Control and Cybernetics

vol. **31** (2002) No. 2

Neural networks for the N-Queens Problem: a review

by

Jacek Mańdziuk

Faculty of Mathematics and Information Science,
Warsaw University of Technology
Plac Politechniki 1, 00-661 Warsaw, Poland
mandziuk@mini.pw.edu.pl

Abstract: Neural networks can be successfully applied to solving certain types of combinatorial optimization problems. In this paper several neural approaches to solving constrained optimization problems are presented and their properties discussed. The main goal of the paper is to present various improvements to the well-known Hopfield models which are intensively used in combinatorial optimization domain. These improvements include *deterministic* modifications (binary Hopfield model with negative self-feedback connections and Maximum Neural Network model), *stochastic* modifications (Gaussian Machine), *chaotic* Hopfield-based models (Chaotic Neural Network and Transiently Chaotic Neural Network), *hybrid* approaches (Dual-mode Dynamic Neural Network and Harmony Theory approach) and finally modifications motivated by *digital implementation* feasibility (Strictly Digital Neural Network).

All these models are compared based on a commonly used benchmark problem - the N-Queens Problem (NQP). Numerical results indicate that each of modified Hopfield models can be effectively used to solving the NQP. Convergence to solutions rate of these methods is very high - usually close to 100%. Experimental time requirements are generally low - polynomial in most cases.

Some discussion of non-neural, heuristic approaches to solving the NQP is also presented in the paper.

Keywords: Hopfield network, N-Queens Problem, neural networks, combinatorial optimization

1. Introduction

In the framework of neural networks there are two possible approaches to solving constraint optimization problems: *evolving deformable template matching* and *Hopfield-type gradient minimization.* The first approach requires the problem to be solved to have appropriate geometrical representation. *Elastic nets*

Le Texier, 1988) are two well known examples of such methods. Typical application domain is the Travelling Salesman Problem (TSP), where both approaches have proved their efficiency (Smith, Potvin and Kwok, 2002). In case of the N-Queens Problem (NQP), however, the condition concerning appropriate geometrical representation of the problem required by these methods is hard to fulfil and therefore solving the NQP with neural nets is actually limited to the Hopfield-type approaches.

Hopfield models (HMs) were introduced in two seminal papers (Hopfield, 1982, 1984) and fully developed in subsequent works (Hopfield and Tank, 1985, 1986; Tank and Hopfield, 1986, 1987). Since their introduction Hopfield models have attracted attention of hundreds of researchers, which resulted in enormous number of papers devoted to that subject. Except for their undisputed advantages HMs have also some drawbacks, the main of them being high possibility of being trapped in a local minimum of the search space which usually represents a good - but not optimal - solution. This limitation has been alleviated by combining gradient minimization of HMs with non-deterministic global minimization methods. This resulted in several stochastic, chaotic and hybrid extensions. In stochastic extensions (Akiyama et al., 1991; Wong, 1991; Mańdziuk, 2000a) escaping from local minima is based on adding stochastic (usually Gaussian) noise to the model. The noise is able to drive the system out of local minima. Similar idea is made use of in chaotic extensions (Nozawa, 1992; Chen and Aihara, 1995; Wang and Smith, 1998). The difference lies in the nature of a driving force, which in this case is deterministic chaos. Another group of Hopfield-based models includes deterministic modifications of the Hopfield networks focused on more efficient problem representation, e.g. the Maximum Neural Network (Funabiki, Takenaka and Nishikawa, 1997). Finally, there are several hybrid Hopfield-type models in which HMs are combined with non-gradient optimization techniques, e.g. Hopfield-Lagrange models (Zhang and Constantinides, 1992; Li, 1996) or Dual-mode Dynamics Neural Network (Lee and Park, 1995). For an overview of Hopfield models' extensions please refer to Mańdziuk (2000b) or Smith, Potvin and Kwok (2002).

In the literature the efficacy of neural optimization methods is typically presented based on their application to the Travelling Salesman Problem. There are two main reasons for such a choice: first, TSP was originally chosen by Hopfield and Tank to illustrate the properties of HMs; second, TSP is the well known NP-hard benchmark problem for operation research methods and heuristic algorithms.

On the other hand, some reasons exist for not considering the TSP as a benchmark problem in this comparative study. As stated in Smith (1996) the TSP is actually not a good benchmark problem for making comparisons between neural and heuristic methods due to a different problem formulation - heuristic and operation research methods take advantage of linear TSP description

Considering the above it seems appropriate to choose a benchmark problem other than the TSP. No particular reasons exist for considering the NQP in that place except for just two: personal interests of the author and availability of papers devoted to solving the NQP with various Hopfield-type models. Moreover, unlike the TSP, the NQP is defined by one parameter only - the problem size, and its solution does not depend on additional context data. This makes results obtained by different methods easily comparable. It should be noted, however, that the NQP is an "easier" problem since its time complexity is polynomial whereas the TSP belongs to the class of NP-hard problems[1].

The paper is organized as follows: In the next section the N-Queens Problem is formulated and several heuristic approaches to solving it are presented. Section 3. introduces classical Hopfield models and describes the generic representation of a constrained optimization problem within their framework. In Section 4. a simple but powerful modification to the binary model based on applying negative self-feedback connections is presented. Section 5. describes the Maximum Neural Network model - an important modification to the classical model, which allows decreasing the number of constraints in the NQP formulation. Section 6. discusses stochastic extension of the HM called Gaussian Machine. The next section describes two hybrid approaches based on combining Hopfield model with non-gradient optimization techniques. Finally, in Section 8. chaotic extensions of HM and their applications to solving the NQP are summarized. An example of hardware motivated Hopfield-type model is also presented in this section. Conclusions are placed in the last section.

2. The N-Queens problem

The N-Queens Problem of size n can informally be stated as follows: place n chess queens on $n \times n$ chessboard so that they do not attack each other. A chess queen attacks along the entire row, column and two diagonals going through the square it is placed on. Certainly, the above implies that necessary and sufficient conditions for the solution of the NQP are the following:

(i) there is exactly one queen placed in each row,
(ii) there is exactly one queen placed in each column,
(iii) there is at most one queen placed on each diagonal.

Formally the problem can be defined in the following way:

DEFINITION 1 [N-Queens Problem] A square $n \times n$ Boolean matrix V represents the solution of the NQP of size n if and only if the following conditions are fulfilled:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij} = n, \tag{1}$$

$$\forall i,j,k,l \in \{1,\ldots,n\} \qquad ((i,j) \neq (k,l) \wedge v_{ij} = 1 \wedge v_{kl} = 1) \Longrightarrow \\ (i \neq k \wedge j \neq l \wedge i-j \neq k-l \wedge i+j \neq k+l). \quad \blacksquare \tag{2}$$

In the above definition $v_{ij} = 1$ represents a queen placed on the square (i,j) of the chessboard. Otherwise, if this square is unoccupied, $v_{ij} = 0$. Condition (1) implies that there are exactly n queens placed on the board. Condition (2) implies that any two different queens placed on the squares (i,j) and (k,l) belong to *different rows* $(i \neq k)$, *different columns* $(j \neq l)$ and *different diagonals* $(i-j \neq k-l$ and $i+j \neq k+l)$.

Interactions along diagonals include two cases: those along diagonals parallel to $(1,1)-(n,n)$ main diagonal and those along diagonals parallel to the other main diagonal $(1,n)-(n,1)$. In the former case it can be observed that for each diagonal the *difference* between row and column indices of its squares is constant, whereas in the latter case the *sum* of these indices is constant (refer, please, to Mańdziuk and Macukow, 1992, for additional explanations).

The NQP can be also formulated in the form of a constrained combinatorial optimization problem:

DEFINITION 2 [The NQP as a constrained optimization problem - ver. 1] Find minimum

$$\begin{aligned} &\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n} v_{ik}v_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq i}}^{n} v_{kj}v_{ij} \\ &+ \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sum_{\substack{k=i-j+1\\k\neq i}}^{n} v_{k,k-i+j}v_{ij} + \sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{\substack{k=1\\k\neq i}}^{n+i-j} v_{k,k-i+j}v_{ij} \\ &+ \sum_{i=1}^{n}\sum_{j=n-i+1}^{n}\sum_{\substack{k=i+j-n\\k\neq i}}^{n} v_{k,i+j-k}v_{ij} + \sum_{i=1}^{n-1}\sum_{j=1}^{n-i}\sum_{\substack{k=1\\k\neq i}}^{i+j-1} v_{k,i+j-k}v_{ij} \end{aligned} \tag{3}$$

under the following constraints

$$\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij} = n, \tag{4}$$

$$v_{ij} \in \{0,1\}, \qquad i,j = 1,\ldots,n. \quad \blacksquare \tag{5}$$

The first two components in (3) represent interactions in rows and columns, respectively. The next four components are responsible for interactions along diagonals in four main triangle submatricies. It can easily be checked that with the constraints (4)–(5) the global minima of (3) correspond to exactly these

There exist a few other equivalent definitions of the NQP as a combinatorial optimization problem. One of them, commonly used in neural network applications is presented below.

