A neural network designed to solve the N-Queens Problem

Jacek Mańdziuk and Bohdan Macukow

Faculty of Mathematics and Information Science,
Warsaw University of Technology,
Plac Politechniki 1, 00-661 Warsaw, POLAND
e-mail: mandziuk@mini.pw.edu.pl
http://www.mini.pw.edu.pl/˜ mandziuk

Abstract

In this paper we discuss the Hopfield neural network designed to solve the N-Queens Problem (NQP). Our network exhibits good performance in escaping from local minima of energy surface of the problem. Only in approximately 1% of trials it settles in a false stable state (local minimum of energy). Extensive simulations indicate that the network is efficient and less sensitive to changes of its initial energy (potentials of neurons). Two strategies employed to achieve the solution and results of computer simulation are presented. Some theoretical remarks about convergence of the network are added.

Published in **Biological Cybernetics 66(4), 375-379, (1992)**

1 Introduction

Designing artificial neural networks is generally based on their similarity to the human brain. With rough simplifications an artificial neuron in the network can be compared to a biological one and connection weights between neurons in the network to synaptic connections in the human brain. So far not much is known about how exactly the human brain works and how many and which neurons among hundreds or thousands that a single neuron is connected with actually contribute to changes of its state (potential). Although we cannot copy or even follow the principles of the human brain work close enough, some approaches to constructions of artificial neural networks have been done. For a review of neural nets structures, architectures and applications see for example (Grossberg 1988, Lippman 1987).

Hopfield-type neural networks (Hopfield 1984) composed of highly-interconnected, analog elements (neurons) can be successfully used in solving optimization problems. Network structure and connection weights between neurons depend on specific constraints of a problem. A single neuron in the network simply sums up signals, multiplied by connection weights, that are sent to it from other neurons and then sends a respond signal back to them. A weighted sum of received signals U and the neuron response $V(U)$ are called input and output potentials, respectively. The function of response $V(U)$ very often is S-shaped. In our paper (see also Yao et al 1989)

$$V(U) = 1/2\ [1 + \tanh(\alpha U)] \tag{1}$$

where α is a gain of that sigmoid function. For sufficiently big values of α, $V(U)$ is of binary character, i.e. approximately

$$V(U) = \begin{cases} 0 & \text{if} \quad U\ <\ 0 \\ 1 & \text{if} \quad U\ >\ 0 \end{cases}$$

Let us consider a network composed of m neurons. Regardless of potentials of single neurons the function of energy E of the whole network can be defined (Hopfield 1984; Hopfield and Tank 1985) as

$$E = -1/2 \sum_{i=1}^{m} \sum_{j=1}^{m} T_{ij} V_i V_j \tag{2}$$

where T_{ij} $(i, j = 1, \ldots, m)$ is weight of a synaptic connection from the $j-th$ neuron to the $i-th$ one. All T_{ij} form a matrix of connection weights. T_{ij} can

be positive (excitatory stimulus) or negative (inhibitory stimulus) or equal to zero, i.e. there is no connection from neuron j to neuron i. The input potential U_i of the $i-th$ neuron is defined by the relation

$$U_i = -\partial E/\partial V_i \qquad (i = 1, \ldots, m) \tag{3}$$

and from (2) we obtain

$$U_i = \sum_{j=1}^{m} T_{ij} V_j \qquad (i = 1, \ldots, m) \tag{4}$$

Equations $(1) - (4)$ determine evolution of the network towards the state of a local minimum of energy, which is a stable state of the network (Hopfield and Tank 1985).

The neural network defined above can be used for solving optimization problems or, for example, as a CAM (Content Addressable Memory) e.g. (Hopfield 1984). When used for the first purpose the core of the problem is connected with a proper definition of the energy function E. It must be determined in such a way that its minima correspond to valid solutions of the given optimization problem. In our paper we design and apply the network to solve the NQP in a similar way as it was used to solve the Traveling Salesman Problem (TSP) (Hopfield and Tank 1985; Yao et al 1989). The energy function is modified according to the constraints of the NQP.

