Boltzmann Machines and Their Applications

Emile H.L. Aarts and Jan H.M. Korst

Philips Research Laboratories
P.O. Box 80.000, 5600 JA Eindhoven, the Netherlands

Abstract

In this paper we present a formal model of the Boltzmann machine and a discussion of two different applications of the model, viz. (i) solving combinatorial optimization problems and (ii) carrying out learning tasks. Numerical results of computer simulations are presented to demonstrate the characteristic features of the Boltzmann machine.

Keywords: Boltzmann machines, simulated annealing, combinatorial optimization, learning.

1 Introduction

Many researchers believe that massive parallelism rather than raw speed of individual processors may provide the computational power required to carry out the increasingly complex calculations imposed by e.g. combinatorial optimization [3,13,15] and artificial intelligence [4,6,7]. A revolutionary development in the field of parallel computer architectures is based on the so-called connectionist models [5,6]. These models incorporate massive parallelism and the assumption that information can be represented by the strengths of the connections between individual computing elements.

The Boltzmann machine, introduced by Hinton *et al.* [4,6,11], is a novel approach to connectionist models using a distributed knowledge representation and a massively parallel network of simple stochastic computing elements. The computing elements are considered as logic units having two discrete states, viz. 'on' or 'off'. The units are connected to eachother. With each connection a connection strength is associated representing a quantitative measure of the hypothesis that the two connected units are both 'on'. A consensus function assigns to a configuration of the Boltzmann machine (determined by the states of the individual units) a real number which is a quantitative measure of the amount of consensus in the Boltzmann machine with respect to the set of underlying hypotheses. The state of an individual unit (if not externally forced) is determined by a stochastic function of the states of the units it is connected to and the associated connection strengths. Maximization of the consensus function corresponds to maximization of the amount of information contained within the Boltzmann machine.

Interest in the Boltzmann machine extends over a number of disciplines, i.e. future computer architectures [6,23], knowledge representation in intelligent systems [2,6,23,24], modelling of neural behaviour of the brain [11,21], and research on applications, e.g. pattern recognition [10,11,18,20,22] and combinatorial optimization [3,15]. Especially the research on recognition and synthesis of speech with a Boltzmann machine has recently shown a number of interesting achievements [18,20,22].

In this paper we discuss a formal model and two different applications of the Boltzmann machine. In §2 we present a graph-theoretical model of the structure of the Boltzmann machine and a stochastic state-transition mechanism that can be applied to maximize the consensus function. In §3 a model of the Boltzmann machine is presented that can be applied to solve combinatorial optimization problems. For two distinct 0-1 integer formulations of the traveling salesman problem (as an assignment problem and as a quadratic assignment problem) it is shown that near-optimal solutions can be obtained. This is done by mapping the corresponding 0-1 variables onto the computing elements of the Boltzmann machine and by setting the connection strengths so as to represent the cost function and the constraints of the optimization problem. In §4 we discuss the ability of a Boltzmann machine to learn. Learning takes place by examples. To this end a subset of the computing elements is selected. In the learning phase the connection strengths are adjusted such that the Boltzmann machine, when left completely free to set its states, tends to show on that subset of computing elements a number of patterns (given by the examples) with an a priory given probability distribution. After completion of the learning phase the Boltzmann machine is able to reproduce complete patterns if part of them are clamped into the given subset of computing elements by maximizing the consensus function. In this way a Boltzmann machine can not only reproduce given examples (memory) but also has associative and restricted inductive capabilities. The essential difference between the applications discussed in §3 and §4 is given by the fact that in the first application (solving combinatorial optimization problems) the Boltzmann machine adjusts the states of the units to a given set of connection strengths that is fixed in time, whereas in the second application (learning) the Boltzmann machine adjusts the values of the connection strengths to a given number of examples which fix the states of the units they are clamped into.

2 A Formal Model of the Boltzmann Machine

The model of the Boltzmann machine originates from the theory of neural networks as given by Hopfield [12] and was introduced by Hinton *et al.* [4,6,11] to cope with learning tasks. In this section we present a formal model of the Boltzmann machine based on a graph-theoretical formulation of the architectural structure of the network (§2.1), a description of the stochastic optimization algorithm that is applied to maximize the consensus function (§2.2) and a discussion of the implementation of parallelism (§2.3).

2.1 Structural Description

A Boltzmann machine consists of a (large) number of logical units, say N, that can be represented by an undirected graph $G = (V, E)$, where $V = \{v_0, \cdots, v_{N-1}\}$ denotes the set of vertices corresponding to the logical units, and $E \subseteq V \times V$ the set of edges corresponding to the connections between the units [2]. An edge $(v_i, v_j) \in E$ connects the vertices v_i and v_j. The set of edges includes all loops, i.e. $\{(v_i, v_i) \mid v_i \in V\} \subset E$. With each vertex v_i a number is associated denoting the state of the corresponding i-th logical unit, i.e. 0 or 1 corresponding to 'off' or 'on', respectively.
A configuration k of the Boltzmann machine is uniquely defined by the states of all individual vertices. The state of vertex v_i in configuration k is denoted by $r_k(v_i)$. The configuration space $\mathcal{R}$ denotes the set of all possible configurations ($|\mathcal{R}| = 2^N$).
An edge (v_i, v_j) is defined to be activated in a given configuration k if $r_k(v_i)\ r_k(v_j) = 1$. With each edge (v_i, v_j) a connection strength $s(v_i, v_j) \in \mathbf{R}$ is associated determining the strength of the connection between the vertices v_i and v_j. The connection strength $s(v_i, v_j)$ is considered as a quantitative measure of the desirability that the edge (v_i, v_j) is activated. If $s(v_i, v_j) \gg 0$ then it is considered very desirable that (v_i, v_j) is activated, if $s(v_i, v_j) \ll 0$ it is considered very undesirable. The consensus C_k of a configuration k denotes the overall desirability of all the activated edges in

