Solving the N-Queens Problem with a binary Hopfield-type network. Synchronous and asynchronous model

Jacek Mańdziuk
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

The application of a discrete Hopfield-type neural network to solving the NP-Hard optimization problem - the N-Queens Problem (NQP) is presented. The applied network is binary, and at every moment each neuron potential is equal to either 0 or 1. The network can be implemented in the asynchronous mode as well as in the synchronous one with n parallelly running processors. In both cases the convergence rate is up to 100% and the experimental estimate of the average computational complexity is polynomial. Based on the computer simulation results and the theoretical analysis the proper network parameters are established. The behaviour of the network is explained.

Key Words : neural networks, optimization problems, discrete Hopfield network, the N-Queens Problem, NP-Hard

Published in **Biological Cybernetics 72(5), 439-446, (1995)**

1 Introduction

In the early eighties Hopfield (1982) introduced a neural network model based on the idea similar to the one used in the spin glass theory in quantum mechanics. In the next papers (Hopfield 1984, Hopfield and Tank 1985) the authors presented applications of that model to the Pattern Recognition Problem and to solving the Travelling Salesman Problem (TSP).

Later on, the quality of the results as well as the convergence rate reported by Hopfield and Tank were totally questioned by Wilson and Pawley (1988) and, in some aspects, by Kamgar-Parsi and Kamgar-Parsi (1990).

On the other hand, since the "classical" papers by Hopfield and Tank, various modifications to the original model were introduced, e.g. (Brandt et al. 1988; Bizzarri 1991) with very good results.

Most of the papers devoted to solving optimization problems by the use of neural networks dealt with the TSP which became a "standard problem" for that researches.

In this paper we also state some remarks concerning the TSP, but our main interest is bound with the N-Queens Problem (NQP). Due to its conceptual simplicity, the NQP is adequate to fully illustrate the network mechanism.

In the NQP one is to place n chess queens on a square chessboard composed of n rows and n columns, in such a way that they don't attack one another. In other words, the configuration of queens must fulfil the constraint that any two different queens are placed in the different rows, columns and diagonals. One of the possible configurations for $n = 8$ is presented in Fig. 1.

The problem is NP-Hard and the "conventional" way of solving it is based on extensive search methods.

The network proposed in this paper differs from the original Hopfield network mainly in the two following aspects:
(i) the starting point of the network can be chosen in a completely random way,
(ii) neuron potential at time $t + 1$ does not directly depend on its value at time t.

The point (**i**) as well as the complete binarization of the network are considered to be its biggest advantages. The completely discrete network

is much simpler than the continuous one both conceptually and as to the hardware realization.

The possibility of starting the network from any random point with no negative influence on the convergence rate and quality of the results shows that the network is indeed reliable.

The model proposed in this paper is a modification of the continuous Hopfield-type neural network that was previously used to solve the NQP (Mańdziuk and Macukow 1992). Since the general idea of the network is described in the above cited paper we only very briefly remind here the concept of the Hopfield network and present our modifications.

In the Hopfield network composed of m neurons the energy function is of the form

$$E = -1/2 \sum_{i=1}^{m} \sum_{j=1}^{m} t_{ij} v_i v_j \tag{1}$$

where t_{ij} is weigth of connection from the output of the $j-th$ neuron to the input of the $i-th$ one and v_i is the output potential of the $i-th$ neuron (defined below). For each neuron in the network the so-called input and output potentials can be defined, denoted by u and v, respectively. The input potential of the $i-th$ neuron is described by (2)

$$u_i = -\partial E / \partial v_i \qquad (i = 1, \ldots, m) \tag{2}$$

From (1) and (2) we obtain

$$u_i = \sum_{j=1}^{m} t_{ij} v_j \qquad (i = 1, \ldots, m) \tag{3}$$

Usually, the output potential is a modified hyperbolic tangent function, e.g.

$$v(u) = 1/2 \, [1 + \tanh(\alpha u)] \tag{4}$$

The network can work either *synchronously* or *asynchronously*. In the synchronous mode all the neurons (or only some of them) are updated simultaneously. In the latter case neurons are updated consecutively in either a completely random order or in the "random sequential order" (Yao et. al. 1989, Mańdziuk and Macukow 1992).

