8 neighbors for each cell, but solving site-percolation with 4 nearest neighbors is also possible. The template for this case would be: $A = \{0, 0.5, 0, 0.5, 4, 0.5, 0, 0.5, 0\}$,$B = \{0, 0, 0, 0, 4, 0, 0, 0, 0\}$,$z = 1$.

The input picture of the template is the actual random binary image and the initial state will contain only the first row of the image. The template values are chosen in a way that pixels which have an input value equal to 1 (are black) and have at least one neighbor with state value 1 will become black. In this manner a flow starts from the first row making black all the pixels which were black on the input picture and are connected through neighbors to the first row. If on the final output there remain black pixels in the last row, then percolation exists. This simple template is a function in the image processing library of the Bi-i v2 [23]. In general the influence of the $A, B, z$ parameters can be checked by analyzing the corresponding differential equation system. A small programming experience with the most common templates can help us defining the proper template for a new operation. It is immediate to realize that by using a locally variant CNN in which these $A$ and $B$ matrices can vary from cell to cell also bond-percolation and directed percolation problems would be solvable with the same simple algorithm. In these cases the $A$ matrices should be separately and consistently defined for each cell, having non-zero elements only in the directions of the predefined bonds.

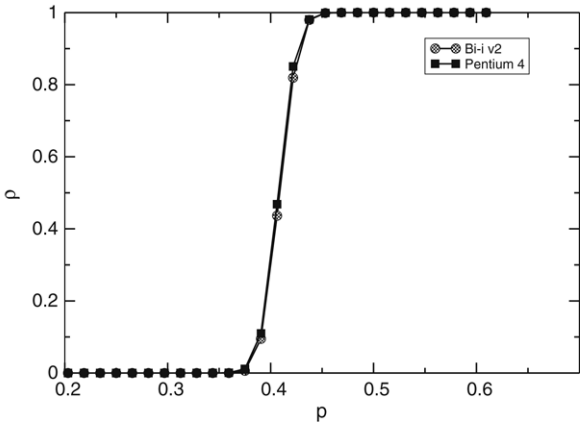


Fig. 2. Simulated site-percolation probability as a function of the density of black pixels. Circles are results obtained on the CNN-UM chip, squares are simulation results on a normal PC type digital computer.

Applying this template on many random images and by changing also the $p$ activation probabilities it is possible to study the phase-transition in the classical site-percolation problem. After a good statistics it is possible to determine how the probability of percolation $\rho$ depends as a function of the density of activated pixels, and consequently where the critical density $p_c$ is. The CNN code written for this application can be downloaded from [24].

Results for the $\rho(p)$ curve obtained on the ACE16K chip are plotted with circles in Fig. 2. On the same graph it is also sketched with square symbols the MC simulation results obtained on a digital Pentium 4, 2.8 GHz computer, using a recursion-type algorithm for detecting percolation. The lattice size in both cases is $128 \times 128$ and the results are averaged for 10000 different random images for each activation probability values. The two curves show a pretty good agreement. The percolation threshold resulting from the simulated $\rho(p)$ curves are in good agreement with the accepted $p_c$ critical value for this case (site-percolation on a square lattice with 8 neighbors): $p_c = 0.407$ [25].

Comparing the speed of the Monte Carlo type simulations performed on digital computers and on the ACE16K chip

the following observations can be summarized: (i) with the presently available chip size ($L = 128$) and the experimental version of the CNN-UM the simulation is almost 10 times slower than on a digital computer with a 2.8 GHz Pentium 4 processor, (ii) on the CNN-UM the time needed for detecting percolation grows linearly with the linear size of the respective image, while on digital computers it scales with the square of the linear size of the lattice. Increasing thus the size of the chip will definitely favor the CNN-UM architecture for such Monte Carlo type simulations.

In the site-percolation problem binary images are used, the state of the system being saved in local logic memories. The ACE16K chip was developed mainly for image processing on analog images and the number of local logic memories is thus small. This inconvenience effects considerably the implementation of the algorithm since many extra data transfer has to be done. We have to mention also that some of the CNN instructions are not efficient on the present chip. The new chips (presently under fabrication) are capable of performing these operations much more efficiently, increasing further the computational speed on the CNN-UM.