the given configuration and is defined as

$$C_k = \sum_{(v_i,v_j)\in E} s(v_i,v_j)\, r_k(v_i)\, r_k(v_j). \tag{1}$$

Next, a neighbourhood $\mathcal{R}_k \subset \mathcal{R}$ is defined as the set of configurations that can be obtained from a given configuration k by changing the state ($0 \rightarrow 1$ or vice versa) of one of the vertices. Thus, a neighbouring configuration $k^{(i)} \in \mathcal{R}_k$ obtained by changing the state of vertex v_i is given by

$$r_{k^{(i)}}(v_j) = \begin{cases} r_k(v_j) & j \neq i \\ 1 - r_k(v_j) & j = i. \end{cases} \tag{2}$$

The corresponding difference in the consensus $\Delta C_{kk^{(i)}}$ is given by

$$\Delta C_{kk^{(i)}} = C_{k^{(i)}} - C_k = (1 - 2\, r_k(v_i))(\sum_{(v_i,v_j)\in E_{v_i}} s(v_i,v_j)\, r_k(v_j) + s(v_i,v_i)), \tag{3}$$

where E_{v_i} denotes the set of edges incident with vertex v_i. From eq. 3 it is apparent that the effect on the consensus by changing the state of vertex v_i is completely determined by the states of the vertices v_j that are connected to v_i and by the corresponding connection strengths. Consequently, the differences in consensus $\Delta C_{kk^{(i)}}$ can be computed locally, thus allowing parallel execution.

2.2 Consensus Maximization

Maximization of the consensus function is done by means of a parallel implementation of the simulated annealing [14] (or statistical cooling [1]) algorithm. Simulated annealing is a general randomization technique to solve combinatorial optimization problems. It is based on the ergodic theory of Markov chains (for a detailed description the reader is referred to [1,17]). In a similar way the concepts of Markov chains can be used to describe the transitions among the configurations within the Boltzmann machine required to maximize the consensus function of eq. 1. In this section we describe the sequential implementation of the simulated annealing algorithm, i.e. vertices are allowed to change their states one at a time. In the next section it is shown how this procedure can be extended to parallel implementations.
A Markov chain consists of a sequence of trials where the outcome of a trial depends probabilistically on the outcome of the previous trial. In the case of the Boltzmann machine a trial consists of two steps. Given a configuration k then firstly, a neighbouring configuration $k^{(i)}$ is generated and secondly, it is evaluated whether or not $k^{(i)}$ is accepted. If it is accepted the outcome of the trial is $k^{(i)}$, otherwise it is k. With this process a transition probability T_{kl} can be associated which defines the probability of obtaining configuration l as an outcome of a given trial provided the outcome of the previous trial is configuration k. Here, the transition probabilities T_{kl} are defined as follows:

$$T_{kl}(c) = \begin{cases} T_{kk^{(i)}}(c) & l = k^{(i)} \\ 1 - \sum_{j=0}^{N-1} T_{kk^{(j)}}(c) & l = k, \end{cases} \tag{4}$$

and

$$T_{kk^{(i)}}(c) = P_{kk^{(i)}}(c) B_{kk^{(i)}}(c), \tag{5}$$

where for a given value of the control parameter c ($c \in \mathbf{R}^+$), $P_{kk^{(i)}}(c)$ denotes the generation probability, i.e. the probability of changing the state of vertex v_i given the configuration k, and $B_{kk^{(i)}}(c)$ denotes the acceptance probability, i.e. the probability of accepting the state transition of vertex v_i given the configuration k. In our applications the generation probability is chosen independent of c and k, and uniformly over all the N vertices, i.e.

$$P_{kk^{(i)}}(c) = N^{-1}. \tag{6}$$

The acceptance probability is chosen as

$$B_{kk^{(i)}}(c) = \frac{1}{1 + e^{-\Delta C_{kk^{(i)}}/c}}, \tag{7}$$

where $\Delta C_{kk^{(i)}}$ is given by eq. 3. From eq. 7 it is evident that the response of a vertex to a proposed change of its state is determined by its local connections (it only depends on $\Delta C_{kk^{(i)}}$). Maximization of the consensus function takes place by starting off at an initial value of c with a (randomly) chosen initial configuration and subsequently generating a sequence of Markov chains according to eqs. 4-7. The value of c is lowered slowly in between subsequent Markov chains until it approaches 0. The Markov chains are generated by continuously trying to change the states of the individual vertices and applying the acceptance criterion of eq. 7. Eventually as c approaches 0 state transitions become more and more infrequent and finally the Boltzmann machine stabilizes in a configuration which is taken as the final solution.
It can be shown that for sufficiently long Markov chains and a fixed value of c equilibrium is achieved [1,17], in the sense that there exists a unique equilibrium vector $\pi(c) \in (0,1)^{|\mathcal{R}|}$ whose components $\pi_k(c)$ determine the probability that the Boltzmann machine occurs in configuration k when equilibrium is achieved. The components take the form [1]