Application of neural networks to solving the NQP was also examined by Shackleford (1989).

Unlike in the TSP, where the task is not only to obtain a valid tour for the salesman, but also to minimize the length of that tour, in the NQP any obtained valid solution is automatically a satisfactory one, since all solutions are equally good.

Results of the computer simulation, presented further in this paper, show that the network designed to solve the NQP is very effective and has stable states mostly in solutions of the problem, i.e. almost every simulation test leads to a solution unless time restrictions are introduced to the process. Those results are comparable with ones obtained for the TSP in Hopfield model (Bizzarri 1991; Brandt et al 1988). Another very promising approach to solve optimization problems is based on Kohonen's self-organizing feature maps (Kohonen 1984). The implementation of Kohonen's idea for the TSP can be found in (Angéniol et al 1988; Fort 1988).

2 A neural network for the N-Queens Problem

In order to solve the NQP, for a given $n \in \mathcal{N}$, n queens must be placed on a square $n \times n$ chessboard in such a way that they don't attack one another. Two chess queens attack each other if they are placed in the same row or in the same column or on the same diagonal of the chessboard. It is obvious that in a solution of the NQP there must be exactly one queen placed in each row and exactly one in each column.

The network is represented by a square $n \times n$ matrix. Two different neurons (i, j) and (k, l) ($i, j, k, l \in \{1, \ldots, n\}$) are connected if and only if either
- neurons are in the same row ($i = k$),
- neurons are in the same column ($j = l$), or
- neurons belong to the same diagonal ($i + j = k + l$ or $i - j = k - l$),

and then the weight of a synaptic connection between them is negative. In all the other cases there is no synaptic connection between neurons (i, j) and (k, l), that is the weight of a connection between them is equal to 0. None of the neurons is connected to itself either. All connections are symmetric, i.e. the weight of a connection from neuron (i, j) to neuron (k, l) is equal to the one from (k, l) to (i, j).

In the resulting matrix V we use the treshold $S = 0.99$, i.e. if $V_{ij} > S$ it becomes 1 and if $V_{ij} < 1 - S$ it becomes 0, otherwise it doesn't change $(i, j = 1, \ldots, n)$. In termination (stable) state $V_{ij} = 1$ if and only if there is a queen on the field (i, j).
The energy function E is given by

$$\begin{aligned} 2E &= A \sum_{i=1}^{n} \sum_{j=1}^{n} \left[\left(\sum_{\substack{k=1 \\ k \neq j}}^{n} V_{ik} \right) V_{ij} + \left(\sum_{\substack{k=1 \\ k \neq i}}^{n} V_{kj} \right) V_{ij} \right] + \\ &\quad + B \sum_{i=2}^{n} \sum_{j=1}^{i-1} \left[\left(\sum_{\substack{k=i-j+1 \\ k \neq i}}^{n} V_{k,k-i+j} \right) V_{ij} \right] + \end{aligned}$$

$$+B\sum_{i=1}^{n}\sum_{j=i}^{n}\left[\left(\sum_{\substack{k=1\\k\neq i}}^{n+i-j}V_{k,k-i+j}\right)V_{ij}\right]+ \tag{5}$$

$$+B\sum_{i=1}^{n}\sum_{j=n-i+1}^{n}\left[\left(\sum_{\substack{k=i+j-n\\k\neq i}}^{n}V_{k,i+j-k}\right)V_{ij}\right]+$$

$$+B\sum_{i=1}^{n-1}\sum_{j=1}^{n-i}\left[\left(\sum_{\substack{k=1\\k\neq i}}^{i+j-1}V_{k,i+j-k}\right)V_{ij}\right]+$$