### 3.2. The Ising model on the CNN-UM

As a second specific problem in statistical physics we now consider the well-known two-dimensional Ising model. Implementing an MC study for this model on the CNN-UM is however not trivial. As it will be argued later a straightforward application of the usual Glauber [26] or Metropolis [27] algorithms could lead to unexpected problems exactly due to the completely parallel architecture of the dynamics.

In the Ising model, the spins can have two possible states $\sigma = \pm 1$. On the CNN-UM these spin states can be mapped as "black" or "white" states of the cells. Without an external magnetic field the hamiltonian of the system is

$$H = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j, \tag{3}$$

$\langle i, j \rangle$ representing nearest neighbors. There are many different MC type methods for studying this basic lattice model. Most of them like the Metropolis [27] or the Glauber [26] algorithm are of serial nature, meaning that at each step we update one single spin. Working however in parallel with all spins, could create some unexpected problems since nearest neighbors are updated simultaneously. Imagine for instance an initial state where the spin-values are assigned following a chessboard pattern. This state will have a zero total magnetization. Let us consider now the zero-temperature ferromagnetic case and the Glauber or Metropolis algorithm. Contrary to what is expected, this system will not order in a pure "black" or "white" ferromagnetic phase but it will continuously switch between the two complementary chessboard patterns. For eliminating the parallel update of the neighbors that causes such problems, and still taking advantage of the parallel nature of the computer, we impose an extra chessboard mask on the array. At each odd (even) step we update simultaneously the spins corresponding to the black (white) cells of the chessboard mask. For updating the chosen spins the Metropolis algorithm is then used. It is simple to realize that our method is equivalent to the classical serial Metropolis dynamics in which the spins are updated in a well-defined order. Detailed balance and ergodicity are thus naturally satisfied [27,26], so the obtained equilibrium statistics should be the right one.

Implementing the above scheme on the CNN-UM is realized as follows. In each step we first build three additional masks: the first marks the spins which have all 4 neighboring spins with similar orientation ($\Delta E = 8J$), the second one marks the spins which have 3 neighbors with similar orientation ($\Delta E = 4J$), and the third represents all the other spins for which $\Delta E \leq 0$. Separating these cells is relatively easy using logic operations and some special templates which can shift the images in different directions. We generate two random images with probability $\exp(-8J/kT)$ and $\exp(-4J/kT)$ and we perform an AND operation between the random image and the corresponding mask. After uniting the results of these two and the third mask ($\Delta E \leq 0$) we get a new mask which marks all spins which have to be flipped. Finally, we use the chessboard mask and allow only those spins to flip which correspond to black (white) pixels if the time-step is odd (even). The CNN code developed for studying this problem can be also downloaded from the home-page dedicated to this study [24]. It is worth mentioning that cluster algorithms, like the one proposed by Swendsen and Wang [28] or Wolf [29], seem to be also appropriate for the parallel architecture of the CNN-UM. As shown in the site-percolation problem clusters can be easily detected using figure reconstruction templates. This cluster detection can be effectively used to detect the cluster of correlated spins as described in [29].

Simulation results obtained with a Metropolis type algorithm are presented on Fig. 3. On this figure we compare results of (i) the classical Metropolis algorithm on a digital computer, (ii) the results of our parallel algorithm simulated on a digital computer and (iii) the results obtained on the ACE16K chip. By plotting the average magnetization, the specific heat and the susceptibility as a function of the temperature one can conclude that different results are in good agreement with each other. All simulations were performed on a $128 \times 128$ lattice using free boundary conditions.

Fig. 3(d) shows the time needed for 1 MC step as a function of the lattice size $L$. While on a PC type computer this scales as $L^2$, on the CNN-UM the time does not depend on the lattice size (each command is executed in a fully parallel manner on the whole lattice). The time measured on the ACE16K chip with $L = 128$ was 4.8 ms, while on a Pentium 4 PC working at 2.8 GHz under Linux operating system the time needed for 1 MC step was 2 ms. For this lattice size the simulations are still faster on the classical digital computers, however the difference is again caused by the low number of local binary memories, which results in time-consuming extra operations in our algorithms. Taking into account the trend that the size of the CNN chip (Table 1) and the number of local memories will increase in the near future these results are also promising.