$$\pi_k(c) = \pi_0(c) \exp\left(\frac{C_k - C_{max}}{c}\right), \tag{8}$$

where C_{max} denotes the maximal value of the consensus of the Boltzmann machine for a given set of connection strengths, and π_0 a normalization constant such that $\sum_{k\in\mathcal{R}} \pi_k(c) = 1$. From eq. 8 it is apparent that configurations corresponding to a higher consensus have a larger probability of occurring than configurations with a lower value of the consensus. This result plays an important role in the construction of the learning algorithm presented in §4. Furthermore, it can be shown that under the same asymptoticity conditions and for c approaching to 0 the final solution obtained by the simulated annealing algorithm is an optimal solution [1,17]. In practical implementations of the algorithm, however, the asymptoticity conditions are never attained and thus convergence to an optimal solution is not guaranteed anymore, i.e. the algorithm obtains a local optimum of the consensus function which is close (or even equal) to the maximal consensus. Consequently the algorithm is an approximation algorithm.
The quality of the final solution obtained by the algorithm is determined by the convergence of the algorithm which is governed by a set of parameters, known as the cooling schedule [17]. The parameters are: the start value of c, a decrement rule to lower the value of c, the length of the individual Markov chains and a stop criterion that justifies termination of the algorithm. The computer simulations presented in this paper (§3.3 and §4.3) are carried out with different cooling schedules and for the detailed choices of the parameters the reader is referred to [3] and [2], respectively.

2.3 Parallelism

As mentioned in the previous subsections vertices only require local information to calculate a (possible) state transition. Thus the generation of a Markov chain (§2.2) can be done in parallel. To discuss this subject we distinguish between synchronous and asynchronous parallelism.

Synchronous parallelism
A vertex only needs to know the states of its neighbouring vertices (i.e. the vertices to which it is connected) and the corresponding connection strengths to calculate a state transition. Consequently, vertices that are not connected to each other, are allowed to change their state in parallel.

Synchronous parallelism can be obtained by partitioning the set of vertices V into disjoint subsets $\{V_1, \cdots, V_m\}$, such that vertices in the same subset are not connected to each other. For synchronization reasons a clocking scheme is introduced. In each clock cycle a subset V_i is chosen (randomly) and the vertices of V_i are allowed to change their states simultaneously. For optimal exploitation of parallelism the number of subsets m should be as small as possible. The problem of finding such a partition of minimal size is known as the graph coloring problem. Due to the regular structure of the Boltzmann machine for most applications (see for instance §3 and §4) the problem of finding such a partition is easy.
The amount of parallelism that can be obtained in this way strongly depends on the connection pattern in a Boltzmann machine and it will be relatively small if the amount of connectivity is large. For instance in the case of the implementation of the traveling salesman problem with n cities on a Boltzmann machine (§3) a minimal partition of the n^2 vertices contains $2n$ subsets and consequently the efficiency compared to full parallelism equals $\frac{1}{2n}$ which is small for large n. Furthermore, using this mode of parallelism requires some form of global synchronization of the vertices, which one would like to avoid.

Asynchronous parallelism
Asynchronous parallelism is based on the assumption that *all* vertices can change their states in parallel. This may introduce errors in the calculation of the difference in consensus. If two neighbouring vertices v_i and v_j change their states in parallel, this may lead to an activation or deactivation of the connection (v_i, v_j) which is not accounted for in the evaluation of the differences in consensus $\Delta C_{kk(i)}$ and $\Delta C_{kk(j)}$. For example, suppose that both vertices v_i and v_j are 'off' and suppose that they both accept a state transition to 'on' (based on the information that they are both 'off'), then connection (v_i, v_j) is activated, whereas it is not accounted for in the calculation of the difference in consensus and therefore not in the acceptance criterion of eq. 7 (evidently the same holds for state transitions in the reverse order). This may lead to a decrement of the consensus, despite the fact that both vertices anticipated an increase of the consensus. However, the probability that these unaccounted (de-)activations occur, decreases during the optimization process. From the definition of the simulated annealing algorithm it can be shown [1,17,14] that the probability of a vertex to change its state decreases as the value of the control parameter decreases and eventually it approaches 0 as c approaches 0. Thus the probability that in the course of the optimization process two neighbouring vertices change their state simultaneously is negligible and the states obtained by erroneously accepted state transitions will be corrected. In this way asynchronous parallelism can be successfully exploited. Moreover, synchronization is no longer necessary.
The computer simulations presented in §3 and §4 are based on asynchronous parallelism and from the numerical results it is concluded that efficiencies up to 95 % can be obtained.