$$+C\left(\sum_{i=1}^{n}\sum_{j=1}^{n}V_{ij}-n_{-})\right)^{2},$$

where A,B,C,n_- are positive constants. According to (3) we have

$$-U_{ij}=A\Big(\sum_{\substack{k=1\\k\neq j}}^{n}V_{ik}+\sum_{\substack{k=1\\k\neq i}}^{n}V_{kj}\Big)+R_{ij}+S_{ij}+C\Big(\sum_{k=1}^{n}\sum_{l=1}^{n}V_{kl}-n_{-}\Big), \tag{6}$$

where

$$R_{ij}=\begin{cases}B\displaystyle\sum_{\substack{k=i-j+1\\k\neq i}}^{n}V_{k,k-i+j} & \text{if} \quad i-j>0\\[2ex] B\displaystyle\sum_{\substack{k=1\\k\neq i}}^{n+i-j}V_{k,k-i+j} & \text{if} \quad i-j\leq 0\end{cases}$$

and

$$S_{ij}=\begin{cases}B\displaystyle\sum_{\substack{k=i+j-n\\k\neq i}}^{n}V_{k,i+j-k} & \text{if} \quad i+j>n\\[2ex] B\displaystyle\sum_{\substack{k=1\\k\neq i}}^{i+j-1}V_{k,i+j-k} & \text{if} \quad i+j\leq n\end{cases}$$

$(i,j=1,\ldots,n)$.

We used symbols R_{ij} and S_{ij} to improve the notation, since the equations that characterize diagonals in each of the four triangle matrices in a square

matrix differ from one another. Those triangle matrices are defined along two main diagonals : one from $(1,1)$ to (n,n) and the other one from $(n,1)$ to $(1,n)$. Let us, for example, consider the triangle matrix lying below the main diagonal from $(1,1)$ to (n,n). For every diagonal of that matrix, field (i,j) belongs to it if and only if $n > i-j > 0$, where $i-j$ is equal to the distance between the main diagonal and the diagonal that field (i,j) belongs to. Three other cases were examined in an analogous way.

Those of the sums in (5) and in (6) which are multiplied by A are responsible for interactions between neurons in rows and in columns. Those terms fulfil a constraint that there must be at most one queen placed in each row and at most one in each column. Sums multiplied by B represent interactions on diagonals and mean that there must be at most one queen placed on each diagonal. The component multiplied by C forces the sum of outputs of all neurons in the network to be close to a constant n_-, such that $n_- - n \in (0,1)$, according to the condition that the number of queens placed on the $n \times n$ chessboard must be equal to n. In the last section we briefly discuss the case $n_- = n$ for which, in a state of n queens, the above component disappears. The simulation process is described as follows (see Fig. 1) :

(i) all initial outputs V_{ij}^0 $(i,j = 1,\ldots,n)$ are arbitrarily set and from (5) starting value of energy E^0 is obtained,

(ii) neuron (i,j) is chosen at random and from (6) the corresponding value U_{ij} is calculated according to outputs of other neurons, and then from (1) V_{ij} is obtained.

Operation (ii) is repeated $5 \cdot n^2$ times. We shall call every n^2 repetitions of (ii) an internal iteration. Five internal iterations compose one external iteration, and then a new value of E from (5) is counted. The simulation process terminates if after 5 succesive external iterations the energy remains constant, or if the number of external iterations exceedes 25 and the network still does not achieve a stable state, i.e. a constant value of E.

The number $5 \cdot n^2$ repetitions of the point (ii) is a result of some preliminary tests, which showed it is sufficient for the process. Termination criteria of the simulation were established arbitrarily.

3 Results of computer simulations

Output potentials V_{ij}^0 $(i,j = 1,\ldots,n)$ are initial parameters of the simulation. We have checked four strategies for setting initial outputs denoted by a,b,c,d. In those strategies output of each of n^2 neurons was initially set to:
a - random value from $[0,\beta]$,
b - random value from $[0,1]$,
c - random value from $[1-\beta,1]$,
d - random value from $[0,\beta]+1/n$,
where $\beta \approx 0.03$.