2 The N-Queens Problem

2.1 Network definition

The network is represented by a square boolean matrix $V_{n\times n}$. For the sake of simplicity, henceforth we shall use the same symbol v_{ij} for both the neuron in the network and the element of the matrix V. The energy function E is given by

$$\begin{aligned} E \;=\; & \frac{1}{2}\Bigg\{ A\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\left(\sum_{\substack{k=1\\k\neq j}}^{n} v_{ik}\right) v_{ij} \;+\; \left(\sum_{\substack{k=1\\k\neq i}}^{n} v_{kj}\right) v_{ij}\right] \;+ \\ & + B\sum_{i=2}^{n}\sum_{j=1}^{i-1}\left[\left(\sum_{\substack{k=i-j+1\\k\neq i}}^{n} v_{k,k-i+j}\right) v_{ij}\right] \;+ \\ & + B\sum_{i=1}^{n}\sum_{j=i}^{n}\left[\left(\sum_{\substack{k=1\\k\neq i}}^{n+i-j} v_{k,k-i+j}\right) v_{ij}\right] \;+ \\ & + B\sum_{i=1}^{n}\sum_{j=n-i+1}^{n}\left[\left(\sum_{\substack{k=i+j-n\\k\neq i}}^{n} v_{k,i+j-k}\right) v_{ij}\right] \;+ \\ & + B\sum_{i=1}^{n-1}\sum_{j=1}^{n-i}\left[\left(\sum_{\substack{k=1\\k\neq i}}^{i+j-1} v_{k,i+j-k}\right) v_{ij}\right] \;+ \\ & + \; C\left(\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij} - (n+\sigma)\right)^2 \Bigg\}, \end{aligned} \tag{5}$$

where A, B, C, σ are positive constants.
According to (2) we have

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

where

$$R_{ij} = \begin{cases} B \sum\limits_{\substack{k=i-j+1 \\ k \neq i}}^{n} v_{k,k-i+j} & \text{if} \quad i-j > 0 \\ B \sum\limits_{\substack{k=1 \\ k \neq i}}^{n+i-j} v_{k,k-i+j} & \text{if} \quad i-j \leq 0 \end{cases}$$

and

$$S_{ij} = \begin{cases} B \sum\limits_{\substack{k=i+j-n \\ k \neq i}}^{n} v_{k,i+j-k} & \text{if} \quad i+j > n \\ B \sum\limits_{\substack{k=1 \\ k \neq i}}^{i+j-1} v_{k,i+j-k} & \text{if} \quad i+j \leq n \end{cases}$$

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

Since the detailed explanation of the choice of the function (5) is given in Mańdziuk and Macukow (1992) let us only state two remarks:
(i) the energy function is chosen in such a way that its global minima correspond to solutions of the problem,
(ii) in (5) and (6) terms multiplied by A correspond to interactions in rows and columns - in each row and in each column at most one chess queen may be placed, terms multiplied by B correspond to the same constraint on diagonals, and finally, those multiplied by C correspond to the global network inhibition for the number of queens on the chessboard.

For evaluation of the neuron output potential, instead of a hyperbolic tangent we used the following hard-limiter threshold:

$$v(u) = \begin{cases} 0 & \text{if} \quad u < 0 \\ 1 & \text{if} \quad u \geq 0 \end{cases} \tag{7}$$

In the simulation tests three ways of setting the initial output potentials were used, denoted by $(a), (b)$ and (c). The output of each of n^2 neurons was initially set to (cf. Mańdziuk 1993):

(a) - 0,
(b) - random value from the set $\{0, 1\}$,
(c) - 1.

In the case (a) none of the neurons is excitated. In the case (b) some randomly chosen neurons are set (set to one), and the rest is reset (set to zero). In the last case all the neurons are set (the starting energy is very high in that case).

2.2 Asynchronous mode

In the asynchronous mode a single computer simulation test was composed of the following steps:

(**i**) all initial output potentials v_{ij} $(i, j = 1, \ldots, n)$ were set and from (5) the starting value of energy E was evaluated,

(**ii**) neuron $(i, j)(i, j \in \{1, \ldots, n\})$ was chosen at random and from (6) and (7) the input and output potentials were calculated,

(**iii**) operation (**ii**) was repeated $5n^2$ times, and then a new value of E from (5) was calculated.

Every n^2 repetitions of (**ii**) was called an *internal iteration*. Five internal iterations composed one *external iteration*. The simulation process terminated if the energy remained constant in the *a priori* established number of the successive external iterations or, the number of external iterations exceeded the constraint for the global number of iterations and the network still did not achieve a stable state, i.e. a constant value of E.