3 Solving Combinatorial Optimization Problems with a Boltzmann Machine

In [15] we show that many well-known combinatorial optimization problems can be implemented on a Boltzmann machine. Our approach is based on the observation that the structure of many combinatorial optimization problems can be directly mapped onto the structure of a Boltzmann machine by choosing the right connections and that the corresponding cost function can be transformed into the consensus function by choosing the right connection strengths. To illustrate this we briefly discuss hereinafter two approaches to implement the traveling salesman problem (TSP) on a Boltzmann machine. For a more detailed description the reader is referred to [3].

Our approaches are based on two different formulations of the TSP, viz. as an assignment problem and as a quadratic assignment problem. We conjecture that the approaches can be generalized to the complete class of assignment and quadratic assignment problems given by definition 2 and 3 of the next subsection.

3.1 The Traveling Salesman Problem

Among all existing combinatorial optimization problems the TSP is probably the best known. Many problems can be reduced to the TSP and for this reason the TSP serves as a good demonstrator to show the feasibility of implementing combinatorial optimization problems on a Boltzmann machine. We will give three possible formulations of the TSP [8].

Definition 1.
Let n be the number of cities and $D = (d_{ij})$ the distance matrix whose entries d_{ij} denote the length of the shortest path from city i to j, then the TSP is defined as the problem of finding a tour of minimal length visiting each of the n cities exactly once.

□

To construct a model of a Boltzmann machine that embodies the TSP it is useful to rewrite the TSP as a 0-1 integer problem. This can be done in two different ways: as an assignment problem (AP) and as a quadratic assignment problem (QAP).

Definition 2. (AP)
Let x_{ij} be a 0-1 variable indicating whether or not the tour goes directly from city i to city j or vice versa and d_{ij} the corresponding distance then the TSP can be formulated in terms of an AP as

minimize

$$\sum_{i,j=0}^{n-1} d_{ij} x_{ij} \tag{9}$$

subject to

$$\sum_{i=0}^{n-1} x_{ij} = 1 \quad (j = 0, \ldots, n-1), \tag{10}$$

$$\sum_{j=0}^{n-1} x_{ij} = 1 \quad (i = 0, \ldots, n-1), \tag{11}$$

$$X = (x_{ij}) \text{ is irreducible (no sub tours).} \tag{12}$$

□

Definition 3. (QAP)
Let x_{ip} and x_{jq} be 0-1 variables indicating whether or not the tour visits city i at the p^{th} position and city j at the q^{th} position in the tour and d_{ij} the corresponding distance then the TSP can be formulated in terms of a QAP as

minimize

$$\sum_{i,j,p,q=0}^{n-1} a_{ijpq} x_{ip} x_{jq} \tag{13}$$

subject to

$$\sum_{i=0}^{n-1} x_{ij} = 1 \quad (j = 0, \ldots, n-1), \tag{14}$$

$$\sum_{j=0}^{n-1} x_{ij} = 1 \quad (i = 0, \ldots, n-1), \tag{15}$$

$$a_{ijpq} = d_{ij} t_{pq}, \tag{16}$$

where $T = (t_{pq})$ is a cyclic permutation matrix whose entries are given by

$$t_{pq} = \begin{cases} 1 & \text{if } q = (p+1) \bmod n \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

□

3.2 Implementing the TSP on a Boltzmann Machine

To implement the TSP on a Boltzmann machine we introduce an architecture corresponding to a grid of $n \times n$ vertices ($N = n^2$). The 0-1 variables x_{ij} correspond to the states of the vertices (see also figure 1). So, each configuration of the Boltzmann machine uniquely represents a value assignment to the 0-1 variables (including assignments not obeying the constraints given by eqs. 10-12 or eqs. 14-17). Vertices are mutually connected according to a given connection pattern using $O(n^3)$ edges. By assigning appropriate connection strengths to the edges it can be shown that the consensus function defined on the global configurations of the Boltzmann machine by eq. 1 has the following properties [3]:

(P1) a configuration corresponds to a local maximum of the consensus function if and only if it corresponds to a tour.

(P2) the shorter the tourlength of a given tour, the higher the consensus of the corresponding configuration.

From these two properties it follows that the consensus is maximal for configurations corresponding to an optimal tour and that near-optimal (with respect to the consensus) configurations correspond to near-optimal tours [3].

The AP and QAP formulations of the TSP give rise to different connection patterns. As an example we discuss the connection pattern for the QAP formulation. For convenience we redefine the set of vertices by $V = \{v_{00}, v_{01}, \cdots, v_{0(n-1)}, v_{10}, \cdots, v_{(n-1)(n-2)}, v_{(n-1)(n-1)}\}$, which reflects the grid structure (see figure 1a). The state of a vertex v_{ip} in configuration k, denoted by $r_k(v_{ip})$, corresponds to the value of the 0-1 variable x_{ip} and $s(v_{ip}, v_{jq})$ denotes the connection strength of the edge (v_{ip}, v_{jq}). Thus, the consensus function can be rewritten as

$$C_k = \sum_{(v_{ip}, v_{jq}) \in E} s(v_{ip}, v_{jq})\, r_k(v_{ip})\, r_k(v_{jq}). \tag{18}$$

Next, the set of edges is defined as the union of three disjoint subsets, viz.