DEFINITION 3 [The NQP as a constrained optimization problem - ver. 2] Find minimum

$$\sum_{p=2}^{2n} \sum_{i+j=p} \sum_{\substack{k+l=p \\ k\neq i}} v_{ij}v_{kl} + \sum_{p=-(n-1)}^{n-1} \sum_{i-j=p} \sum_{\substack{k-l=p \\ k\neq i}} v_{ij}v_{kl}, \quad i,j = 1,\ldots,n \tag{6}$$

under the following conditions

$$\sum_{i=1}^{n} v_{ij} = 1, \quad j = 1,\ldots,n, \tag{7}$$

$$\sum_{j=1}^{n} v_{ij} = 1, \quad i = 1,\ldots,n, \tag{8}$$

$$v_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,n. \quad \blacksquare \tag{9}$$

Again, $v_{ij} = 1$ represents a queen placed on the square (i,j) of the chessboard. The first term in (6) represents interactions along one type of diagonals and the second one - along diagonals of the other type. Constraints (7) and (8) represent interactions in columns and rows, respectively.

2.1. Backtracking search method

Classical approach to solving the NQP is based on *backtracking search* method. The algorithm starts off with an empty chessboard and systematically attempts to put a new queen - starting from the square $(1,1)$. The newly added queen is placed in the next possible (empty) column in the row with the smallest possible index under the general rule that it does not attack any of the queens placed so far (Wirth, 1976). Suppose that $i < n$ queens are already placed in the first i columns according to the above rules. If location of the $i+1$-th queen in column $i+1$ is not possible the algorithm tries to move the queen in the i-th column into another row (with the smallest possible index) and if it succeeds then an attempt to put the queen in column $i+1$ is repeated. If it is not possible to find acceptable row in column i, then the algorithm backtracks to column $i-1$ and tries to reassign the queen in that column, and so on. If the solution of the problem exists the algorithm will find it. If it does not, the algorithm will stop after moving back to the first column and failing to find acceptable row in that column. The solution found by this method for $n = 8$ is presented in Fig. 1. Simple modifications to the algorithm allow to find all solutions for a given n. The method is intuitive and easily implementable but its exponential

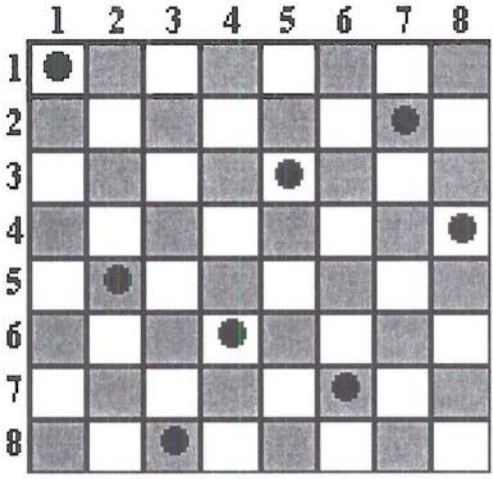

Figure 1. The first NQP solution found by backtracking method for $n = 8$

2.2. Heuristic methods

Among the heuristic approaches the most effective algorithm (in terms of time complexity) is $QS1$ (Sosic and Gu, 1990) and its enhanced version $QS4$ (Sosic and Gu, 1991a, b). The expected experimental time complexity of both of them is linear. $QS1$ is based on local, probabilistic search. First, a permutation σ of indices $\{1, \ldots, n\}$ is randomly chosen and the queens are placed on the squares $(i, \sigma(i))$, $i = 1, \ldots, n$. Certainly, such initial configuration automatically eliminates all conflicts in rows and columns. Conflicts on the diagonals are gradually eliminated by applying the following heuristic procedure: two row indices i and j are randomly selected and the algorithm temporarily moves queen $(i, c(i))$ to $(i, c(j))$ and queen $(j, c(j))$ to $(j, c(i))$, where $c(i), i = 1, \ldots, n$ denotes the column index of the queen in row i (at the beginning $c(i) = \sigma(i)$, $i = 1, \ldots, n$). A new configuration is accepted if and only if it decreases the number of diagonal conflicts on the chessboard. The above procedure is repeated until all conflicts along diagonals are eliminated.

$QS4$ is a modification of $QS1$ in which the initial configuration is chosen so as to eliminate as many diagonal conflicts as possible right from the beginning. Consequently, in consecutive rows queens are placed in colums which do not cause conflicts with any of the previously placed queens. The column index for a queen in the currently considered row is randomly chosen among those indices which fulfil the above condition. If for some i placing the queen in row i without conflict is not possible, the rest of $n - i + 1$ queens are placed in the last $n - i + 1$ rows according to $QS1$ scheme. The procedure of gradual diagonal

(1991a) $QS4$ solves the NQP of size 3 000 000 in less than a minute on IBM RS 6000/530 computer.

Another interesting heuristic approach is based on the *min-conflict heuristics* in which, after the initialization phase (analogous to QS1 or QS4) the following procedure is repeatedly applied: the "most attacked" queen (i.e. the one which causes the highest number of conflicts with other queens) is moved within its column to the square on which it causes the smallest possible number of conflicts (the *min-conflicting* square in the column). In case of two or more possible reassignments the random choice is applied among them. The algorithm is very effective in solving the NQP and according to Russel and Norvig (1995) it solves Million-Queens Problem in less than 50 iteration steps.

2.3. Exact solution

An interesting and not widely known fact is the existence of an exact, analytical solution of the NQP for any $n \geq 4$ (Hoffman, Loessi and Moore, 1969, see also Bernhardsson, 1991). Actually,

- for n even and $n \neq 6k+2, k \in \mathcal{Z}$ a solution is the following configuration:

$$\begin{cases} (i, 2i) \\ (\frac{n}{2} + i, 2i - 1) \end{cases} \quad \text{for} \quad i = 1, \ldots, \frac{n}{2}, \tag{10}$$

- for n even and $n \neq 6k, k \in \mathcal{Z}$ a solution is the following configuration:

$$\begin{cases} (i, 1 + [(2(i-1) + \frac{n}{2} - 1) \pmod n]) \\ (n + 1 - i, n - [(2(i-1) + \frac{n}{2} - 1) \pmod n]) \end{cases} \quad \text{for } i = 1, \ldots, \frac{n}{2}, \tag{11}$$

- for odd n the appropriate case among the above two is applied to $n-1$ and the configuration is completed by placing the queen on the square (n, n).

It is worth noting that the existence of analytical solution of the NQP for any $n \geq 4$ does not question the sense of a search for efficient algorithmical solutions. The existence of analytical solution has no influence on the level of difficulty, which heuristic or AI-based methods actually encounter while solving the problem. Furthermore, the NQP in its wider formulation consists in finding **all solutions** for a given n. In such a case no analytical or polynomially bounded heuristical method is known. In this context the Hopfield neural network method gains great importance due to its ability to generate different solutions depending on initial conditions.

3. Hopfield models

Hopfield network can be implemented in three general forms depending on how *time* and *states* are represented. In this paper *binary model* will denote a model

discrete time and continuous states, and *continuous model* will refer to the continuous time and continuous states model. In most applications either a binary or a continuous model is used. In the next two subsections a very brief description of these two models is presented. For more details please refer to Hopfield's papers (Hopfield, 1982, 1984; Hopfield and Tank, 1985, 1986) or to Mańdziuk (2000b).

3.1. Binary Hopfield model

Binary HM of size N is composed of N fully connected two-state McCulloch-Pitts (1943) neurons $neu_i, i = 1, \ldots, N$. Input activation $u_i(t)$ of neuron neu_i at time t is calculated as

$$u_i(t) = \sum_{j=1}^{N} t_{ij} v_j(t-1) + I_i, \quad i = 1, \ldots, N, \tag{12}$$

where $v_j(t-1)$ is the output activation of neuron neu_j at time $t-1$, I_i is the external input to neuron neu_i and t_{ij} is connection weight from neu_j to neu_i. It is assumed that $t_{ij} = t_{ji}, i, j = 1, \ldots, N$ and $t_{ii} = 0, i = 1, \ldots, N$. Output activation $v_i(t)$ of neuron neu_i at time t is calculated as a binary transformation of its input:

$$v_i(t) = \begin{cases} 1 & \text{if } u_i(t) > 0, \\ 0 & \text{if } u_i(t) \leq 0. \end{cases} \tag{13}$$

At each (discrete) time step some number of neurons calculate their input and then output activations. The network usually operates in one of the two modes: the *synchronous* one - in which *all* neurons update their states at the same time governed by the central clock or the *asynchronous* one - where at each time step only *one* randomly selected neuron updates its input and output activation and the other neurons do not change their states. It can easily be shown (Hopfield, 1982) that in case of asynchronous update and symmetric weights with non-negative self-feedback ($t_{ii} \geq 0,\ i = 1, \ldots, N$) regardless of initial configuration of states $v_i(0),\ i = 1, \ldots, N$, the network converges monotonically to a stable state. This stable state corresponds to a local minimum of the following quadratic energy function:

$$E(t) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} v_i(t) v_j(t) - \sum_{i=1}^{N} I_i v_i(t). \tag{14}$$

In case of synchronous update the network either converges to a stable state or to a cycle of length 2, i.e. it alternately changes between two states $A \longrightarrow$