The network was tested with the folowing values of n:

$$n = 4,\ 8,\ 16,\ 32,\ 48,\ 64 \text{ and } 80 \tag{8}$$

Preliminary simulations together with the theoretical analysis of the network led to the following values of the network parameters:

$$A = B = C = 100,\ \sigma = 0 \tag{9}$$

The constraints for the global number of external iterations (equal to 1000) and for the number of repetitions of the same energy value (equal to 50 for $n \leq 32$ and 100 otherwise) were chosen big enough not to stop the test unless it really settles in the stable state.

2.2.1 Simulation results

For each n from (8), 100 simulation tests were done. The diagram of the average external iterations required by a single test for each initial strategy is depicted in Fig. 2.

According to the results we can state that:
(**i**) the convergence rate is very high - equal to 100% in all cases, except for $n = 8, (a)$ - which is 99%,
(**ii**) there is no significant differences in speed among strategies (a), (b), (c),
(**iii**) the average number of external iterations grows with the problem size

In the following sections the experimental results will be compared with the theoretical analysis of the network behaviour and the experimental computational complexity of the method will be estimated (sections 2.2.2 and 2.2.3).

2.2.2 Network analysis

In order to find the stable states of the network let us denote as $k_{rs}(t)$ the number of queens that attack the field (r, s) at time t, $(r, s \in \{1, ..., n\})$ and by $m(t)$ the number of all queens placed on the chessboard at time t.

Let $\Pi : \mathcal{N}_0 \to \{1, \ldots, n\} \times \{1, \ldots, n\}$ - be the *selecting function*, i.e.

$$\Big(\Pi(t) = (r, s)\Big) \Longleftrightarrow \Big(\text{at time } t \text{ the neuron } (r, s) \text{ was chosen}\Big)$$

and let $f(t)$, be the *function of transition* between the states at time t, i.e.

$$f(t) : \{0, 1\}^{n^2} \times \{1, \ldots, n\} \times \{1, \ldots, n\} \to \{0, 1\}^{n^2}$$

Now, the consecutive steps of the simulation test can be defined as below:

$$V(t + 1) = f(V(t), \Pi(t))$$

where $V(t)$ is the matrix of the output potentials at time t.

Based on the above formalism we can define the *stable state* and the *solution of the problem*:

Definition 1 *(Stable state)*
The state $V(t)$ of the network is called a *stable state* at time t if and only if,

$$\forall (r,s) \in \{1,\ldots,n\}\times\{1,\ldots,n\} \qquad \Big(\Pi(t) = (r,s)\Big) \Longrightarrow \Big(f(V(t),\Pi(t)) = V(t)\Big)$$

Definition 2 *(Solution of the problem)*
Any stable state $V(t)$ at time t that fulfils the following conditions (**i**)-(**ii**) is called the *solution of the problem* (simply: *solution*).
(α) $m(t) = n$
(β) $\forall (r,s) \qquad \Big(v_{rs}(t) = 1\Big) \Longrightarrow \Big(k_{rs}(t) = 0\Big)$
Finally, we define the two other categories of the network states:

Definition 3 *(False stable state)*
Any stable state which is not a solution is called a *false stable state*.

Definition 4 *(Quasi-stable state)*
Any state $V(t)$ of the network from which it is possible to go in one step to another state without changing the energy value is called a *quasi-stable state*, i.e.

$$\exists\ \Pi(t) \quad \Big(V(t+1) \neq V(t) \ \text{ and } \ E(t+1) = E(t)\Big)$$

As previously stated the parameters (9) were chosen after some initial simulations and network analysis. In fact, the only condition that was established firmly was $A = B \stackrel{\text{def}}{=} D$, which meant that attacks along rows or columns were "of the same importance" as those along diagonals.

Now we can rewrite (5) and (6) in the form

$$2E(t) = D\sum_{i=1}^{n}\sum_{j=1}^{n} k_{ij}(t)v_{ij}(t) \ + \ C(m(t) - (n+\sigma))^2 \tag{10}$$

and

$$-u_{ij}(t) = Dk_{ij}(t) + C(m(t) - (n+\sigma)) \tag{11}$$

Let us consider the network at time $t > 0$, and assume that the energy is equal to $E(t)$, and $\Pi(t) = (r,s)$. Then from (7) and (11)

$$\begin{cases} k_{rs}(t) > C/D(n+\sigma-m(t)) & \text{if} \quad v_{rs}(t+1) = 0 \\ k_{rs}(t) \leq C/D(n+\sigma-m(t)) & \text{if} \quad v_{rs}(t+1) = 1 \end{cases} \tag{12}$$