- distance edges: $E_d = \{(v_{ip}, v_{jq}) \mid (i \neq j) \wedge (q = (p+1) \bmod n)\}$,
- inhibitory edges: $E_i = \{(v_{ip}, v_{jq}) \mid ((i = j) \wedge (p \neq q)) \vee ((i \neq j) \wedge (p = q))\}$, and
- bias edges: $E_b = \{(v_{ip}, v_{jq}) \mid (i = j) \wedge (p = q)\}$,

where $i, j, p, q = 0, \ldots, n-1$. Consequently the total number of edges equals $2n^3 - n^2$. The inhibitory edges must ensure that eventually (in the course of the consensus maximization) no row or column will have more than one vertex 'on'. The bias edges must ensure that eventually each row and column will have at least one vertex 'on'. Hence, the inhibitory and bias edges must ensure that the constraints given by eqs. 14-15 are satisfied. The connection strengths of the inhibitory and bias edges are chosen such that their contribution to the total consensus is identical for all tours.
Furthermore, the proposed connection pattern ensures that a distance edge (v_{ip}, v_{jq}) will only be activated if the tour goes directly from city i to city j. The connection strength of a distance edge (v_{ip}, v_{jq}) has a negative value proportional to the distance between the cities i and j. Hence, for a given tour the (negative) contribution of the distance edges to the total consensus is proportional to the corresponding tourlength. So, maximizing the consensus function is identical to minimizing eq. 13 subject to eqs. 14-17.
The consensus function C_k has the two properties P1 and P2 mentioned above if the connection strengths are chosen as follows [3]:

$$\begin{aligned} \forall\,(v_{ip}, v_{jq}) &\in E_d : s(v_{ip}, v_{jq}) &=& -d_{ij}, & (19)\\ \forall\,(v_{ip}, v_{jq}) &\in E_i : s(v_{ip}, v_{jq}) &<& -\min(\mu_i, \mu_j), \text{ and} & (20)\\ \forall\,(v_{ip}, v_{ip}) &\in E_b : s(v_{ip}, v_{ip}) &>& \mu_i, & (21) \end{aligned}$$

where

$$\mu_i = \max\,\{d_{ik} + d_{il} \mid k, l = 0, \cdots, n-1 \,\wedge\, (k \neq l)\}. \quad (22)$$

However, if the connection strengths are chosen according eqs. 19-22, relatively large amounts of computer time are required to obtain good final results [3]. This can be avoided if instead of eq. 22 μ_i is chosen as follows:

$$\mu_i = \sum_{k=0,k\neq i}^{n-1} \sum_{l=k,l\neq i}^{n-1} 2(d_{ik} + d_{il})/(n^2 - 3n + 2). \quad (23)$$

In this case property P1 only holds in the near-optimal regime provided the cities of the TSP are positioned sufficiently random.
For a more detailed discussion on the choices of the connection strengths the reader is referred to [3], where a description of the connection pattern for the AP formulation is given as well.

3.3 Computer Simulations

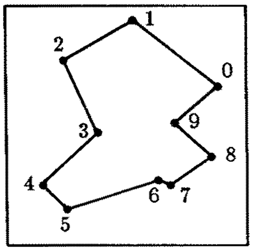
Computer simulations of the Boltzmann machine model for the AP and QAP are carried out for TSP instances with 10 and 30 cities. Figure 1 shows results obtained by the computer simulations for the AP model (figure 1a) and the QAP model (figure 1b). The corresponding tour is shown in figure 1c. The matrices of figs. 1a and 1b correspond to the final states of the vertices of the Boltzmann machine and denote the values of the 0-1 variables of the AP and QAP, respectively.
The simulated annealing algorithm is an approximation algorithm. Final solutions that are equal to optimal solutions are only obtained with probability one in the asymptotic case [1,17]. The quality of the final solutions obtained by the simulations is investigated by collecting statistics. Table 1 shows the results of a statistical analysis for the 10-city and the 30-city TSP taken from [13]. The samples were obtained by running the computer simulation programs M times for different values of the seed of the random generator used in the programs. The various quantities used in the table are defined as follows:

$$\begin{matrix} 0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0&0&0 \end{matrix}$$

(a)

$$\begin{matrix} 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0 \end{matrix}$$

(b)



(c)

Figure 1: *Results of computer simulations for the AP model (a) and the QAP model (b) of a Boltzmann machine. The corresponding tour is shown in (c). The values of the entries of the matrices shown in (a) and (b) correspond to the final values of the 0-1 variables x_{ij} in the AP and the QAP formulation of the TSP, respectively. Note that the result for the AP formulation is unique whereas in the case of the QAP formulation there are $n = 10$ configurations corresponding to the same tour.*

$\bar{\ell}$	:	average tourlength,
σ	:	spreading in the tourlength,
$\ell_{\vee}$	:	smallest observed tourlength,
$\ell^{\wedge}$	:	largest observed tourlength,
M	:	sample size,
I	:	average number of iterations, and
ℓ_{min}	:	smallest known value of the tourlength.