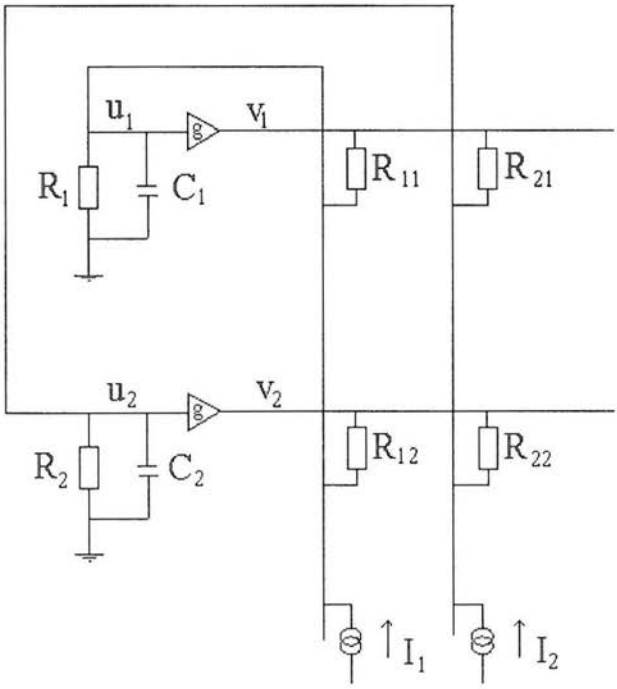

Figure 2. Schematic representation of continuous Hopfield model as an electric circuit. For the sake of clarity only two neurons are presented

3.2. Continuous Hopfield model

Continuous HM has the same topology as the binary model but a different time and states representation. The model can be described and implemented as an electric circuit - see Fig. 2. In Fig. 2, C_i and R_i denote, respectively, capacity and resistance of the neuron $neu_i, i = 1, \ldots, N$, and R_{ij} represents resistance of connection from neu_j to neu_i, $i, j = 1, \ldots, N$. Interactions between neurons in the continuous model are described by the following set of differential equations:

$$C_i \frac{du_i(t)}{dt} = \sum_{j=1}^{N} t_{ij} v_j(t) + I_i - \frac{u_i(t)}{r_i}, \tag{15}$$

where $u_i(t), v_i(t), I_i$ have the same meaning as in binary model, $t_{ij} = \frac{1}{R_{ij}}$ is a weight of connection from neu_j to neu_i, and

$$\frac{1}{r_i} = \frac{1}{R_i} + \sum^{N} t_{ij}. \tag{16}$$

The transfer function $g : u_i(t) \to v_i(t)$ between input and output activation of neuron neu_i is defined as

$$v_i(t) = g(u_i(t)) = \frac{1}{2}(1 + \tanh(\alpha u_i(t))), \tag{17}$$

where α controls the slope of the sigmoid function (17).

Setting circuit parameters according to the following conditions:

$$C_i = C_j \quad \text{and} \quad r_i = r_j, \quad i, j = 1, \ldots, N, \tag{18}$$

and redefining them as in (19)

$$t_{ij} := \frac{t_{ij}}{C_i}, \quad I_i := \frac{I_i}{C_i}, \quad r_i := R, \quad C_i := C, \quad i, j = 1, \ldots, N, \tag{19}$$

simplifies the set of equations (15) to the following form:

$$\frac{du_i(t)}{dt} = \sum_{j=1}^{N} t_{ij} v_j(t) + I_i - \frac{u_i(t)}{RC}, \quad i = 1, \ldots, N, \tag{20}$$

where RC (also denoted by τ) represents the so-called *relaxation time* of the electrical circuit.

The model (20), regardless of the choice of initial state, converges monotonically to a stable state corresponding to a local minimum of the energy function

$$E(t) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} v_i(t) v_j(t) - \sum_{i=1}^{N} I_i v_i(t) + \frac{1}{RC} \sum_{i=1}^{N} C(v_i(t)), \tag{21}$$

where

$$C(x) = \int_{\frac{1}{2}}^{x} g^{-1}(\eta) d\eta. \tag{22}$$

The energy (21) is the *Lyapunov function* for the set (20). It is important to note that in continuous model *no restrictions are imposed on self-feedback connections* $t_{ii}, i = 1, \ldots, N$.

For sufficiently high α in (17) energy (21) can be simplified to

$$E(t) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} v_i(t) v_j(t) - \sum_{i=1}^{N} I_i v_i(t), \tag{23}$$

where the sum of integrals (22) is left out (see Hopfield, 1984; Smith and Portman, 1989, for details).

Note: Mathematical formalism used in description of binary and continuous Hopfield models presented in the last two subsections required that dependence on time of several neural variables (u_i, v_i, etc.) be explicitly stated. However, for the sake of clarity, henceforth the dependence on time of variables in equations as well as within the text will be omitted. The exceptions are Sections 6.1. and

3.3. Generic representation of combinatorial optimization problem in the Hopfield models

The property of minimization of energy (23) by binary, asynchronous model (12)–(13) and continuous model (17), (20) can be taken advantage of in solving the constrained combinatorial optimization problems. Consider a problem of the form:

$$\min_{v \in \mathcal{R}^N} f(v) = v^T Q v + b^T v \tag{24}$$

with symmetric matrix Q and linear constraints[2]:

$$c_j(v) = 0, \quad j = 1 \ldots, r, \quad v_i \in \{0, 1\}, \quad i = 1, \ldots, N. \tag{25}$$

Define the penalty function E^* for this problem:

$$E^*(v) = \beta_0 f(v) + \sum_{j=1}^{r} \beta_j [c_j(v)]^2 \tag{26}$$

where $\beta_i,\ i = 0, \ldots, r$ are positive coefficients.

Having compared the generic form of energy function (23) with penalty function (26) one obtains values of weights t_{ij} and external inputs I_i of HM, that correspond to solving the problem (24)–(25).

3.4. Basic properties of Hopfield models

The main advantage of Hopfield optimization networks is their wide applicability. Both models can be applied to any constrained optimization problem that can be formulated in quadratic form (24) with linear constraints (25).

On the other hand both models perform gradient descent minimization, which in general case does not guarantee that a minimum reached by the network is global. Moreover, the models are very sensitive to the choice of penalty coefficients $\beta_0, \beta_1, \ldots, \beta_r$ in (26) or, equivalently, to the choice of weights $t_{ij},\ i, j = 1, \ldots, N$ and inputs $I_i,\ i = 1, \ldots, N$ in (14) or (23), respectively.

Due to the above disadvantages the Hopfield models in their original formulation are rarely applied to solving optimization problems. In such cases they are usually combined with a heuristic multi-start procedure which allows reaching different local minima (and hopefully the global one among them) due to the use of various initial states (starting points).

There also exist several improvements of HMs which extend their original formulation and in effect break the monotonical gradient descent behavior. Examples of such methods are presented and discussed in the rest of the paper.

4. Binary model with negative self-feedback connections

In this chapter a simple modification to the binary Hopfield model is presented. The difference lies in setting (appropriately chosen) *negative self-feedback connections* for the neurons. In effect a modified model does not converge monotonically and, consequently, under certain conditions, is able to escape from local minima of the energy landscape.

4.1. Problem formulation

In binary HM the NQP of size n can be represented by a square $n \times n$ matrix. According to Definitions 1 and 2 and eq. (26) the energy function for the NQP may be defined in the following way (Mańdziuk, 1995):

$$
\begin{aligned}
E = \frac{1}{2}\Big\{ & A\sum_{i=1}^{n}\sum_{j=1}^{n}\Big[\Big(\sum_{\substack{k=1\\k\neq j}}^{n} v_{ik}\Big)v_{ij} + \Big(\sum_{\substack{k=1\\k\neq i}}^{n} v_{kj}\Big)v_{ij}\Big] \\
& + B\sum_{i=2}^{n}\sum_{j=1}^{i-1}\Big[\Big(\sum_{\substack{k=i-j+1\\k\neq i}}^{n} v_{k,k-i+j}\Big)v_{ij}\Big] + B\sum_{i=1}^{n}\sum_{j=i}^{n}\Big[\Big(\sum_{\substack{k=1\\k\neq i}}^{n+i-j} v_{k,k-i+j}\Big)v_{ij}\Big] \\
& + B\sum_{i=1}^{n}\sum_{j=n-i+1}^{n}\Big[\Big(\sum_{\substack{k=i+j-n\\k\neq i}}^{n} v_{k,i+j-k}\Big)v_{ij}\Big] \\
& + B\sum_{i=1}^{n-1}\sum_{j=1}^{n-i}\Big[\Big(\sum_{\substack{k=1\\k\neq i}}^{i+j-1} v_{k,i+j-k}\Big)v_{ij}\Big] + C\Big(\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij} - (n+\sigma)\Big)^2\Big\}, \qquad (27)
\end{aligned}
$$

where $A, B, C > 0$ and $\sigma \geq 0$. The term multiplied by A corresponds to interactions along rows and columns (at most one queen should be located in each row and in each column) and four terms multiplied by B correspond to diagonal interactions in four main triangle submatricies (again, at most one queen can be placed on each diagonal). The term multiplied by C forces the number of queens on the chessboard to be equal to n.

The role of parameter σ is explained in Section 4.2. For $\sigma = 0$ the minimum of (27) is equal to 0 and corresponds to exactly those network configurations which represent solutions of the NQP.