The change of the energy $\Delta E \stackrel{\text{def}}{=} E(t+1) - E(t)$ is from (10) equal to

$$\Delta E = \begin{cases} C(n - m(t) + \sigma + 1/2) - Dk_{rs}(t) & \text{if} \quad v_{rs}(t+1) - v_{rs}(t) = -1 \\ 0 & \text{if} \quad v_{rs}(t+1) = v_{rs}(t) \\ C(m(t) - n + 1/2 - \sigma) + Dk_{rs}(t) & \text{if} \quad v_{rs}(t+1) - v_{rs}(t) = 1 \end{cases} \tag{13}$$

Generally speaking the quality of the results (the percentage of the tests that led to solutions) essentially depended on the proportion $\frac{C}{D}$ and the parameter σ, for all n tested. The equations presented below can be developed in the general form, i.e. with the parameters C, D and σ. For the sake of clarity we present them in the simpler form based on the conditions

$$C = D \qquad \text{and} \qquad \sigma = 0 \tag{14}$$

Let us assume that at time t, $m(t) = n - f, f \in \mathcal{Z}$ and $\Pi(t) = (r, s)$. Then eqs. (12) and (13) can be rewritten as:

$$\begin{cases} k_{rs}(t) > f & \text{if} \quad v_{rs}(t+1) = 0 \\ k_{rs}(t) \leq f & \text{if} \quad v_{rs}(t+1) = 1 \end{cases} \tag{15}$$

$$\Delta E \begin{cases} < 0 & \text{if} \quad v_{rs}(t+1) - v_{rs}(t) = -1 \\ = 0 & \text{if} \quad v_{rs}(t+1) = v_{rs}(t) \\ < 0 & \text{if} \quad v_{rs}(t+1) - v_{rs}(t) = 1 \text{ and } f > k_{rs}(t) \\ > 0 & \text{if} \quad v_{rs}(t+1) - v_{rs}(t) = 1 \text{ and } f = k_{rs}(t) \end{cases} \tag{16}$$

From (15) we determine the stable states of the network. These are the states for which

$$\forall (p, q) \in \{1, \ldots, n\} \times \{1, \ldots, n\} \qquad \Big(v_{pq}(t) = v_{pq}(t+1)\Big)$$

(i) $f < 0$: there are no stable states in this case,
(ii) $f = 0$:

$$\begin{cases} k_{pq}(t) \geq 1 & \forall (p, q) : v_{pq}(t) = 0 \\ k_{pq}(t) = 0 & \forall (p, q) : v_{pq}(t) = 1 \end{cases}$$

In this case the stable states of the network are exactly the solutions. The energy in that case is equal to 0.
(iii) $f > 0$: stable states in that cases are the false stable states,

$$\begin{cases} k_{pq}(t) \geq f+1 & \forall(p,q) : v_{pq}(t) = 0 \\ k_{pq}(t) \leq f & \forall(p,q) : v_{pq}(t) = 1 \end{cases}$$

- $f = 1$: each unoccupied field is attacked at least twice and the non-empty fields are attacked at most once.
- $f = 2$: the empty fields are attacked at least three times and the non-empty fields are attacked at most twice.
- $f \geq 3$: according to *Proposition* 1 (placed below) the stable states for $f \geq 3$ do not exist.

In order to formulate propositions about the existence of stable states we introduce the following definition:

Definition 5 *(k-dominant set of queens)*
The set $D^n_{k,l} \stackrel{\text{def}}{=} \{(r,s) : \ v_{rs}(t) = 1\}$, where l is the power of that set, and n is the problem size, is called the *k-dominant set of queens* at time t (simply: *k-dominant set*) if condition (β) of the *Definition* 2 is fulfiled and k is the largest number among those which fulfil the following condition:
(γ) $\forall(r,s) \qquad \Big(v_{rs}(t) = 0\Big) \Longrightarrow \Big(k_{rs}(t) \geq k\Big)$

Examples of the k-dominant sets for $n = 8, k = 1, 2$ and various l are presented in Figs. $3 - 6$

Corollary 1
(i) - condition (β) together with cond. (γ) for $k = 2$ *do not* imply cond. (α),
(ii) - condition (α) with cond. (γ) for $k = 2$ *do not* imply cond. (β).
Proof:
(i) - see Fig. 3 as a counter-example for n=8,
(ii) - the implication is not true for n queens placed on the main diagonal of the chessboard.