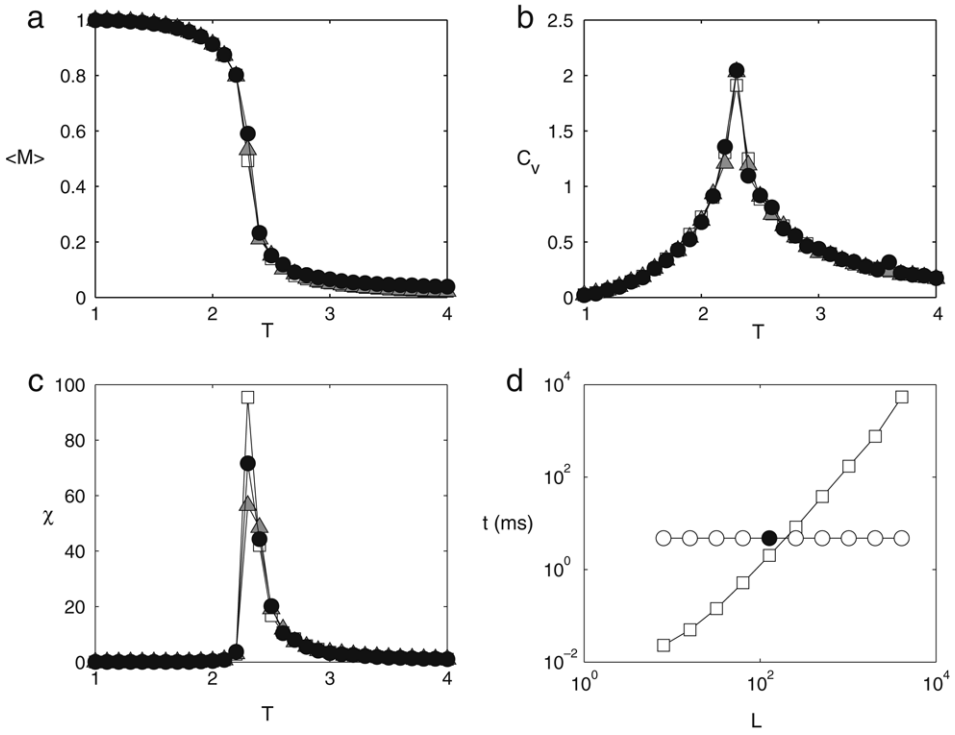


Fig. 3. Average magnetization $M$ (a), specific heat $C_v$ (b) and susceptibility $\chi$ (c) are plotted as a function of the temperature $T$ for the classical Metropolis algorithm on a digital computer (squares), our parallel algorithm simulated on a digital computer (triangles) and the algorithm simulated on the ACE16K CNN-UM chip (circles). Figure (d) compares the simulation time $t$ (in ms) needed for 1 MC step on a Pentium 4 PC with 2.8 GHz (squares) and the CNN-UM (circles) as a function of the lattice size $L$. The filled circle marks the simulation time obtained on the ACE16K chip ($L = 128$).

## 4. Studying spin-glasses on a locally variant CNN

For the CNN-UM's realized and used until now the templates $A, B, z$ are identical for all cells. This means that the strength of the connections between neighboring cells can be changed, but it varies simultaneously and identically for all cells. Here we will consider a locally variant CNN, on which these connection-parameters could be controlled separately for each cell. This would mean that instead of the nine parameters of matrix $A$, one would use nine images (arrays) describing the strength of the parameters. Of course this makes harder to control the system, but the number of possible templates and, thus, the possible applications become much larger.

A hardware like this would need more memories, more template registers and a more complicated control unit. The lack of such a hardware is probably caused by the lack of motivation and not that it is technically infeasible. Very few studies are dealing with such kind of CNN and very few applications have been developed until now.

Here we will define a locally variant CNN which can be used to study and optimize frustrated models like spin-glasses. As it will be shown this CNN is an analog correspondent of locally coupled spin-glasses. The only difference between the two systems is that for CNN the variables are continuous variables in the [-1,1] range and not discrete ones ($\pm 1$) like for the usual Ising-type spin models.