The notion tourlength refers to the length of the final tour obtained by the simulations. As a result of the special choice of the connection strengths discussed above for the QAP model and in ref. [3] for the AP model the property P1 does not hold for the complete regime of values of the consensus function (it only holds in the near-optimal regime). Consequently, consensus maximization may yield a final solution that corresponds to a non-tour. If at the end of the consensus maximization a non-tour is obtained then the maximization algorithm is restarted. The average number of iterations required to obtain a tour is denoted by I.

We end this section with some remarks. In an important work Hopfield and Tank [13] recently introduced networks where the computing elements are linear analog neurons. They showed through computer simulations that near-optimal solutions for combinatorial optimization problems (e.g. (TSP)) could be obtained with these networks. Typical values obtained by Hopfield and Tank for the problem instances discussed above are 2.82 for the 10-city TSP and 5.65 for the 30-city TSP. The networks used by Hopfield and Tank are completely connected, i.e. each individual computing element is connected to all other elements (for a TSP with n cities $O(n^4)$ connections are required). The typical nature of the analog computing elements hampers the implementation of these networks on present-day massively parallel computer systems such as the distributed array processor (DAP) or the connection machine [9]. However, one should take into account that their approach is not designed for implementation on such computer systems but on special neural-network architectures using dedicated hardware. Furthermore, we remark that the typical analog nature of the computing elements, which Hopfield and Tank find to be essential for obtaining satisfactory results with their networks, is circumvented in our approach by the stochastic state-transition mechanism.

	10-city TSP		30-city TSP	
	AP	QAP	AP	QAP
$\bar{\ell}$	2.783	2.815	5.643	5.459
σ	0.097	0.141	0.434	0.312
$\ell_\vee$	2.675	2.675	4.769	4.929
$\ell^\wedge$	3.060	3.277	6.297	6.044
M	100	100	40	25
I	1.1	1.2	1.2	1.9
ℓ_{min}	2.675 [13]		4.299 [13]	

Table 1: *Numerical results obtained for the 10-city and the 30-city TSP (for explanation of the symbols see text).*

4 Learning in a Boltzmann Machine

In the previous section it is shown how the model of the Boltzmann machine can be used to solve combinatorial optimization problems. A special class of optimization problems is given by the set of classification problems [19]. For instance, let $\mathcal{N} = \{n_1, \ldots, n_k\}$ denote a set of k items (e.g. images) and m a partly distorted item (e.g. a corrupted image) then the problem is to find an item $n_i \in \mathcal{N}$ that is "closest to" m. Here the notion "closest to" can be defined in different ways depending on the metric defined on the set $\mathcal{N}$. In principle it should be possible to implement these problems on a Boltzmann machine. However, choosing the appropriate connections and their strengths is very difficult since different items may give rise to conflicting connection strengths. This major problem can be overcome by "learning" the Boltzmann machine the appropriate connection strengths. By applying a learning algorithm the connection strengths are adjusted over a number of iterations (learning cycles) to adapt themselves to a given set of learning examples (e.g. the set of items $\mathcal{N}$) that are subsequently clamped into a subset of the vertices of the Boltzmann machine (i.e. the environmental vertices). The learning algorithm discussed in this section starts off by setting all connection strengths equal to zero. Next, a sequence of learning cycles is completed, each cycle consisting of two phases. In the first phase learning examples are clamped subsequently into the environmental vertices (clamped situation) and for each learning example the Boltzmann machine is equilibrated using the current set of connection strengths (the meaning of equilibration is explained in §4.2). In the second phase all vertices are free to adjust their state (free-running situation) and again equilibration is carried out. Inbetween subsequent learning cycles the connection strengths are adjusted using statistical information obtained from the two situations of the previous learning cycle. This process is continued until the average change (over a number of learning cycles) of the connection strengths becomes zero. After completion of the learning algorithm the Boltzmann machine is able to complete a partial example (i.e. a situation where only a subset of the environmental vertices is clamped) by maximizing the consensus. In this way a Boltzmann machine can not only reproduce given examples but also has associative and restricted inductive capabilities.

Hereinafter we discuss the learning algorithm in more detail and present a number of examples to demonstrate the characteristic features of learning in a Boltzmann machine.

4.1 Extension of the Structural Description

To arrive at a formal description that is well suited for the formulation of a learning algorithm that can be used in a Boltzmann machine we add a few extensions to the structural description of §2.1. For this purpose the set of vertices is divided into three disjoint subsets, V_i, V_h and V_o, with $V = V_i \cup V_h \cup V_o$, where V_i, V_h and V_o denote the sets of input, hidden and output vertices, respectively. The union $V_i \cup V_o$ is denoted by V_{io}, the set of environmental vertices, and by definition we have $|V_{io}| = m$. An environmental configuration l is determined by the states of the vertices $v_i \in V_{io}$. The state of an environmental vertex v_i in an environmental configuration l is denoted by $q_l(v_i)$. The environmental configuration space $\mathcal{Q}$ denotes the set of all possible environmental configurations ($|\mathcal{Q}| = 2^m$). With each environmental configuration l a subspace $\mathcal{Q}_l$ can be associated:

$$\mathcal{Q}_l = \{k \in \mathcal{R} \mid \forall v_j \in V_{io} : r_k(v_j) = q_l(v_j)\}, \tag{24}$$

consisting of all configurations for which the states of the environmental vertices are given by l. The learning set $\mathcal{T}$ denotes the set of examples (environmental configurations) that can be clamped into the environmental vertices. Clearly we have $\mathcal{T} \subset \mathcal{Q}$. An input determines the states of the input vertices $v_j \in V_i$. The set of all possible inputs is denoted by $\mathcal{X}$. Similarly, an output is determined by the states of the output vertices $v_j \in V_o$. The set of all possible outputs is denoted by $\mathcal{Y}$. By definition, we have $\mathcal{Q} = \mathcal{X} \times \mathcal{Y}$.