Now, let us show that potential solutions of the problem, from the theoretical viewpoint, are only those states of the network that represent 2- or 3-dominant sets.

Proposition 1 *(estimation of the parameter k for k-dominant sets)*
For any $n, l \in \mathcal{N}$ the set $D^n_{k,l}$ *does not* exist for $k \geq 4$.

Proof:
A chess queen can attack at most $4(n-1)-1$ fields (a row, a column and two diagonals). Thus l queens can attack less than $4l(n-1)$ fields. The number of empty fields in case l queens are placed on the chessboard is equal to $n^2 - l$.
Since for $l < n$

$$4l(n-1) < 4n(n-1) = 4(n^2-n) < 4(n^2-l)$$

then, $k < 4$ in that case.
For $l = n$ from (β) we have that, the queens are placed in the pairwise different rows, columns and diagonals. Since the n queens that fulfil (β) attack along exactly $2n$ diagonals, and there exist $2(2n-3)$ diagonals of length not less than 2, then from $2n-3 > n$ for $n > 3$ we have that there must exist an empty field that is attacked along at most one diagonal. That field cannot be attacked more than 3 times.
The subcases for $n = 1, 2, 3$ with $l = n$ and the subcase $l > n$ are obvious.

Corollary 2
Every solution is either a 2-dominant set or a 3-dominant set.

Proof:
From (β) we obtain that the queens are placed in the pairwise different rows, columns (and also diagonals). Since $m(t) = n$, then each empty field is attacked at least twice. Thus, $k \geq 2$. According to *Proposition* 1, $k \leq 3$.

2.2.3 Conclusions

Comparing the above theoretical analysis with the simulation results (Fig. 2 and sect. 2.2.1) leads to the following remarks:
(**i**) since, in practice, the convergence rate is equal to 100%, the number of stable states for $f = 1$ and $f = 2$, i.e. with $n-1$ and $n-2$ queens placed, resp. is relatively small,
(**ii**) from the theoretical analysis it is clear that instead of $\sigma = 0$, any other value from the interval $(0, 0.5)$ may be used with exactly the same quality of the results,
(**iii**) in case $\sigma = 0.5$, from (13) there exist *quasi-stable states*, e.g.:

$$m(t) = n \qquad \text{and} \qquad \exists p, q : (v_{pq}(t) = 0) \wedge (k_{pq}(t) = 0)$$

$$m(t) = n \qquad \text{and} \qquad \exists p, q : (v_{pq}(t) = 1) \wedge (k_{pq}(t) = 1)$$

$$m(t) = n - 1 \qquad \text{and} \qquad \exists p, q : (v_{pq}(t) = 0) \wedge (k_{pq}(t) = 1)$$

$$m(t) = n - 1 \qquad \text{and} \qquad \exists p, q : (v_{pq}(t) = 1) \wedge (k_{pq}(t) = 2)$$

etc.

In each of the above cases chosing neuron (p, q) results in changing its state without changing the network energy,
(**iv**) by the simulation results the average number of external iterations is, except for $n = 4$, about $4.31 \cdot n^{0.47}$, which implies the average experimental computational complexity of the method be polynomial and equal to $O(n^{4.47})$ (the average number of external iterations, by $O(n^2)$ neurons updated in each iteration, by $O(n^2)$ elementary operations required to compute the neuron's potentials).

We would briefly describe how values 4.31 and 0.47 were obtained. Let $T(n)$ be the function of the average number of external iterations required in one simulation test. Based on the values of $T(n)$ except for $n = 4$ it was clear that the shape of the function resembles $4n^{0.5}$. Since the results for $n = 4$, which were of the least importance, had unproportionally big influence on the function shape they were left out.

$T(n)$ was assumed to be of the form:

$$T(n) = a \cdot n^k$$

Thus,

$$ln(T(n)) = ln(a) + k \cdot ln(n) \tag{17}$$

which is of the linear form with respect to parameters a and k. The *least square fit* method applied to the form (17) led to the solution

$$a = 4.31 \qquad \text{and} \qquad k = 0.47$$

with the square fit error for the base, i.e. *not logarithmic* form equal to 3.557 (the biggest component was equal to 2.087 - for $n = 16$) and the biggest absolute linear error equal to 8.31% - also for $n = 16$.