In the first case all neurons have low output potentials. In the second case there are no restrictions to neuron output potentials. In the third one they have high outputs. In the last case initial outputs are close to $1/n$.

We also used two strategies to simulate internal iterations : full and part (denoted by F and P, respectively). In case F, in an internal iteration a neuron to be modified was chosen in a completely random way. In case P, in every internal iteration we randomly chose a permutation of the numbers $1,\ldots,n^2$, and then neurons were modified according to that permutation. In both cases five internal iterations composed an external one.

In our simulation process constants were set as follows :
$n = 8, \qquad A = B = C = 100, \qquad \alpha = 15, \qquad n_- = 8.25,\ 8.50,\ 8.75.$

For each of the eight strategies $(a,F),(a,P),\ldots,(d,P)$ and for each of the three values of n_- we simulated 150 tests. The results are presented in Table 1. A closer example of the simulation of one test is presented in Fig. 2.

Since in our model $A = B = C = 100$ we could have actually had one constant, say D, equal to 100 instead of A,B,C, or we could even have set $\alpha = 1500$ and omitted constants A,B,C at all. We haven't done it for three reasons. First we wanted our network to be similar to and comparable with the Hopfield's network used for solving the TSP. Moreover in our preliminary simulations constants A,B,C were set in various ways and the condition $A = B = C$ appeared just as a result of that simulations. At last since the set of constants we finally established is not neccesserily the best one, we thought that the model with four independent constants A,B,C,α would be more clear and more improvable than the one with constant α only.

The network described above had no problems with solving the NQP. Generally, better results were obtained for strategy P than for strategy F. In case P strategies a and d seemed to predominate slightly strategies b and

c, especially for $n_- = 8.75$. In case F strategy c worked a little better than the other ones.

Taking into account only those trials in which a solution of the problem was obtained we can say that the network settled in a stable state faster in strategy P than in F. In strategy P the network needed only a few, mostly between 1 and 8, external iterations to achieve the minimum of energy, and only occasionally it needed more than 15 iterations (average was about 5.4). In strategy F the average number of external iterations needed to obtain a solution was about 7.2. Moreover, there were more tests in that case in which over 15 iterations were done before the network settled in a stable state than in case P.
The greatest number of different solutions was obtained by strategy (c, P).

Equations $(1) - (4)$, together with a binary character of $V(U)$ (for big values of α) guarantee that if there hadn't been constraints for the number of external iterations in the process and for the number of repetitions of a constant value of energy, for each strategy the network would almost every time have eventually settled in a state corresponding to a solution of the NQP (with data A, B, C, α, n_- set as above).
We shall present the outline of the proof of this statement.

After some iterations output potentials of all neurons in the network would be equal to either 0 or 1, due to big values of A, B, C, α.

Let us denote by k_{ij} $(i, j = 1, \ldots, n)$ the number of neurons that have output potential equal to 1 and interact with a neuron (i, j), and by m the number of all neurons in the network that have output potential equal to 1. Let also δ be equal to $n_- - n$, and denote $D \stackrel{\text{def}}{=} A = B = C$. Then

$$U_{ij} = -D(\ k_{ij} + m - n - \delta\ ),$$

where $\delta = 0.25$ or $\delta = 0.50$ or $\delta = 0.75$, and

$$V_{ij} = \begin{cases} 0 & \text{if} \quad k_{ij} > n - m \\ 1 & \text{if} \quad k_{ij} \leq n - m \end{cases} \tag{7}$$

If $m > n$ then $V_{ij} = 0$, no matter to k_{ij}. That is for $m > n$ the network would reset (set to 0) output potentials of all neurons chosen, until $m \leq n$. If $m < n$ then in most cases there exists a neuron (i, j) such that $k_{ij} \leq 1$ and $V_{ij} = 0$. That neuron would be set (set to 1) when chosen. In many cases there also exist neurons that have output potential equal to 1 and are