Let us now consider a CNN in which the $A$ parameters are locally defined: $A(i, j; k, l) \in [-1, 1]$, where $(i, j)$ and $(k, l)$ mark two neighbor cells. We consider symmetric connections: $A(i, j; k, l) = A(k, l; i, j)$ and $A(i, j; i, j) = 1$ for all $(i, j)$. The parameters $B$ which control the effect of the input image will be taken simply as: $B(i, j; i, j) = b$ and $B(i, j; k, l) = 0$. The parameter $z$ is not needed, so finally our template is defined by $\{A, b\}$. The state-equation of the system writes as:

$$\frac{\mathrm{d}x_{i,j}(t)}{\mathrm{d}t} = -x_{i,j}(t) + \sum_{<k,l>\in N(i,j)} A_{i,j;k,l}\, y_{k,l}(t) + b u_{i,j}, \quad (4)$$

where we emphasize again, that $x_{i,j}$ is the state value, $y_{i,j}$ is the output, and $u_{i,j}$ is the input of the cell $(i, j)$ with neighborhood $N(i, j)$ (8 neighbors and itself).

It has been shown by Chua et al. [6] that if the connection matrix is symmetric $A(i, j; k, l) = A(k, l; i, j)$ it is possible to define a Lyapunov function of the CNN which behaves like an energy function of the system. For the CNN defined above with self-connecting parameters $A(i, j; i, j) = 1$, this energy can be written as:

$$E(t) = -\sum_{\langle i,j;k,l\rangle} A(i, j; k, l) y(i, j) y(k, l) - b \sum_{i,j} y(i, j) u(i, j), \quad (5)$$

where $\langle i, j; k, l\rangle$ denotes the nearest-neighbor connections and each pair of neighbors $(i, j)$ and $(k, l)$ is taken only once into the sum. By choosing the parameter $b = 0$, the energy of this special CNN corresponds to the energy of a generalized Ising type system with locally varying coupling constants on a square lattice. The only difference is that spins are usually defined as

$\pm 1$ while here we have continuous values between [-1, 1]. Since the $A(i, j; k, l)$ coupling constants can be positive and negative as well locally coupled spin-glasses can be also mapped in such systems. In the following, we will be especially interested in the case when the $A(i, j; k, l)$ couplings leads to frustration and the system behaves as a spin-glass [30,31].

This Lyapunov function has two important properties (for details see [6]):

1. it is always a monotone decreasing function $\mathrm{d}E/\mathrm{d}t \leq 0$, so starting from an initial condition $E$ can only decrease during the dynamics of the CNN.

2. the final steady state is a local minimum of the energy: $\mathrm{d}E/\mathrm{d}t = 0$. In addition to these, our CNN has also another important property: due to the fact that all $A(i, j; i, j) = 1$ it can be shown that the output values of the cells in a final steady state will always be $\pm 1$.

Considering these properties we can conclude that the local minima of CNN and the generalized Ising-type spin model coincide. Starting from an initial condition the final steady state of the template — meaning the result of an operation — will be always a local minimum of the generalized Ising type spin model with connections defined in matrix $A$. The fact that one single operation is needed for finding a local minimum of the energy, gives us the opportunity for developing very fast optimization algorithms.

As already emphasized we will consider the complicated frustrated case (locally coupled spin-glass type model) where the $A$ coupling parameters can take both positive and negative values. We are searching for the minimum energy configuration of such models. Our algorithm is based on ideas similar to the well-known simulated annealing method. The noise is included with random input images, and the strength of an external magnetic field is governed by parameter $b$. Whenever $b$ is different from zero, our CNN template minimizes the energy with form (5): the first part of it being the energy of the pure Ising-type model and the second part is minimal when the state of the system is equal to the input image. For those familiar with Ising type models it is evident that the input image acts as an external magnetic field. If $b$ is a very large number, the result will be the input image itself, if $b = 0$ the result is a local minimum of the pure Ising-type system. For values in between, our result is a "compromise" between the two cases, so slowly decreasing $b$ will result in a similar process like in simulated annealing: first big fluctuations of the energy are allowed, but slowly we drive the system to a low energy state, even if we can not be sure that the global minimum was found.

The steps of the algorithm would thus be as follows:

1. we start from a random initial condition $x$, and $b = b_0$ (with this value the result of the template is almost exactly the same as the input image).

2. We generate a binary random input image $u$ with $1/2$ probability of black pixels,

3. the defined CNN template is performed using the $x$ initial state and the $u$ input image,

4. we decrease the value of $b$ with $\Delta b$,

5. we repeat steps 2–4, until $b = 0$, always using the results of the template as initial state in the next step.