Applaying the same as above procedure to the values of $T(n)$ with $n = 4$ included led to the following results:

$$a = 2.31 \qquad \text{and} \qquad k = 0.63$$

with square fit error equal to 43.93 (the biggest value was equal to 14.96 - for $n = 16$), and the biggest absolute linear error equal to 22.27% - for $n = 16$.

2.3 Synchronous mode

In the synchronous mode the simulation test was performed according to the following steps:

(**i**) all initial output potentials v_{ij} $(i, j = 1, \ldots, n)$ were set and from (5) the starting value of energy E was evaluated,

(**ii**) n neurons were randomly chosen and from (6) and (7) the corresponding potentials were calculated (simultaneously for all n neurons),

(**iii**) operation (**ii**) was repeated $5n$ times, and then the value of the energy, from (5) was updated.

Similarily to the asynchronous case every n repetitions of (**ii**) was called an *internal iteration*, and five internal iterations composed an *external* one.

The termination criterion was the same as the one used in the asynchronous case.

In all tests the following values of the parameters were used:

$$A = B = 100, \quad C = 40, \ \sigma = 2$$

and like in the former case, for each n from (8) - except for $n = 80$, 100 tests were performed. Unfortunately, for $n = 80$, the equipment used (PC 80486/33) appeared to be too slow to finish the simulations in the reasonable time.

2.3.1 Simulation results

Based on the simulation results we can say that:
(**i**) the convergence rate is equal to 100% except for the case $n = 16, (a)$ and $n = 16, (c)$ for which is equal to 99%,
(**ii**) there are no significant differences in speed among the initial strategies (see Fig. 7),
(**iii**) simulations for $n = 64$ were also performed with $\sigma = 3$, which resulted in lower convergence rate (90%), but much smaller number of iterations (e.g. 27.48 for strategy (b)).
(**iv**) the function estimating the average number of external iterations is, except for $n = 4$, of the form $2.82 \cdot n^{0.67}$, which implies the average experimental computational complexity of the method be polynomial and equal to $O(n^{4.67})$ - *for the sequential RAM machine* - (the average number of external iterations, by $O(n^2)$ neurons updated in each external iteration, by $O(n^2)$ elementary operations required to compute the neuron's potentials).

We used the same method as the one described in the asynchronous case (cf. 2.2.3) to obtain values for a and k, and for the same reason the case $n = 4$ was omitted.

The respective errors were equal to 3.35 (the biggest component being equal to 2.16 - for $n = 48$), and 4.26% - for $n = 16$.

In case $n = 4$ was included we obtained $a = 1.71$, $k = 0.81$, with the errors respectively equal to 36.89 (max. component equal to 13.93 - for $n = 64$) and 14.30% - for $n = 16$.

Unfortunately, we were unable to successfully simulate the network for $O(n^2)$ parallely (synchronously) chosen neurons, because in that case the network usually oscillated between the two metastable states, and wasn't able to settle.

3 Final remarks

In this paper a completely binary model of the network for the N-Queens Problem is described. The network can work either in the asynchronous mode or in the synchronous mode with n processors, where n is the problem size.

The convergence rate is very high, up to 100%.

The experimental estimate of the computational complexity of the proposed network is (according to the performed simulations) equal to $O(n^{4.47})$ in the asynchronous case, and $O(n^{4.67})$ in the synchronous case (performed on the sequential computer).

Based on the theoretical analysis of the network performance in the asynchronous case we can describe and characterize the stable states, the solutions, as well as the false stable states of the network. Such analysis is more complicated in the synchronous case, but also in that case the adequate set of the parameters was established.

Generally speaking, the proposed binary network seems to be useful in solving other optimisation problems, especially those which are discrete. Based on that model we have also tried to solve the TSP, with some success, but we are not yet satisfied and convinced whether such limited, binary approach is strong enough for solving the TSP.

In future we plan to show network's efficiency in solving other optimization problems.