"attacked" more than $n - m$ times. Reseting output potential of any such neuron the network would also have changed its configuration.
Finally, if $m = n$ and the state of the network is not a solution of the NQP then there must exist a neuron (i, j) such that $V_{ij} = 1$ and $k_{ij} > 0$. That neuron would be reset while chosen.
Thus eventually, because of a random character of the way of chosing neurons, the network with great probability would reach a state corresponding to a solution of the NQP.

From (7) it is obvious that the network would not change output potential of any neuron in such a state ($m = n,\ k_{ij} = 0$ for $V_{ij} = 1$ and $m = n,\ k_{ij} \geq 2$ for $V_{ij} = 0$).
The above considerations imply that almost all stable states of the network are those corresponding to solutions of the problem.

Our further simulations have confirmed this fact. When we left out the constraint for the number of external iterations and changed the other constraint from 5 to 30, giving the network "more chances" to leave any state that is not stable, a solution was obtained in 99% tests, however, a few times even more than 40 iterations (not including 30 made in a stable state) were required. Results of the simulation made with the above weaker constraints are presented in Table 2. For $n_- = 8.25$ and $n_- = 8.75$ the network converged to a solution in all tests. Worse results were obtained for $n_- = 8.50$, especially for strategy F. Generally, strategy P was about three iterations faster than strategy F.

4 Final remarks

Unfortunately there exist some configurations of less than n queens that form a false stable state, i.e. a stable state which is not a solution of the problem. Our simulations showed that the percentage of those false stable states is negligible. For example a state of seven queens $(3, 1, 4, 7, \ , 2, 5, 8)$ (the first queen in the third column, the second queen in the first column,..., the fifth one left, ..., the eight queen in the eight column), for $n = 8$ is unchangable.

It is important that $\delta < 1$. For $n_- = 9$ only approximately 40% tests led to a solution.

Simulation may also be performed successfully for $n_- = 8$, but in that case a slightly different interpretation of the resulting matrix V must be

applied. Since $\delta = 0$ then in stable states there would appear values of $V_{ij} = 1/2$. If we interpret any neuron that has output potential equal to $1/2$ also as a field on which a queen is placed the results of the simulation would be similar to and equally good as those for $\delta = 0.25, 0.50, 0.75$. In this case similar, but slightly more complicated, considerations about the behaviour of the network can be applied, as well.

References

[1] Angéniol B., De La Croix Vaubois G., Le Texier J.Y., (1988), **Self-organizing feature maps and the Travelling Salesman Problem**, *Neural Networks*, 1:289-293,

[2] Bizzarri A.R., (1991), **Convergence properties of a modified Hopfield-Tank model**, *Biological Cybernetics*, 64:293-300,

[3] Brandt R.D., Wang Y., Laub A.J., Mitra S.K., (1988), **Alternative networks for solving the Traveling Salesman Problem and the List-Matching Problem**, *Proc. Int. Conf. on Neural Networks*, 333-340,

[4] Fort J.C., (1988), **Solving a combinatorial problem via self-organizing process: an application of the Kohonen algorithm to the Traveling Salesman Problem**, *Biological Cybernetics*, 59:33-40,

[5] Grossberg S., (1988), **Nonlinear neural networks: principles, mechanisms, and architectures**, *Neural networks*, 1:17-61,

[6] Hopfield J.J., (1984), **Neurons with graded response have collective computational properties like those of two-state neurons**, *Proc. Natl. Acad. Sci. USA*, 81:3088-3092,

[7] Hopfield J.J., Tank D.W., (1985), **"Neural" computation of decisions in optimization problems**, *Biological Cybernetics*, 52:141-152,

[8] Kohonen T., (1984), **Self-organization and associative memory**, *Springer-Verlag, Berlin*,

[9] Lippman R.P., (1987), **An introduction to computing with neural nets**, *IEEE ASSP Magazine*, April:4-22,

[10] Shackleford J.B., (1989), **Neural data structures: programming with neurons**, *Hewlett-Packard Journal*, June:69-78,

[11] Yao K.C., Chavel P., Meyrueis P., (1989), **Perspective of a neural optical solution to the Traveling Salesman optimization problem**, *SPIE, Optical Pattern Recognition ll*, 1134:17-25.