The input activation u_{ij} of neuron neu_{ij} is defined as follows:

$$
u_{ij} = -A\Big(\sum^{n} v_{ik} + \sum^{n} v_{kj}\Big) - R_{ij} - S_{ij} - C\Big(\sum^{n}\sum^{n} v_{kl} - (n+\sigma)\Big), \qquad (28)
$$

where

$$R_{ij} = \begin{cases} B \sum\limits_{\substack{k=i-j+1 \\ k \neq i}}^{n} v_{k,k-i+j} & \text{if } i-j > 0, \\ B \sum\limits_{\substack{k=1 \\ k \neq i}}^{n+i-j} v_{k,k-i+j} & \text{if } i-j \leq 0, \end{cases} \tag{29}$$

and

$$S_{ij} = \begin{cases} B \sum\limits_{\substack{k=i+j-n \\ k \neq i}}^{n} v_{k,i+j-k} & \text{if } i+j > n, \\ B \sum\limits_{\substack{k=1 \\ k \neq i}}^{i+j-1} v_{k,i+j-k} & \text{if } i+j \leq n, \end{cases} \tag{30}$$

for $i, j = 1, \ldots, n$. Transfer function $u_{ij} \longrightarrow v_{ij}$ is of the form (13) for all neurons.

Note that in order to fulfil conditions for monotonic convergence of binary model to a stable state (local minimum of energy) in classical Hopfield's formulation the *weights of self-feedback connections should be non-negative.* Applying this condition, however, results in a very poor convergence to solutions (e.g. smaller than 5% for $n = 8$). The idea of leaving out the above condition in binary model - in case of the NQP introduced in Mańdziuk (1995) - causes strong improvement of the convergence to solutions rate (up to 100%). Based on simulation results and analytical calculations presented in Mańdziuk (1995) it can be shown that network's ability to occasionally increase its energy can be very helpful in escaping from local minima of the energy landscape.

4.2. Results of simulations

A modified binary model defined in the previous subsection can be implemented in two modes: asynchronous (sequential) mode and partly synchronous, n-parallel one. In the former case at each time step a selected neuron neu_{ij}, $i, j = 1, \ldots, n$ updates its state based on (28) and (13). Neurons are either selected with equal probability at each time step or in the so-called randomly selected order. In the latter case a permutation of all neurons is randomly chosen at the beginning of each epoch and then neurons are sequentially updated according to that permutation.

In n-parallel mode at each time step n randomly selected neurons are updated simultaneously based on (28) and (13).

In both modes after every K time steps a new value of energy (27) is calcu-

and to $5n$ in n-parallel mode. The simulation test was stopped if the energy did not change in it_{stop} subsequent measurements or the limit it_{max} for the number of time steps was exceeded.

Simulations in asynchronous mode were performed with the following parameters:

$$n = 8, 16, 32, 64, 80, \quad A = B = C = 100, \quad \sigma = 0, \quad it_{\max} = 1000K,$$
$$it_{\text{stop}} = \begin{cases} 50, & \text{for } n \leq 32, \\ 100, & \text{for } n = 64, 80, \end{cases} \tag{31}$$

and in n-parallel mode with

$$n = 8, 16, 32, 64, \quad A = B = 100, \quad C = 40, \quad \sigma = 2, \quad it_{\max} = 1000K,$$
$$it_{\text{stop}} = \begin{cases} 50, & \text{for } n \leq 32, \\ 100, & \text{for } n = 64. \end{cases} \tag{32}$$

Selections (31) and (32) differ by the choice of C and σ. Smaller value of C in n-parallel simulations is caused by the danger of possible oscillations which may occur due to partly parallel update. Negative effect of relatively small value of C is breaking the balance between fulfilling local constraints (in rows, columns and diagonals) and the global constraint for the number of queens on the chessboard. Consequently, the network tends to decrease its overall activation and prefers states with smaller than n number of active neurons (queens). This tendency is balanced by setting $\sigma > 0$.

Three methods for setting the initial state of the network were tested in both modes:

(a) $v_{ij} = 0, \quad i, j = 1, \ldots, n,$

(b) $v_{ij} = \begin{cases} 1 & \text{with probability } \frac{1}{2}, \\ 0 & \text{otherwise}, \end{cases} \quad i, j = 1, \ldots, n,$

(c) $v_{ij} = 1, \quad i, j = 1, \ldots, n.$

For each tested n and for each of the above initial conditions 100 tests were performed in each simulation mode. The convergence rate was equal to 100% in all cases, except for $n = 8, (a)$ in the asynchronous mode when it was equal to 99% and $n = 16, (a)$ and $n = 16, (c)$ in n-parallel mode when it was also equal to 99%.

Experimental computational complexity of asynchronous mode and n-parallel one is polynomial and equal to $\mathbf{O(n^{4.47})}$ and $\mathbf{O(n^{4.67})}$, respectively, see Mańdziuk (1995, 2000b) for details.

4.3. The choice of starting point

Based on the results presented in the previous subsection it can be concluded that the choice of starting point has practically no influence on convergence rate. Moreover, as stated in Mańdziuk (1995) it has also no impact on the average

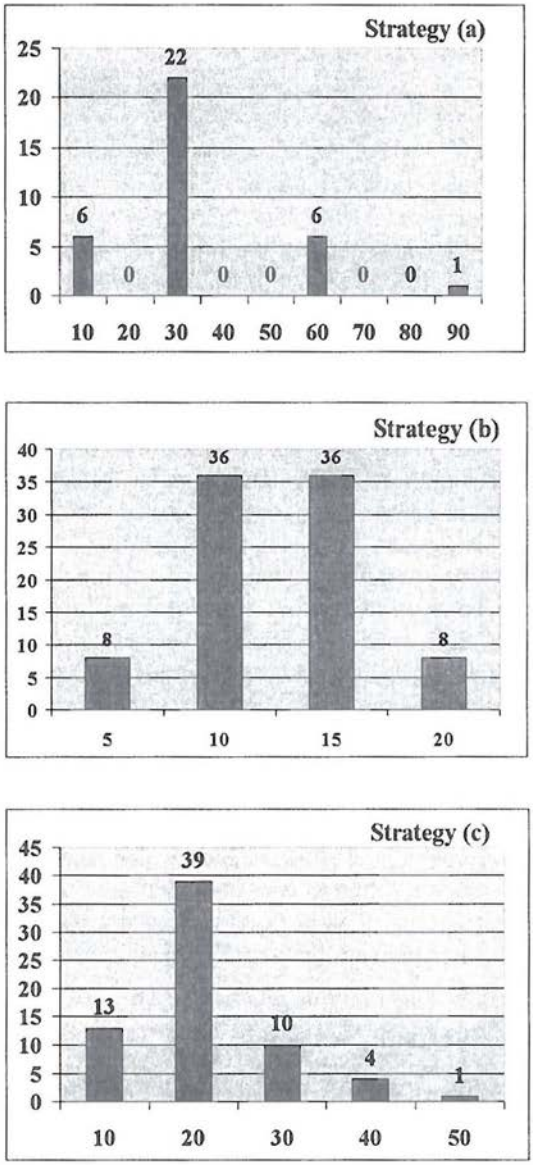

Figure 3. Frequency histograms for strategies $(a), (b)$ and (c). Values on each histogram represent the numbers of different solutions with respective frequencies. See description within the text for details

However, if the NQP is considered in a wider perspective, namely as the problem of finding **all solutions** (or at least several different ones) for a given n, it may occur that there exists some dependence between the strategy of chosing initial state and observed solutions.

In order to verify the above hypothesis, for each of initial strategies (a), (b), (c), 1 000 simulations for $n = 8$ were run. The convergence rate remained very high and equal to 99.9%, 99.8%, 100%, respectively, for strategies (a), (b), (c). The average number of iterations required to achieve a solution state remained at practically the same level: 10.66, 9.95, 10.61, respectively.

However, a closer look at solutions obtained in all three cases revealed some qualitative differences. It turned out that the number of different solutions achieved by the network was equal to 35, 92 and 67, respectively, for strategies (a), (b) and (c). These results are consistent with intuition - the basins of attraction of specific starting points generated in strategies (a) and (c) do not cover all possible solution states[3].

It is also interesting to compare the frequency of reaching particular solutions. These data are presented in Fig. 3. Each value of the histogram represents the number of different solutions related to corresponding frequency. For example, on the top histogram, y-value 22 corresponding to x-value 30 indicates that 22 different solutions (out of 35 found) were attained between 21 and 30 times each. As can be observed in Fig. 3, in strategy (a) two-third of possible solutions were either not attained or attained very rarely. On the other hand 7 solution states were achieved more than 50 times each (in 999 successful tests). In strategy (b) the shape of frequency distribution of solutions resembles Gaussian type distribution. Each solution was achieved at least once, and none of them more than 20 times. At last, in case (c) the shape of frequency distribution is familiar to Poisson type one, but with about 25% of non-attainable solutions.

In summary: a simple modification to binary HM based on setting negative self-feedback connections resulted in very high efficiency of the model regardless of the initial state choice. Almost perfect convergence to solutions achieved in the simulations can be explained analytically (Mańdziuk, 1995).

5. Maximum Neural Network

Maximum Neural Network (MNN) (Takefuji and Lee, 1992) is also an example of a simple modification to binary HM. MNN is composed of binary neurons and has the same topology as HM. The difference lies in the internal transfer (activation) function which in MNN is defined in the following way:

$$v_{ij}(t) = \begin{cases} 1 & \text{if } u_{ij}(t) = \max\{u_{i1}(t), u_{i2}(t), \dots u_{in}(t)\}, \\ 0 & \text{otherwise,} \end{cases} \quad i, j = 1, \dots, n. \tag{33}$$

Function max in (33) returns the first argument among those for which the maximum is achieved. In case of typical for HM problem representation in the form of a two-dimensional matrix the above transfer function *guarantees that there exists exactly one active element in each row* of the matrix. In most combinatorial problems solved by HM - including the NQP - this property is required in solution states. One advantage of a guaranteed fulfilment of this condition is the possibility of "natural" n-parallel implementation of MNN in which n neurons placed in the same row are updated simultaneously.