6. When $b = 0$ the image is saved and the energy is calculated.

Usually in simulated annealing many steps at a single temperature are needed. Here the CNN template working in parallel replaces all these steps. We could choose to perform more templates at a given value of $b$, but the results would not improve. Instead, we choose to repeat this whole cooling process several times. As a result of these, different final states will be obtained, and one gets higher probability of finding the right global minima.

In our experiments spin-glass system with connections $A(i, j; k, l) = \pm 1$ were simulated. The $p$ probability of the positive connections can be varied (influencing the amount of frustration in the system) and we considered local interactions with the first 8 nearest neighbors. The initial value of $b$ was chosen $b_0 = 5$ (with this value the result of the template will be almost equivalent with the random input image). Choosing the value of $\Delta b$ is a more delicate problem, a proper value had to be found, which provides an acceptable compromise between the quality of the results and speed of the algorithm.

For testing the algorithm and measuring the number of steps needed to achieve this we always previously have to search for the exact global minima. In case of small systems with $L = 5, 6$ this search was quick and exact. For bigger systems the classical simulated annealing was used with decreasing rate of the temperature 0.99 and performing 1000 Monte Carlo steps at each temperature.

As shown on Fig. 4(a). in case of a lattice with size $L = 8$, the number of steps needed for finding the global optimum shows a minimum at a given value of $\Delta b$. In this simulation at each value of $\Delta b$ the number of steps needed is averaged over 4000 different systems - covering widely different densities of the + connections. As shown on Fig. 4(b) this optimal $\Delta b$ value also depends on the size of the lattice $L$. For bigger lattices we could not afford to do the simulations with the optimal parameter values since the time needed would increase too much ($\Delta b$ was too small). We have thus worked with a fixed $\Delta b = 0.05$ value. As an example at a $p = 0.4$ probability of the positive connections we have also plotted the number of steps needed for finding the optimal energy previously searched with simulated annealing. Results in this sense are presented on Fig. 5(a). As expected an exponential growth with the systems size is observable. The number of steps depends also on the $p$ probability of positive connections, as illustrated on Fig. 5(b) for a system with $L = 7$. 5000 different systems were analyzed at each value of $p$. It was found that the system is almost equally hard to solve for all $p$ values in the rage of $p \in (0, 0.6)$.

Now, some thoughts about the estimated speed of such an optimizing algorithm. Since there is no real CNN available with locally variant connections we were only simulating how a CNN would work. For this we had to solve a huge system of partial differential equations. This process gets very slow for bigger lattices and this is the reason why all of our results were obtained on rather small system sizes. A real locally variant CNN chip with its parallel architecture would solve one template in a very short time of the order

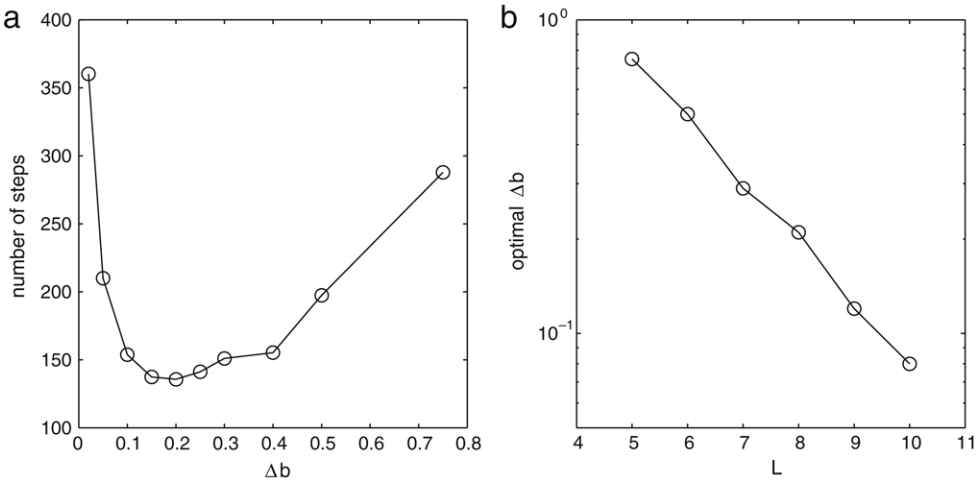


Fig. 4. (a) In case of a system with $8 \times 8$ cells, the number of steps needed to get the global minima is plotted as a function of $\Delta b$. 4000 different systems were considered covering the whole range of the possible $p$ values. (b) The optimal value of $\Delta b$ is shown in function of the lattice size $L$.