Iteration 0
$E = 56580.948279$

.7	.8	.8	.8	.7	.3	.1	.5
.4	.2	.7	.9	.2	.5	1.	.1
.2	.7	1.	.7	.3	.2	.9	.3
.9	.9	.8	.8	.8	.6	.4	.8
.2	.1	.0	.6	.6	.1	.8	.1
.9	.9	.2	.3	.6	.3	.9	.2
.7	.6	1.	.9	.6	.6	.1	.4
.7	.1	.8	.9	.6	.4	.2	.8

Iteration 1
$E = 203.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
1	0	0	0	0	0	0	0
0	0	0	0	**1**	0	0	0
0	0	0	0	0	0	**1**	0
0	0	0	0	0	0	0	0
0	**1**	**1**	0	0	0	0	0
0	0	0	0	0	0	0	**1**

Iteration 2
$E = 203.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
1	0	0	0	0	0	0	**1**
0	0	**1**	0	**1**	0	0	0
0	0	0	0	0	0	**1**	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	**1**	0	0	0	0	0	0

Iteration 3
$E = 78.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**
0	0	0	0	**1**	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	**1**	0	0	0	0	0	0

Iteration 4
$E = 78.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	0	0	0	**1**	0	0
0	**1**	0	0	0	0	0	0

Iteration 5
$E = 103.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	0
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	**1**
0	0	0	0	**1**	0	0	0

Iteration 6
$E = 103.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	0
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	**1**
0	0	0	0	**1**	0	0	0

Iteration 7
$E = 3.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0

Iteration 8
$E = 3.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0

Iteration 9
$E = 3.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0

Iteration 10
$E = 3.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0

Iteration 11
$E = 3.125000$

0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
1	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0

Strategy (b,F) $n_- = 8.25$
Stable state achieved at iteration 7
Energy : 3.125000 Obtained solution : (4, 6, 8, 2, 7, 1, 3, 5)

Figure 2: An example of the simulation in one test. Initial output potentials - random values from $[0, 1]$ are, for the sake of clarity of the figure, rounded to the nearest one tenth.