5.1. The NQP formulation

In MNN condition (1) from Definition 1 is always fulfilled[4] and therefore the energy function is designed based on condition (2) only. One possible formulation is the following (Funabiki, Takenaka and Nishikawa, 1997):

$$E = \frac{A}{2}\sum_{j=1}^{n}\Big(\sum_{k=1}^{n} v_{kj} - 1\Big)^2 \tag{34}$$
$$+\frac{B}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij}\Big(\sum_{\substack{1\le i+k,\, j+k\le n\\ k\neq 0}} v_{i+k,j+k} + \sum_{\substack{1\le i+k,\, j-k\le n\\ k\neq 0}} v_{i+k,j-k}\Big),$$

for $A, B > 0$.

The component multiplied by A reaches its minimum (equal to zero) if and only if there is exactly one queen placed in each column. The component multiplied by B is minimized (and equal to zero) if and only if at most one queen is placed on each diagonal.

In MNN the following equation describing neuron's dynamics is used:

$$\Delta u_{ij} = -A\Big(\sum_{k=1}^{n} v_{kj} - 1\Big) \tag{35}$$
$$-B\Big(\sum_{\substack{1\le i+k,\, j+k\le n\\ k\neq 0}} v_{i+k,j+k} + \sum_{\substack{1\le i+k,\, j-k\le n\\ k\neq 0}} v_{i+k,j-k}\Big) + Ch\Big(\sum_{k=1}^{n} v_{kj}\Big)$$

where

$$h(x) = \begin{cases} 1 & \text{for } x = 0,\\ 0 & \text{otherwise,}\end{cases} \tag{36}$$

is a hill-climbing function. The role of the last term in (35) is to enhance the activity of neuron neu_{ij} in case there are no active neurons in column j.

Please note that neuron's dynamics equation in MNN defines the *change* of input potential as negative gradient of the energy function, i.e. $\Delta u_{ij} =$

$-\frac{\Delta E}{\Delta v_{ij}} + h(\cdot)$ which differs from Hopfield's formulation of the binary model in which $u_{ij} = -\frac{\Delta E}{\Delta v_{ij}}$.

5.2. Simulation results

Computer simulations were performed in three modes: asynchronous (sequential) mode, n-parallel mode and synchronous (n^2-parallel) one. In the first two cases the following set of coefficients was applied:

$$A = B = 1, \quad C = \begin{cases} 1 & \text{if } t \bmod 20 < 15, \\ 4 & \text{otherwise.} \end{cases} \tag{37}$$

For n^2-parallel mode the choice of coefficients was slightly different:

$$A = B = 1, \quad C = \begin{cases} 0 & \text{if} \quad t \bmod 20 < 15, \\ 3 & \text{otherwise.} \end{cases} \tag{38}$$

In eqs. (37) and (38) t denotes time step. The parameter C in synchronous mode takes smaller values than in the other two cases in order to prevent the system from potential oscillations. In all three modes additional constraints on minimum and maximum values of input activations are imposed - see Funabiki, Takenaka and Nishikawa (1997) for details.

In sequential mode the convergence to solutions rate of MNN is between 76% and 97% for $8 \leq n \leq 50$ and equal to 100% for $100 \leq n \leq 500$. Results for n-parallel simulations reach $96-100\%$ for $8 \leq n \leq 50$ and 100% for $100 \leq n \leq 500$. It is important to note that in both simulation modes the average number of epochs required to converge to solution *is practically independent of the problem size.* Relatively poorer results are obtained for n^2-parallel mode: $40 - 97\%$ for $8 \leq n \leq 50$ and $93 - 100\%$ for $100 \leq n \leq 500$.

Experimental data suggests that MNN can be effectively used in solving the NQP, especially in n-parallel mode, when at each time step all neurons in a randomly selected row are updated simultaneously. An approximate experimental complexity of MNN in this mode is polynomial.

In the recent MNN paper by Funabiki, Kurokawa and Ohta (2002), published in this issue, it is shown that similarly to the Hopfield binary model adding negative self-feedback connections to MNN visibly improves the efficacy of the model. For example, for $n = 10\,000$ the success rate increases from 75.5% (for MNN without self-feedback connections) to 98.5% in case of modified MNN.

6. Stochastic optimization

In the previous sections two deterministic improvements to the binary Hopfield network, aimed at breaking the monotonic convergence of the model, were presented. Another group of Hopfield-type models focused on avoiding trapping in a local minimum of the energy surface are extensions of HM (binary or contin-

to higher energy states. Such stochastic extensions make use of a general purpose, global optimization method called Stochastic Simulated Annealing (SSA) (Aarts and Laarhoven, 1987; Aarts, 1989) originally proposed in Kirkpatric, Gelat Jr. and Vecchi (1983), Cerny (1985).

Stochastic extensions of continuous HM are usually derived from the Langevin Equation-based minimization (Gidas, 1986). These models include Stochastic Neural Network (Levy and Adams, 1987), Diffusion Machine (Wong, 1991), Stochastic Model (Mańdziuk, 2000a), or Pulsed Noise Model (Mańdziuk, 2000a). Introduction to stochastic Hopfield-type networks and properties of the above models are presented in Mańdziuk (2000a).

Stochastic extensions of binary HM include Boltzmann Machine (Hinton, Sejnowski and Ackley, 1984) and Gaussian Machine (Akiyama, Yamashita, Kajura and Aiso, 1989; Akiyama et al., 1991). Description of GM and its application to solving the NQP are presented in the next two subsections.

6.1. Gaussian machine

The underlying features of Gaussian Machine (GM) (Akiyama et al., 1989; Akiyama et al., 1991) are *implementation feasibility* and *low time requirements.* The topology of GM and the representation of the optimization problem being solved are the same as in HM.

The first difference between the two models lies in the definition of neuron's dynamics, which in GM is of the following form (see (20)):

$$\frac{du_i(t)}{dt} = -\frac{u_i(t)}{\tau} + \sum_{j=1}^{N} t_{ij} v_j(t) + I_i + \eta_i(t), \tag{39}$$

where $\eta_i(t) = N(0, \sigma_i(t))$ is Gaussian noise with variance decreasing in time. Noises η_i, η_j are pairwise independent, i.e. $cov(\eta_i, \eta_j) = \delta_{ij}$. σ_i^2 depends on the variable parameter $T(t)$ (called temperature):

$$\sigma_i(t) = kT(t), \tag{40}$$

where $k > 0$ is a constant parameter and T decreases in time in the following way:

$$T(t) = \frac{T_0}{1 + \frac{t}{\tau_T}}, \tag{41}$$

where T_0, τ_T are predefined constants.

Another difference with the HM is the use of adaptive sigmoidal transfer function in GM:

In the above, a_0 controls the slope of function (42) and similarly to T changes in time in a hiperbolic way:

$$a_0(t) = \frac{A_0}{1 + \frac{t}{\tau_{a_0}}}, \tag{43}$$

where A_0, τ_{a_0} are constants. In general case τ_{a_0} and τ_T can be different.

GM operates in sequential (asynchronous) mode and minimizes the energy function (21), which in the limit $a_0 \to 0$ simplifies to (23). Subsequent states of the system are chosen in a stochastic way with a long term tendency to decrease system's energy. In the initial period, temperature T is high and consequently Gaussian noises in (39) are high enough to let the system escape from any minimum of the energy surface. Along with gradual temperature decrease the intensities of noises also decrease and GM becomes more like HM. For $T = 0$, GM is equivalent to *continuous HM*.

Following Akiyama et al. (1991) it is worth emphasising that if high priority is assigned to *low time requirements* of the model, then in discretization of (39) the time step Δt can be set to

$$\Delta t = \tau = 1 \tag{44}$$

and, consequently, (39) can be approximated in a relatively simple form:

$$u_i(t+1) = \sum_{j=1}^{N} t_{ij} v_j(t) + I_i + \eta_i(t). \tag{45}$$

Therefore, in case of (44) the GM actually exemplifies stochastic extension of *discrete HM* (with discrete time and sigmoidal neurons).

6.2. Application to the NQP

In computer simulations of solving the NQP with GM presented in Akiyama et al. (1991) the following energy function was used.

$$\begin{aligned} E = {} & \frac{A}{2} \sum_{i=1}^{n} \Big(\sum_{j=1}^{n} v_{ij} - 1 \Big)^2 + \frac{B}{2} \sum_{j=1}^{n} \Big(\sum_{i=1}^{n} v_{ij} - 1 \Big)^2 \\ & + \frac{C}{2} \sum_{p=2}^{2n} \sum_{i+j=p} \sum_{\substack{k+l=p \\ k \neq i}} v_{ij} v_{kl} + \frac{D}{2} \sum_{p=-(n-1)}^{n-1} \sum_{i-j=p} \sum_{\substack{k-l=p \\ k \neq i}} v_{ij} v_{kl} \\ & + \frac{A+B}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij}(1 - v_{ij}), \end{aligned} \tag{46}$$

The first four terms in the above equation control the number of queens in rows, columns and diagonals, respectively. The last term forces binarization of solution states, since it is minimized if and only if all variables v_{ij}, $i, j = 1, \ldots, n$, are binary.