4.2 A Learning Algorithm

The starting point of the learning algorithm presented in this subsection is given by the learning algorithm introduced by Hinton *et al.* [11,4]. The objective of the learning algorithm is to adjust the connection strengths such that the Boltzmann machine in the free-running situation tends to be with a large probability in environmental configurations that are used as examples in the clamped situation. To this end the probability distributions P and P' defined over the set of environmental configurations are used. P_l denotes the probability that the states of the environmental vertices are given by the environmental configuration l in the clamped situation. Similarly, P_l' denotes the probability that the states of the environmental vertices are given by l in the free-running situation. The probability distribution P is determined by the specific environmental configurations contained in the learning set and the frequency they are clamped into the environmental vertices. P_l is large if: (i) l is contained in the learning set and (ii) l is frequently clamped into the environmental vertices. The probability distribution P' depends on the connection strengths $s(v_i, v_j)$ and as will be pointed out hereinafter P' can be related to the stochastic transition mechanism applied by the simulated annealing algorithm. The objective of the learning algorithm can be formulated as follows: *modify the connection strengths of the Boltzmann machine such that P' is close to P.* We then say that the Boltzmann machine has obtained an understanding of the learning set (e.g. if $P \approx P'$ the Boltzmann machine can obtain a correct output when only the input vertices are clamped).
An information-theoretic measure of the distance between the two probability distributions P and P' is given by the asymmetric divergence G, which is defined as [16]

$$G = \sum_{l \in \mathcal{Q}} P_l \ln \frac{P_l}{P_l'}. \tag{25}$$

The objective of the learning algorithm can be rephrased as: *minimize G by changing the connection strengths.*
Before we describe how G can be minimized we readdress the modelling of the state transitions

of the individual vertices. In the learning algorithm the transitions of free-running vertices are modelled stochastically such that equilibrium is achieved, (see §2.2). In equilibrium the Boltzmann machine tends to be in configurations corresponding to large values of the consensus (see eq. 8). If equilibrium is achieved P'_l is given by

$$P'_l(c) = \sum_{k \in \Omega_l} \pi_k(c), \tag{26}$$

where the $\pi_k(c)$ are the components of the equilibrium vector given by eq. 8. It can be shown [4] that the partial derivative of G with respect to $s(v_i, v_j)$ can be written as

$$\frac{\partial G}{\partial s(v_i, v_j)} = \frac{<p'_{ij}> - <p_{ij}>}{c}, \tag{27}$$

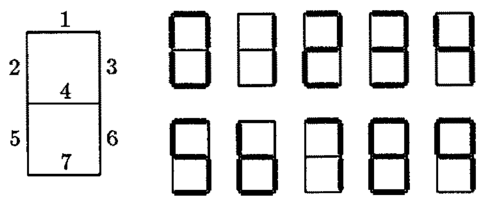
where $<p_{ij}>$ denotes the expectation of the probability that the edge (v_i, v_j) is activated in the clamped situation and $<p'_{ij}>$ denotes the expectation of the probability that (v_i, v_j) is activated in the free-running situation. Both expectations are defined under the condition that equilibrium is reached. To minimize G it suffices to collect statistics on $<p_{ij}>$ and $<p'_{ij}>$ and to change the connection strength proportionally to the difference between the expectations, i.e.

$$\Delta s(v_i, v_j) = \eta\ (<p_{ij}> - <p'_{ij}>), \tag{28}$$

where η denotes a constant.
The learning algorithm can be described as follows: the algorithm starts off with all connection strengths set to zero (tabula rasa). Next, a number of learning cycles is completed by changing repeatedly from the clamped situation to the free-running situation and collecting statistics on the expectations $< p_{ij} >$ and $< p'_{ij} >$. In between subsequent learning cycles the connection strengths are adjusted until G is minimal. In this way the environmental information from the learning set is distributed over and stored in the connection strengths. The learning algorithm consists of a number of learning cycles, that can be depicted schematically in pseudo-Pascal as follows:

```
PROCEDURE LEARNING_CYCLE;
begin
   for i := 1 to number_of_times do
   begin
      CHOOSE_EXAMPLE(according to $P$);
      EQUILIBRATE(hidden vertices);
      COLLECT_STATISTICS($< p_{ij} >$);
   end;
   for i := 1 to number_of_times do
   begin
      RUN_FREE;
      EQUILIBRATE(all vertices);
      COLLECT_STATISTICS($< p'_{ij} >$);
   end;
   ADJUST_CONNECTION_STRENGTHS;
end;
```