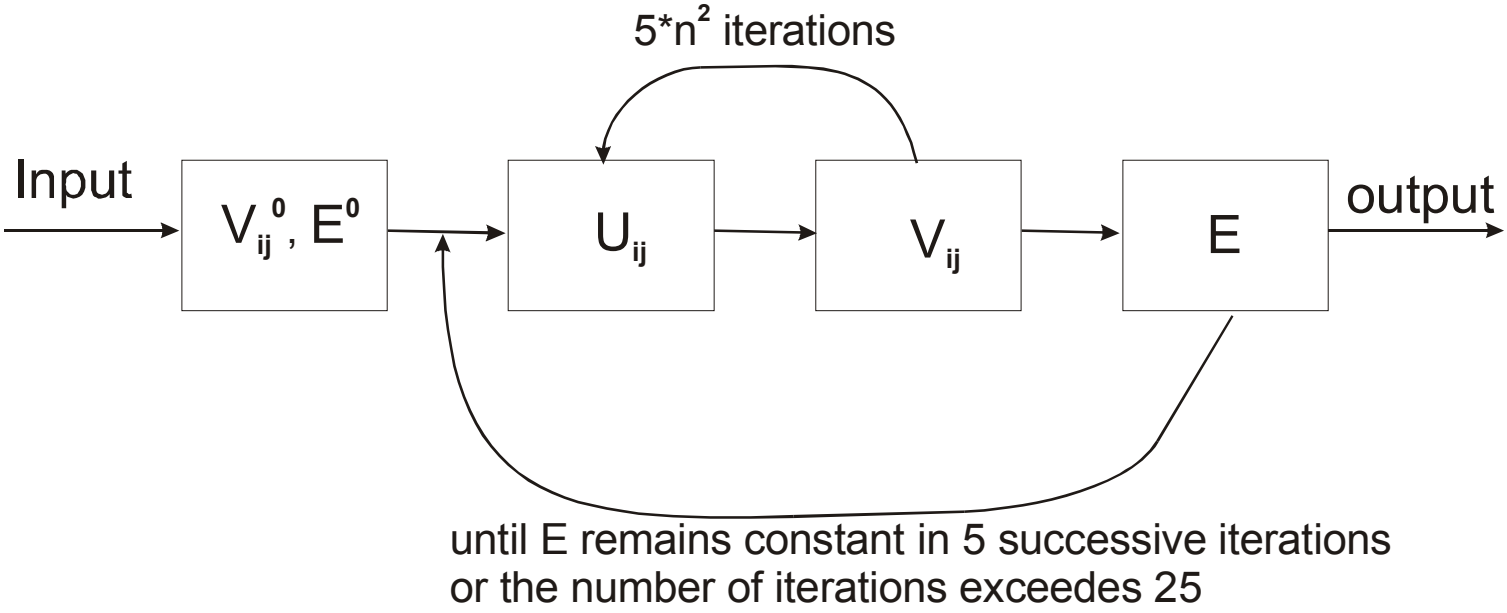

Figure 1: A diagram of dynamical evolution of the network.

	$n_ = 8.25$		$n_ = 8.50$		$n_ = 8.75$	
(a, F)	**133**	**64**	**128**	**66**	**137**	**55**
(b, F)	**132**	**69**	**134**	**75**	**128**	**71**
(c, F)	**136**	**73**	**135**	**67**	**135**	**68**
(d, F)	**132**	**67**	**133**	**68**	**127**	**68**
(a, P)	**146**	**76**	**145**	**69**	**148**	**69**
(b, P)	**144**	**74**	**136**	**70**	**140**	**73**
(c, P)	**143**	**75**	**142**	**77**	**141**	**77**
(d, P)	**145**	**72**	**144**	**74**	**146**	**67**

Table 1: Results of the computer simulation in all tested strategies. In each field of the table a value on the left is equal to the number of solutions of the NQP, and the one on the right denotes the number of different solutions, both obtained in 150 tests. The number of different solutions of the NQP for n=8 is equal to 92.

	$n_ = 8.25$		$n_ = 8.50$		$n_ = 8.75$	
(a, F)	**50**	**9.49**	**49**	**8.14**	**50**	**10.22**
(b, F)	**50**	**9.16**	**50**	**9.56**	**50**	**12.72**
(c, F)	**50**	**8.34**	**46**	**8.26**	**50**	**8.80**
(d, F)	**50**	**10.84**	**48**	**9.00**	**50**	**11.02**
(a, P)	**50**	**6.36**	**49**	**7.02**	**50**	**7.32**
(b, P)	**50**	**5.98**	**50**	**5.80**	**50**	**8.04**
(c, P)	**50**	**6.58**	**50**	**5.92**	**50**	**6.16**
(d, P)	**50**	**5.14**	**49**	**5.51**	**50**	**6.10**

Table 2: Results of the simulation with weaker constraints for the number of repetitions of a constant value of energy and for the global number of iterations. For each strategy 50 tests were made. The left value is equal to the number of obtained solutions and the right one represents the average number of external iterations required to achieve a solution, excluding iterations made in a stable state, i.e. 30 in each test. The average is counted only for tests that led to a solution.