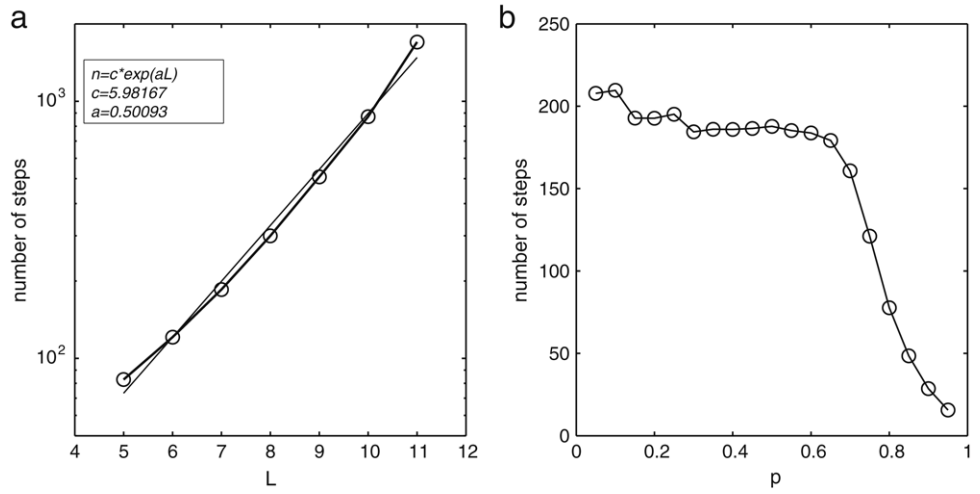


Fig. 5. (a) The number of steps needed to find the optimal energy as a function of the lattice size $L$. The density of positive connections is fixed to $p = 0.4$, and the parameter $\Delta b = 0.05$ is used. (b) For a system with size $L = 7$ the number of steps needed for getting the presumed global minima is plotted as a function of the probability of positive connections $p$.

of microseconds, independently from the lattice size. At each step of our algorithm we are also generating a binary random image. This process was already 5 time faster on the ACE16K chip than on a 2.8 GHz Pentium 4 digital computer and needed around 100 μs (see [18]). It is important, that we do not need to save information at each step, only at the end of each cooling process. Saving an image is in the range of 10 milliseconds on the ACE16K. Making a rough approximation, a chip with similar properties like the ACE16K and by using $\Delta b = 0.05$ should be able to solve between 1000–5000 steps in one second, *independently from the lattice size*.

Spin-glass like systems have many applications in which global minimum is not crucial to be exactly found. In most of the practical applications the minimization is needed only with a margin of error. In such cases the number of requested steps will decrease drastically. For example it has been shown that using spin-glass models as error-correcting codes, their cost-performance is excellent [32], and the systems usually are not even in the spin-glass phase. In this manner finding acceptable results could be very fast even on big lattices considering the parallel structure of the CNN.

## 5. Conclusions

Cellular wave computers [4,3] and, as a special case, the Cellular Neural Network Universal Machine [5] represent a new computation paradigm. There are inspired from some basic principles of our nervous system (specially the retina): (i) several thousands of cells (neurons) locally interconnected with each other and working totally in parallel; (ii) the states are described with analog values; and (iii) the evolution of the system is continuous in time. The system is mainly used for developing visual microprocessors [13,8,2]. In the present work we have shown that the CNN-UM chips can also be useful in studying complex problems of statistical physics. The natural noise of the chip can be effectively used in stochastic

simulations, and the parallel nature is very appropriate when simulating lattice models. Experimental results are promising. By increasing the lattice size of the chips and the number of local memories in the near future will definitely favor such kind of computational approaches. Quasi three-dimensional chips with several layers of cells are already appearing, introducing a new level of complexity and many new possibilities for applications. It is also believed that cellular wave computing is very probable to develop much further, going beyond the standard CNN model. As an example, a non-standard CNN was presented in Section 4, with locally variant coupling parameters. It was used with success for optimizing frustrated spin-glass type systems. It seems that CNN computers and, in general, cellular wave computing could soon become a useful complementary tool for making complex computer simulations on lattice models.