References

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

[2] 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. ICNN*, 333-340,

[3] Hopfield J.J., (1982), **Neural networks and physical systems with emergent collective computational abilities**, *Proc. Natl. Acad. Sci. USA*, 79:2554-2558,

[4] 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,

[5] Hopfield J.J., Tank D.W., (1985), **"Neural" Computation of Decisions in Optimization Problems**, *Biological Cybernetics*, 52:141-152,

[6] Kamgar-Parsi B., Kamgar-Parsi B., (1990), **On Problem Solving with Hopfield Neural Networks**, *Biological Cybernetics*, 62:415-423,

[7] Mańdziuk J., Macukow B., (1992), **A neural network designed to solve the N-Queens Problem**, *Biological Cybernetics*, 66:375-379,

[8] Mańdziuk J., (1993), **Application of neural networks to solving optimization problems**, *Ph.D. Thesis, Warsaw University of Technology, (in Polish)*,

[9] Wilson G.V., Pawley G.S., (1988), **On the stability of the Travelling Salesman Problem algorithm of Hopfield and Tank**, *Biological Cybernetics*, 57:63-70,

[10] 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.

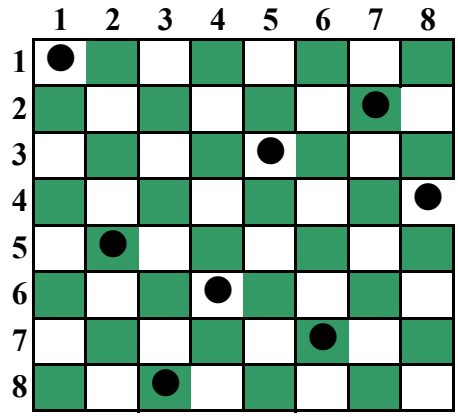

Figure 1: An example of the NQP solution for *n=8*

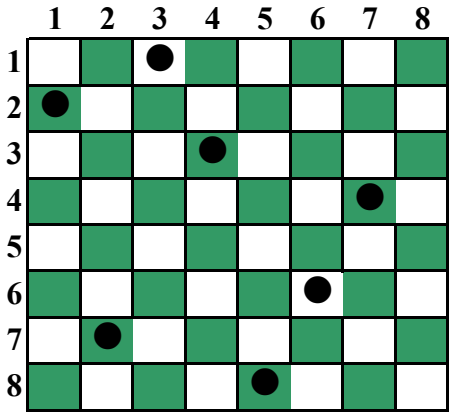

Figure 3: An example of the *k-dominant* set for *n=8*, ***k=1*** *and* ***l=n-1***

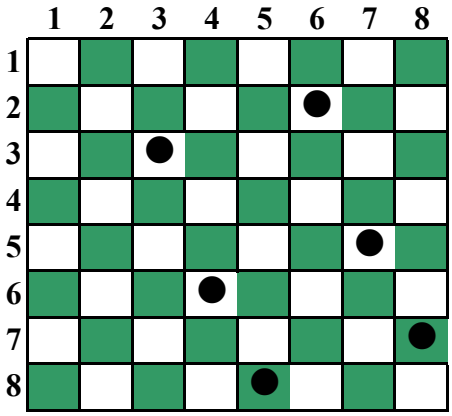

Figure 4: An example of the *k-dominant* set for *n=8*, ***k=1*** *and* ***l=n-2***

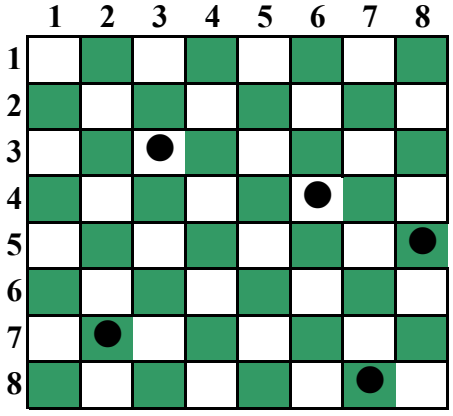

Figure 5: An example of the *k-dominant* set for *n=8*, ***k=1*** *and* ***l=n-3***

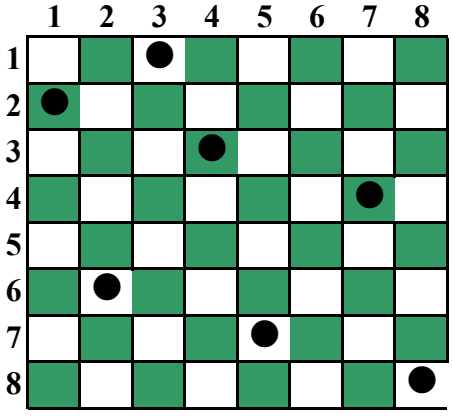

Figure 6: An example of the *k-dominant* set for *n=8*, ***k=2*** *and* ***l=n-1***