As reported in Akiyama et al. (1991) in case of $n = 8$ and the parameters

$$A = B = C = D = 1, \quad \tau_{a_0} = \tau_T = 3, \quad A_0 = 2.1, \quad T_0 = 0.05 \tag{47}$$

the convergence rate to solutions is approximately equal to 50%. It is important to note that each test took *at most 25 iterations*, i.e. every test that did not converge within 25 epochs was considered unsuccessful[5].

Our independent simulations performed with the above restrictions confirmed the rate of convergence to solutions at the level of about 50%. Additionally, a limited search for suboptimal set of system's parameters, with $A = B = C = D = 1$ fixed, was done which resulted in the following choice:

$$\tau_{a_0} = \tau_T = 3, \quad A_0 = 2.1, \quad T_0 = 0.09. \tag{48}$$

With the above set of coefficients the convergence to solutions rate was equal to 57%. It was also observed that the network in a wide range of parameters achieved $45 - 50\%$ convergence to solutions.

In summary it is interesting to notice that relatively poor results for the NQP (compared to other neural and heuristic methods) are in contrast to relatively good results achieved for the Travelling Salesman Problem (Akiyama et al., 1991). Further analysis of GM and search for possible enhancements of presented results seem to be interesting research issues. Promising modifications include application of a much smaller integration step $\Delta t \ll 1$ or increase of the limit for the number of epochs in the search process.

7. Hybrid approaches

Another group of extensions to the Hopfield networks are hybrid models which combine HMs with other optimization techniques.

7.1. Dual-mode Dynamics Neural Network

Dual-mode Dynamics Neural Network (D2NN) (Lee and Park, 1995) is an interesting example of hybrid approach in which gradient dynamics of asynchronous, binary HM is applied alternately with guided weights perturbation process. In D2NN in each epoch at first gradient minimization is performed by HM and if obtained minimum is not satisfactory, weights are corrected (according to some criteria) in order to let the system escape from current local minimum *towards the basin of attraction of the global minimum.* In other words, the weight space

is modified in a way that potentially allows the model to reach deeper minimum in the next epoch.

D2NN attempts to overcome two limitations of HM: convergence to local minima and requirement for quadratic form of energy. Actually, for some optimization problems (especially real-life ones) transformation of the cost function to the form (14) is very expensive, if at all possible. In order to avoid the above drawbacks the *cost function in D2NN is only linked (but not equal) to network's energy function.* The process of weights perturbation in D2NN - which modifies network's energy function - is guided by the values of the cost function in specific points of a solution space.

D2NN is composed of two layers: *base layer* and *supervisory layer.* The base layer is implemented as a binary HM with symmetric weights and without self-feedback connections. Most of the base layer neurons represent the cost function and the constraints is the same way as in HM. These neurons are called *base neurons.* Additionally, in the base layer there exist some number of *hidden neurons*, which are not directly involved in representation of the problem - they can be used, for example, to provide external stimuli.

The supervisory layer is composed of *supervisory neurons*, which are connected to base neurons, but are not connected to hidden ones. There are also no connections among supervisory neurons. Each supervisory neuron except one is dedicated to one particular constraint. Additional supervisory neuron is devoted to the cost function. Weights between base neurons and supervisory ones are defined according to problem constraints. These weights are not altered during simulation process. The cost function is defined over the set of supervisory neurons in such a way that it reaches global minimum if and only if all constraints represented by these neurons are fulfilled. In each epoch, once the network settles down after minimization phase each supervisory neuron checks out whether the respective constraint is fulfilled. If it is not, the weights in the base layer are modified accordingly (with symmetry condition kept). Modification of weights is guided by the task of minimization of the *external* cost function (see Lee and Park, 1995, for formal mathematical description of the weights perturbation phase in D2NN).

7.2. Solving the NQP

The NQP of size n is represented by a square $n \times n$ matrix V with one-to-one correspondence between matrix elements and base neurons. Moreover, in base layer there exists one hidden neuron neu_{00}, responsible for providing external stimuli to all base neurons neu_{ij}, $i, j = 1, \ldots, n$. This hidden neuron is always ON. The input to a base neuron neu_{ij}, $i, j = 1, \ldots, n$, is defined as

$$u_{ij} = \sum^{n} \sum^{n} t_{ij,kl} v_{kl} + t_{ij,00} v_{00} \tag{49}$$

where $t_{ij,kl}$ is connection weight between base neurons neu_{kl} and neu_{ij}. Binary transfer function (13) is used for the internal input-output mapping in base neurons.

Supervisory layer is composed of $6n-2$ neurons each of which controls one NQP constraint: n row and n column constraints and $4n-2$ diagonal ones (see Definition 3):

$$\text{(row)} \quad S_k^r = \sum_{i=1}^{n} v_{ki}, \quad k = 1, \ldots, n, \tag{50}$$

$$\text{(column)} \quad S_k^c = \sum_{i=1}^{n} v_{ik}, \quad k = 1, \ldots, n, \tag{51}$$

$$\text{(diagonal)} \quad S_k^{d_1} = \sum_{j-i=k} v_{ij}, \quad k = -(n-1), \ldots, (n-1), \tag{52}$$

$$\text{(diagonal)} \quad S_k^{d_2} = \sum_{j+i=k} v_{ij}, \quad k = 2, \ldots, 2n. \tag{53}$$

Note that in case of the NQP there is no need for additional neuron that controls the cost function.

Based on (50)–(53) weights $w_{k,ij}^{con}$ between neuron neu_{ij} in base layer and neuron S_k^{con} in supervisory layer are set, where $con \in \{r, c, d_1, d_2\}$ denotes the type of constraint. For example, $w_{2,34}^r = 0$ since neuron neu_{34} does not belong to the second row. Similarly $w_{3,36}^{d_1} = 1$ since neuron neu_{36} belongs to a diagonal for which index difference between column and row is equal to 3.

The cost function is defined as a function of supervisory neurons in the following way:

$$F = \frac{1}{2}\Big(\sum_{k=1}^{n}(S_k^r - 1)^2 + \sum_{k=1}^{n}(S_k^c - 1)^2 + \sum_{k=-(n-1)}^{n-1} f^2(S_k^{d_1} - 1) + \sum_{k=2}^{2n} f^2(S_k^{d_2} - 1)\Big), \tag{54}$$

where

$$f(x) = \begin{cases} x & \text{for } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{55}$$

Note that $f(x-1) = 0$ for $x \leq 1$. Moreover, $F = 0$ if and only if all NQP constraints are fulfilled.

Computer simulations of D2NN confirmed its efficacy in solving small and medium size NQP. For n between 4 and 40, 100% convergence to solutions was reported (Lee and Park, 1995). The success of D2NN lies in its ability to escape from local minima of the energy surface. An important advantage of D2NN - though not actually used in case of the NQP - is its wide applicability, since

The weak point of D2NN is *high computational complexity* since in this model the search in n^2-dimensional binary state space is replaced by the search in continuous n^4-dimensional weight space. This makes application of D2NN to the NQP of large size an infeasible task.

7.3. Harmony theory neural network

An interesting approach to solving the NQP based on Harmony Theory is presented in Tambouratzis (1997). The Harmony Theory Neural Network (HTNN) for solving constraint satisfaction problems (Smolensky, 1986) is composed of two layers with sigmoidal neurons: the upper layer represents constraints and the lower one represents elements of the problem (in case of the NQP - chessboard squares). The only connections allowed are these between the layers, i.e. there are no connections within the same layer. Unlike in the Hopfield networks, in HTNN, instead of the explicitly stated energy function, the *consensus function of harmony* is implicitly defined. Each network state is accompanied by a harmony value, which measures the "goodness" of this state. States with more conflicts are assigned lower harmony values compared to those with smaller number of conflicts.

Network simulation is based on simulated annealing procedure, which starts off with adequately high initial temperature and lowers it gradually in the course of simulation. Two processes accompany this gradual temperature decrease: *sharpening of the gain* in sigmoidal activation function - which causes "gradual binarization" of the state space, and *magnification of discrepancies* of harmony values between states. As a result system's state space shrinks since states with low harmony values are not considered in the search process anymore. Changes of harmony values are governed by the threshold parameter k which has critical effect on selection of the optimal solution. Another important factor is the assignment of positive *strength values* to upper layer nodes, which represent relative importance of constraints related to these nodes (see Smolensky, 1986, for more details).

HTNN applied to solving the NQP is composed of n^2 neurons in the lower layer and $4(n^2-1)$ neurons in the upper one (Tambouratzis, 1997). Each lower layer neuron represents a particular chessboard square and each upper layer neuron encodes one constraint corresponding to a particular chessboard square: n^2 row inhibition constraints (one per square), n^2 column inhibition constraints (one per square), n^2-2 left-diagonal constraints (one per square except the two corner squares) and similarly n^2-2 right-diagonal constraints (as before).

As reported in Tambouratzis (1997) with appropriate choice of parameter k, HTNN is effective in solving the NQP of small size, i.e. $4 \leq n \leq 32$. Evaluation of HTNN effectiveness for larger problem instances and its application to other combinatorial optimization problems (e.g. TSP) are interesting open research

8. Examples of other Hopfield-based approaches

This section briefly summarizes a few more examples of neural approaches to solving the NQP: Chaotic Models and Strictly Digital Neural Network.