## Acknowledgments

The support of the Jedlik Laboratories of the P. Pázmány Catholic University is gratefully acknowledged.

## References

[1] J. von Neumann, Papers of John von Neumann on Computing and Computer Theory, MIT Press and Tomash Publ, Los Angeles, San Francisco, 1987.
[2] www.anafocus.com.
[3] T. Roska, J. Circuits Syst. Comput. 5 (2) (2003) 539.
[4] T. Roska, Electron. Lett. 43 (8) (2007) 427.
[5] T. Roska, L.O. Chua, IEEE Trans. Circuits Syst. II 40 (1993) 163.
[6] L.O. Chua, L. Yang, IEEE Trans. Circuits Syst. 35 (1988) 1257.
[7] K.R. Crounse, L.O. Chua, IEEE Trans. Circuits Syst. I: Fundamental Theory Appl. 43 (1996) 353.
[8] Á. Zarándy, C. Rekeczky, IEEE Circuits Syst. Mag. 5 (2005) 36.
[9] T. Roska, L.O. Chua, D. Wolf, T. Kozek, R. Tetzlaff, F. Puffer, IEEE Trans. Circuits Syst. I: Fundamental Theory Appl. 42 (1995) 807.
[10] T. Kozek, L.O. Chua, T. Roska, D. Wolf, R. Tetzlaff, F. Puffer, K. Lotz, IEEE Trans. Circuits Syst. I: Fundamental Theory Appl. 42 (1995) 816.
[11] J.M. Cruz, L.O. Chua, IEEE Trans. Circuits Syst. I: Fundamental Theory Appl. 42 (1995) 715.
[12] K.R. Crounse, T. Yang, L.O. Chua, Fourth IEEE International Workshop on Cellular Neural Networks and their Applications, Seville, Spain, 1996.
[13] A. Rodriguez-Vázquez, G. Linan Cembrano, et al., IEEE Trans. Circuits Syst. I 51 (2004) 851.
[14] L.O. Chua, T. Roska, Cellular Neural Networks and Visual Computing, Cambridge University Press, 2002.
[15] T. Kozek, T. Roska, Internat. J. Theory Appl. 24 (1996) 49.
[16] I. Petrás, T. Roska, L.O. Chua, IEEE Trans. Circuits Syst. I: Fundamental Theory Appl. 50 (2003) 619.
[17] K.R. Crounse, L.O. Chua, IEEE Trans. Circuits Syst. 42 (1995) 583.
[18] M. Ercsey-Ravasz, T. Roska, Z. Néda, Internat. J. Modern Phys. C 17 (6) (2006) 909.
[19] M.E. Yalcin, J. Vandewalle, P. Arena, A. Basile, L. Fortuna, Internat. J. Circuit Theory Appl. 32 (2004) 591.
[20] D. Stauffer, A. Aharony, Introduction to Percolation Theory, second ed., Taylor and Francis, London, 1992.
[21] M. Sahimi, Application of Percolation Theory, Taylor and Francis, London, 1994.
[22] D. Stauffer, Amer. J. Phys. 45 (10) (1977) 1001.
[23] I. Szatmári, P Földesy, C. Rekeczky, Á Zarándy, in: Proceedings of the CNNA-2002 Frankfurt, Germany, 2002.
[24] M. Ercsey-Ravasz, T. Roska, Z. Néda, http://www.phys.ubbcluj.ro/zneda/cnn.html.
[25] K. Malarz, S. Galam, Phys. Rev. E 71 (2005) 016125.
[26] J. Glauber, J. Math. Phys. 4 (1963) 194.
[27] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, J. Chem. Phys. 21 (1953) 1087.
[28] R.H. Swendsen, J.S. Wang, Phys. Rev. Lett. 58 (1987) 86.
[29] U. Wolff, Phys. Rev. Lett. 62 (1989) 361.
[30] S.F. Edwards, P.W. Anderson, J. Phys. F 5 (1975) 965.
[31] D. Sherrington, S. Kirkpatrick, Phys. Rev. Lett. 35 (1975) 1792.
[32] N. Sourlas, Nature 339 (1989) 693.