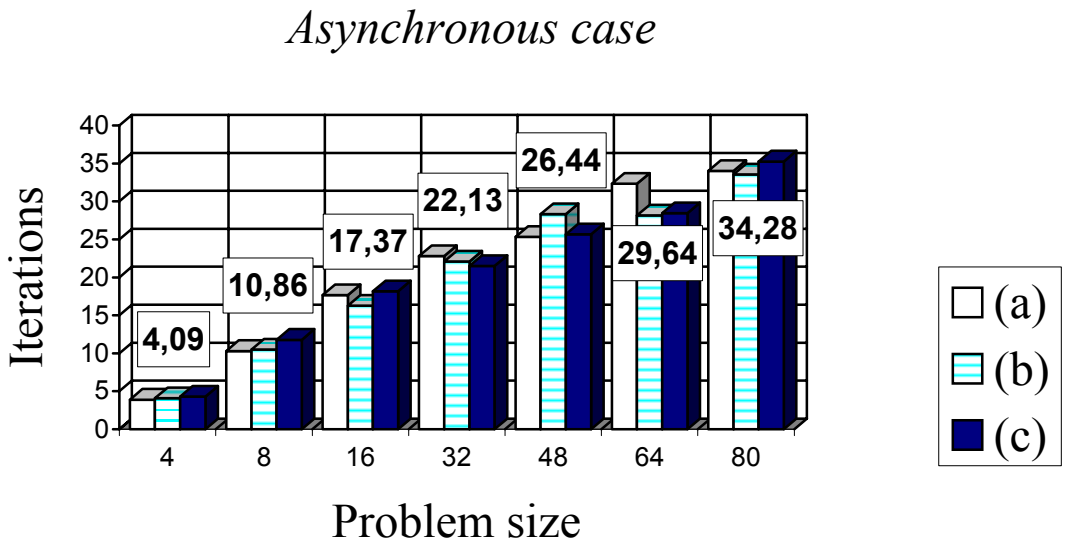

Figure 2: The average number of external iterartions for starting strategies (a), (b) and (c) in **asynchronous mode**. For each n, the number on the respective figure represents the mean value over the set of average values obtained in the cases (a), (b), (c).

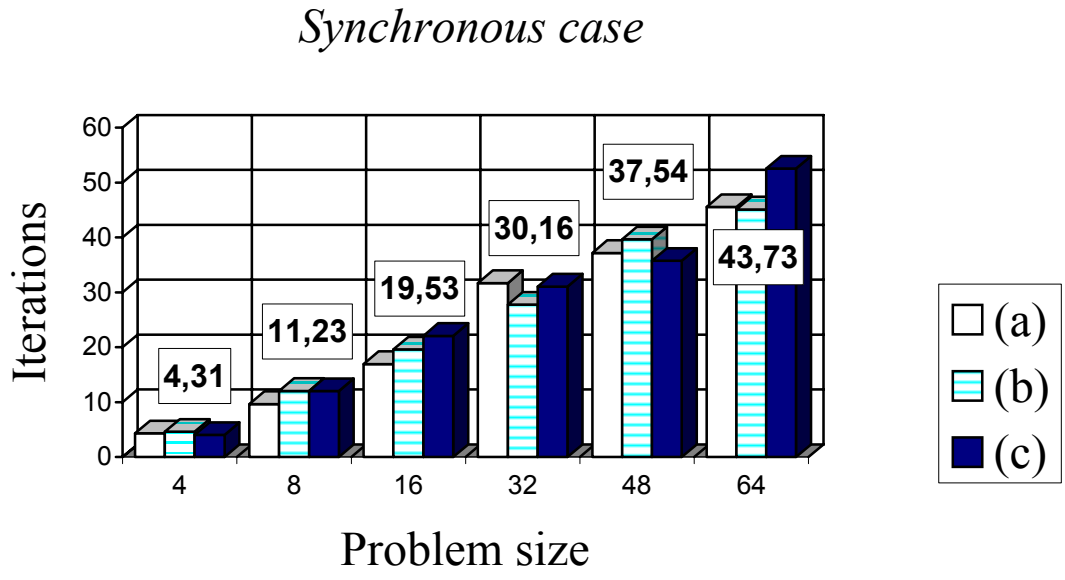

Figure 7: The average number of external iterartions for starting strategies (a), (b) and (c) in **synchronous mode**. For each n, the number on the respective figure represents the mean value over the set of average values obtained in the cases (a), (b), (c).