8.1. Chaotic optimization

Several extensions of HMs based on deterministic chaos have been proposed in the literature. One well known example is Chaotic Neural Network (CNN) (Aihara, Takabe and Toyoda, 1990) in which network's dynamics is defined in the following way:

$$v_i(t) = \frac{1}{1 + e^{-\alpha u_i(t)}}, \tag{56}$$

$$u_i(t+1) = ku_i(t) + s\Big(\sum_{j=1}^{N} t_{ij}v_j(t) + I_i\Big) - z_i v_i(t). \tag{57}$$

In (56)–(57) $v_i, u_i, t_{ij}, I_i, \alpha$ have the same meaning as in HM, $s > 0$ and $0 \leq k \leq 1$ are scalar coefficients, and $z_i > 0$ is a weight of self-feedback connection.

Chaos in CNN is generated by accumulation of negative self-feedbacks $-z_i v_i(t)$ over time. In effect the network breaks its monotonic convergence and is able to escape from local minima of the energy function.

CNN was later modified to Transiently Chaotic Neural Network (TCNN) (Chen and Aihara, 1995) in which chaotic term is controlled by the simulated annealing procedure and consequently the amount of chaos in the system is gradually decreased to zero. Application of the simulated annealing procedure eliminates the critical problem in CNN concerning the arbitrary choice of the moment when the chaotic search process should be stopped. In order to force the convergence of TCNN (by gradual decrease of the amount of chaos in the network) the input potential to neuron neu_i is defined by the following equation:

$$u_i(t+1) = ku_i(t) + s\Big(\sum_{j=1}^{N} t_{ij}v_j(t) + I_i\Big) - z_i(t)(v_i(t) - I_0), \tag{58}$$

where

$$z_i(t+1) = (1-\beta)z_i(t) \tag{59}$$

and $t_{ij} = t_{ji}$, $t_{ii} = 0$, $i, j = 1, \ldots, N$, $z_i(t) \geq 0$, $I_0 > 0$, $1 \geq \beta \geq 0$. The output potential $v_i(t)$ of neuron neu_i in TCNN is defined by eq. (56).

The term $z_i v_i(t)$ from eq. (57) which implements the self-feedback mechanism is in TCNN (eq. (58)) replaced by $z_i(t)(v_i(t) - I_0)$. Parameter $z_i(t)$ is interpreted as the *temperature* of the model in the process of chaotic simulated annealing. Consequently, eq. (59) describes a very fast, exponential annealing

Since $I_0 > 0$ and $z_i(t)$ decreases in time, TCNN after the initial *bifurcation* period finally stabilizes and resembles more and more the Hopfield model. At temperatures close to zero TCNN is equivalent to the Hopfield model. Therefore, unlike CNN, TCNN is guaranteed to converge (analogously to the HM).

TCNN model is well known as the efficient tool for solving the Travelling Salesman Problem. In asynchronous simulations (according to randomly selected permutation of neurons) the optimal path for the 10-city TSP was found in 100% of 5 000 tests (Chen and Aihara, 1995). Similar results were obtained in our simulations. TCNN also appeared to be very efficient in fully synchronous simulations with the average tour length equal to 100.3% of the optimal tour length (Mańdziuk, 2000b).

We have also tried to apply the TCNN model to solving the NQP of size $n = 8$ and $n = 16$ with energy function (27). Quite surprisingly the results were not as good as expected. The main problem was the appropriate choice of system's coefficients, since the model seemed to be very sensitive to that selection. Actually we were unable to select the set of coefficients for the asynchronous update mode that would provide satisfactory results. After a certain number of trials we found out that the best results for the NQP are obtained in synchronous mode with the *synchronity factor* (sf) between 0.85 and 0.95[6].

The best convergence to solutions rate for $n = 8$ was equal to 91% and was obtained with the following set of parameters:

$$\begin{aligned} &A = 2, \quad B = 2, \quad C = 1, \quad \sigma = 0.9, \\ &k = 0.9, \quad \alpha = 250, \quad I_0 = 0.65, \quad sf = 0.85, \\ &z(0) = 0.08, \quad u(0) = 0.5, \quad s = 0.015, \quad \beta = 10^{-4}. \end{aligned} \tag{60}$$

For $n = 16$ the best rate was also equal to 91% and the parameter set was as follows:

$$\begin{aligned} &A = 2, \quad B = 2, \quad C = 1, \quad \sigma = 2.2, \\ &k = 0.9, \quad \alpha = 250, \quad I_0 = 0.65, \quad sf = 0.95, \\ &z(0) = 0.08, \quad u(0) = 0.5, \quad s = 0.015, \quad \beta = 10^{-4}. \end{aligned} \tag{61}$$

Several trials with faster annealing schedule, i.e. with smaller value of $\beta = 10^{-3}$ resulted in 87% convergence to solutions for $n = 8$ (with $\sigma = 2.2$ and other parameters as in (60)) and 85% convergence for $n = 16$ (with $\sigma = 2.5$ and other parameters as in (61)).

Another approach to building Chaotic Neural Networks based on adding deterministic chaos to the system was proposed in Nozawa (1992). An interesting modification to Nozawa (1992) was reported in Ohta (1999), where it is proposed that the network autonomously magnify its self-feedback connections (and consequently increase the amount of chaos in the system) in case it

[6]The *synchronity factor* is defined as the fraction of randomly selected neurons that are

is trapped in a local minimum. The model was applied to the NQP of sizes 50, 100 and 200 with visible improvement of solution rate compared to Nozawa (1992) – from 84.3% to 98.8%, in case of $n = 200$.

8.2. Digital models

Binary Hopfield-type neural networks are especially interesting in the context of digital hardware implementation. One of such implementations is SDNN - *Strictly Digital Neural Network* (Nakagawa, Page and Tagliarini, 1989). SDNN is designed based on "k-out-of-n" principle (Page and Tagliarini, 1988), which can be summarized as follows: suppose that a constrained satisfaction problem is represented by m multiple-sets (one set per constraint) $S_r, r = 1, \ldots, m$ each composed of n_r neurons and that in each set $S_r, r = 1, \ldots, m$, k_r neurons must be ON if the constraint is satisfied. The energy term corresponding to the r-th constraint can therefore be expressed in the following way:

$$E_r = \Big(\sum_{i=1}^{n_r} v_i - k_r\Big)^2 + \sum_{i=1}^{n_r} v_i(1 - v_i), \tag{62}$$

where the second term forces binarization of the network states. Eq. (62) can be rewritten in the form

$$E_r = -\frac{1}{2}\sum_{i=1}^{n_r}\sum_{j=1, j\neq i}^{n_r} -2v_iv_j - \sum_{i=1}^{n_r} v_i(2k_r - 1) + k_r^2, \tag{63}$$

which has the form of Hopfield's energy (14) (with $t_{ij} = -2, i \neq j$, $I_i = 2k_r - 1$) in case scalar k_r^2 is neglected.

The above approach allows for a systematic design of a Hopfield-type binary network *with strictly digital weights*. Input u_i to neuron neu_i in SDNN is defined as:

$$u_i = \sum_{r=1}^{m_i}\Big(\sum_{j\neq i}^{n} t_{r_{ij}} v_j + I_{r_i}\Big), \tag{64}$$

where m_i is the number of sets S_r to which neu_i belongs, $t_{r_{ij}}$ is weight of connection between neu_j and neu_i in the r-th set S_r, and I_{r_i} is an external input to neuron neu_i generated in the set S_r.

It is easy to show that, if neuron neu_i satisfies "k-out-of-n" rule, then it must be in one of only two feasible states u_i^{ON} or u_i^{OFF} corresponding to being ON or OFF, respectively, where

$$u_i^{ON} = m_i \quad \text{and} \quad u_i^{OFF} = -m_i. \tag{65}$$

The above construction can be directly implemented in hardware allowing efficient, parallel computation of minimal energy states (Nakagawa and Kita-

Application of SDNN to the NQP is based on appropriate combination of several "1-out-of-n" rules, according to the row, column and diagonal constraints. The fact that in solution states input to each neuron is restricted to only two values (see eq. (65)) allows a great speed-up in hardware simulations of SDNN compared to the classical Hopfield's approach. According to the tests reported in Nakagawa and Kitagawa (1991) for the problem sizes of up to 3 000 experimental computational complexity is $0(1)$ in parallel mode and $0(n^2)$ in sequential mode, with 100% convergence to solutions.

Another hardware oriented approach based on *systolic arrays* and its application to solving the NQP is discussed in Funabiki, Kurokawa and Ohta (2002), in this issue.

9. Conclusions

Neural networks are well suited to solving certain types of optimization problems. Among neural approaches there are two basic ones: deformable template matching and gradient Hopfield method. The latter can be effectively applied to solving the N-Queens Problem, which is one of classical benchmarks in combinatorial optimization domain.

The key advantage of the Hopfield networks (binary, discrete and continuous) is their simple and effective implementation. Another good point is generality of Hopfield's approach - the only requirement is quadratic formulation (24)-(25) of the problem being solved.

On the other hand the main limitation of the Hopfield networks is high possibility of being trapped in a local minimum since optimization method implemented in these models is purely gradient. Another weak point is related to high sensitivity of Hopfield models to the choice of internal parameters (energy coefficients).

Various modifications to original Hopfield's formulation have been proposed in the literature. These improvements can generally be divided into four groups:

- modifications of energy form (or equivalently more efficient problem representations),
- deterministic modifications (e.g. negative self-feedback connections, other transfer functions, neurons with hysteresis),
- simulated annealing methods (stochastic, chaotic and mean field approaches),
- hybrid methods (combining HMs with non-gradient optimization methods).

In this paper several examples of the above methods are presented and their effectiveness tested based on the N-Queens Problem. Advantages and limita-

Numerical results indicate that each of the modified Hopfield models can be effectively used in solving the NQP. Convergence to solutions rate of these methods is very high - usually close to 100%. Experimental time requirements are generally low - polynomial in most cases.

On the other hand it is worth noting that for a large group of combinatorial optimization problems (e.g. the Travelling Salesman Problem) neural networks are generally less efficient than heuristic algorithms. The main reason for such efficiency difference lies in the nature of neural optimization methods which represent a very general, non-dedicated approach as opposed to highly dedicated heuristic algorithms.

References

AARTS, E.H.L. (1989) *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* John Wiley & Sons, New York.

AARTS, E.H.L. and LAARHOVEN, P.J.M. (1987) *Simulated Annealing: Theory and Applications.* Dordrecht, The Netherlands.

AIHARA, K., TAKABE, T. and TOYODA, M. (1990) Chaotic neural networks. *Phys. Lett. A*, **144** (6–7), 333–340.

AKIYAMA, Y., YAMASHITA, A., KAJURA, M. and AISO, H. (1989) Combinatorial Optimization with Gaussian Machines. *Proceedings of International Joint Conference on Neural Networks (IJCNN'89)*, **I**, 533–540.

AKIYAMA, Y., YAMASHIRA, A., KAJIURA, M., ANZAI, Y. and AISO, H. (1991) The Gaussian Machine: A Stochastic Neural Network for Solving Assignment Problems. *Journal of Neural Network Computing*, 43–51.

ANGENIOL, B., VAUBOIS, G. and LE TEXIER, J.Y. (1988) Self-Organizing Feature Maps and the Travelling Salesman Problem. *Neural Networks*, **1**, 289–293.

BERNHARDSSON, B. (1991) Explicit Solutions to the N-Queens Problem for all N. *SIGART Bulletin*, **2** (2), 7.

CERNY, V. (1985) Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, **45** (1), 41–51.

CHEN, L. and AIHARA, K. (1995) Chaotic Simulated Annealing by a Neural Network Model with Transient Chaos. *Neural Networks*, **8** (6), 915–930.

DURBIN, R. and WILLSHAW, D.J. (1987) An Analogue Approach to the Traveling Salesman Problem using an Elastic Net Method. *Nature*, **326**, 689–691.

FUNABIKI, N., TAKENAKA, Y. and NISHIKAWA, S. (1997) A maximum neural network approach for N-queens problems. *Biological Cybernetics*, **76**, 251–255.

FUNABIKI, N., KUROKAWA, T. and OHTA, M. (2002) Binary Neural Networks to N-Queens Problem and their VLSI Implementations. *Control and Cy-*

GIDAS, B. (1986) The Langevin equation as a global minimization algorithm. In: E. Bienenstock et al. eds., *Disordered Systems and Biological Organizations.*, Springer–Verlag, Berlin/Heidelberg, 321–326.

HINTON, G.E., SEJNOWSKI, T.J. and ACKLEY, D.H. (1984) Boltzmann Machines: Constraint Satisfaction Networks that Learn. *Technical Report CMU-CS-84-119*, Department of Computer Science, Carnegie Mellon University, Pittsburgh, USA.

HOFFMAN, E.J., LOESSI, J.C. and MOORE, R.C. (1969) Constructions for the Solution of the m Queens Problem. *Mathematics Magazine*, 66–72.

HOPFIELD, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science USA*, **79**, 2554–2558.

HOPFIELD, J.J. (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science USA*, **81**, 3088–3092.

HOPFIELD, J.J. and TANK, D.W. (1985) "Neural" Computation of Decisions in Optimization Problems. *Biological Cybernetics*, **52**, 141–152.

HOPFIELD, J.J. and TANK, D.W. (1986) Computing with Neural Circuits: A Model. *Science*, **233**, 625–633.

KIRKPATRIC, S., GELAT JR., C.D. and VECCHI, M.P. (1983) Optimization by Simulated Annealing. *Science*, **220**, 671–680.

LEE, S. and PARK, L. (1995) Dual-mode dynamics neural network for combinatorial optimization. *Neurocomputing*, **8** (3), 283–304.

LEVY, B. C. and ADAMS, M.B. (1987) Global optimization with stochastic neural networks. *Proceedings IEEE Conf. on Neural Networks, San Diego, USA*, **III**, 681–689.

LI, S. Z. (1996) Improving convergence and solution quality of Hopfield-type neural networks with augmented Lagrange multipliers. *IEEE Transactions on Neural Networks*, **7** (6), 1507–1516.

MAŃDZIUK, J. (2000a) Optimization with the Hopfield network based on correlated noises: experimental approach. *Neurocomputing*, **30** (1-4), 301–321.

MAŃDZIUK, J. (2000b) *Hopfield type neural networks. Theory and applications.* Akademicka Oficyna Wydawnicza EXIT, (in Polish).

MAŃDZIUK, J. (1995) Solving the N-Queens Problem with a binary Hopfield-type network. Synchronous and asynchronous model. *Biological Cybernetics*, **72** (5), 439–446.

MAŃDZIUK, J. and MACUKOW, B. (1992) A neural network designed to solve the N-Queens Problem. *Biological Cybernetics*, **66**, 375–379.

MCCULLOCH, W.S. and PITTS, W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.

NAKAGAWA, T., PAGE, E.W. and TAGLIARINI, G.A. (1989) SDNN: A Computation Model for Strictly Digital Neural Networks and its Applications. *Proceedings 5th AAAIC'89*, Dayton, OH, USA.

with Strictly Digital Neural Networks for Combinatorial Optimization in Large Scale N-Queen Problem. In: Kohonen, T., Makisara, K., Simula, O. and Kangas, J., eds., *Artificial Neural Networks*, 1181–1184.

NOZAWA, H. (1992) A neural network model as a globally coupled map and applications based on chaos. *Chaos*, **2** (3), 377–386.

OHTA, M. (1999) On the Self-Feedback Controlled Chaotic Neural Network and its Application to N-Queen Problem. *Proceedings Iinternation Joint Conference on Neural Networks (IJCNN'99)*, 4 pages (CD-ROM).

PAGE, E.W. and TAGLIARINI, G.A. (1988) Algorithm Development for Neural Networks. *Proceedings SPIE*, **880**, 11–18.

RUSSEL, S.J. and NORVIG, P. (1995) *Artificial Intelligence: A Modern Approach*. Prentice Hall, Enlewood Cliffs, New Jersey.

SMITH, K.A. (1996) An Argument for Abandoning the Traveling Salesman Problem as a Neural-Network Benchmark. *IEEE Transactions on Neural Networks*, **7** (6), 1542–1544.

SMITH, K.A., POTVIN, J-Y. and KWOK, T. (2002) Neural network models for combinatorial optimization: deterministic, stochastic and chaotic approaches. *Control and Cybernetics*, this issue.

SMITH, M.J.S. and PORTMAN, C.L. (1989) Practical Design and Analysis of a Simple "Neural" Optimization Circuit. *IEEE Transactions on Circuits and Systems*, **36** (1), 42–50.

SMOLENSKY, P. (1986) Information processing in dynamical systems: Foundations of harmony theory. In: Rumelhart, D.E. and McClelland, J.L., eds., *Parallel Distributed Processing: Foundations, vol. 1*, MIT Press, Cambridge, MA, 194–281.

SOSIC, R. and GU, J. (1990) A Polynomial Time Algorithm for the N-Queens Problem. *SIGART Bulletin*, **1** (3), 7–11.

SOSIC, R. and GU, J. (1991a) 3,000,000 Queens in Less Than One Minute. *SIGART Bulletin*, **2** (2), 22–24.

SOSIC, R. and GU, J. (1991b) Fast Search Algorithms for the N-Queens Problem. *IEEE Transactions on Systems, Man, and Cybernetics*, **21** (6), 1572–1576.

TAKEFUJI, Y. and LEE, K.C. (1992) Neural network computing for knight's tour problems. *Neurocomputing*, **4**, 249–254.

TAMBOURATZIS, T. (1997) A Simulated Annealing Artificial Neural Network Implementation of the N-Queens Problem. *International Journal of Intelligent Systems*, **12**, 739–751.

TANK, D.W. and HOPFIELD, J.J. (1986) Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit. *IEEE Transactions on Circuits and Systems*, **CAS-33** (5), 533–541.

TANK, D.W. and HOPFIELD, J.J. (1987) Collective Computation in Neuronlike Circuits. *Scientific American*, December, 62–70.

[illegible] (1999) On Chaotic Simulated Annealing. *IEEE*

Transactions on Neural Networks, **9** (4), 716–718.

WIRTH, N. (1976) *Algorithms + Data Structures = Programs.* Prentice-Hall, Inc.

WONG, E. (1991) Stochastic neural networks. *Algorithmica*, **6**, 466–478.

ZHANG, S. and CONSTANTINIDES, A.G. (1992) Lagrange programming neural networks. *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, **39** (7), 441–452.