By definition environmental configurations l not included in the learning set $\mathcal{T}$ have a corresponding probability $P_l = 0$. Consequently, the asymmetric divergence G is not properly defined for



digit	input	output
0	(1,1,1,0,1,1,1)	(1,0,0,0,0,0,0,0,0,0)
1	(0,0,1,0,0,1,0)	(0,1,0,0,0,0,0,0,0,0)
2	(1,0,1,1,1,0,1)	(0,0,1,0,0,0,0,0,0,0)
3	(1,0,1,1,0,1,1)	(0,0,0,1,0,0,0,0,0,0)
4	(0,1,1,1,0,1,0)	(0,0,0,0,1,0,0,0,0,0)
5	(1,1,0,1,0,1,1)	(0,0,0,0,0,1,0,0,0,0)
6	(0,1,0,1,1,1,1)	(0,0,0,0,0,0,1,0,0,0)
7	(1,0,1,0,0,1,0)	(0,0,0,0,0,0,0,1,0,0)
8	(1,1,1,1,1,1,1)	(0,0,0,0,0,0,0,0,1,0)
9	(1,1,1,1,0,1,0)	(0,0,0,0,0,0,0,0,0,1)

(c)

Figure 2: *Schematic representation of the seven-segment display of the digits 0 - 9 (a) and (b), and the coded representation (c).*

these environmental configurations unless $P_l' = 0$, which can only be realized by infinitely large connection strengths due to the stochastic nature of the consensus optimization. To avoid this problem noise is introduced which allows the environmental vertices in the clamped situation to change their states with a small probability.

Equilibration is carried out by using the simulated annealing algorithm. Again a sequence of Markov chains is generated at descending values of c. However, in contrast to the consensus optimization described in §2.2, the value of c is not lowered until it approaches 0 but the algorithm is terminated at some finite value of $c > 0$ for which the acceptance ratio $\chi(c)$ is smaller than some fixed value χ_f (usually $\chi_f = 0.25$), where $\chi(c)$ equals the number of accepted transitions divided by the number of proposed transitions at a given value of c. In this way equilibration is rapidly achieved [2].

The results obtained by the learning algorithm during the learning phase can be tested by clamping an input into the input vertices. The set of inputs in the test is denoted by the test set. For a given input the consensus is maximized using the connection strengths obtained from the learning phase. Here, both the hidden and the output vertices are considered as free vertices, i.e. they are allowed to change their state during the consensus optimization. After optimization of the consensus the test can be completed by comparing the resulting output with the "expected" output. Consensus optimization is again carried out using the simulated annealing algorithm as described in §2.2. The number of vertices is discussed separately for each example.

4.3 Computer Simulations

In this section the learning behaviour of the Boltzmann machine is discussed by means of a number of simulation experiments. Simulations are carried out to illustrate the three characteristic functions of the Boltzmann machine mentioned above, e.g. memory, association and induction [1]. The simulations are based on a model of the Boltzmann machine in which

- all input vertices are mutually connected,
- all output vertices are mutually connected,

	0	1	2	3	4	5	6	7	8	9
0	100	-	-	-	-	-	-	-	-	-
1	-	98	-	-	-	-	-	2	-	-
2	-	-	99	1	-	-	-	-	-	-
3	-	-	-	100	-	-	-	-	-	-
4	-	-	-	-	100	-	-	-	-	-
5	-	-	-	-	-	100	-	-	-	-
6	-	-	-	-	-	-	100	-	-	-
7	-	13	-	-	-	-	-	87	-	-
8	7	-	2	-	-	-	-	-	91	-
9	-	-	-	-	3	-	-	-	-	97

(a)

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

(b)

Figure 3: *(a) Score matrix for the memory test,(b) Hamming-distance matrix for the inputs*

- all hidden vertices are connected to all input and output vertices, and
- all vertices are connected to themselves.

Memory

To demonstrate the capability of the Boltzmann machine to store information which can be retrieved afterwards, the following experiment is discussed. For the display of decimal digits the standard form is the seven-segment display (figure 2a and 2b). Each of the seven segments is a separate display (e.g. a LED or Liquid Cristal) that can be turned 'on' or 'off'. The display can be controlled by a binary seven-tuple where a 0 corresponds to 'off' and a 1 to 'on'. Thus, the digits 0 up to 9 can be coded by the inputs given in figure 2c.

The goal of the experiment is to learn the Boltzmann machine the codes corresponding to the 10 digits of figure 2c by using the learning algorithm described in §4.2. For this purpose a Boltzmann machine was modelled consisting of 7 input, 10 hidden and 10 output vertices. The learning set was given by the input-output combinations of figure 2c. During each learning cycle 20 randomly chosen learning examples were clamped into the environmental vertices. After a learning phase consisting of 870 learning cycles the results obtained in the learning phase were tested. During the test phase inputs given in figure 2c were used. Each input of the test set was clamped 100 times into the input vertices. The results of the test phase are shown in the score matrix given in figure 3a. For 972 of the 1000 inputs (97%) a correct output was retrieved. For the digits 0, 3, 4, 5 and 6 a full score was obtained. In the case of an incorrect retrieval (3%) it appears that the input corresponding to the retrieved output have a large "similarity" to the clamped input (1 and 7, 2 and 3, 0, 2 and 8, 4 and 9). A large similarity may be defined here as a small Hamming distance between the corresponding inputs. In the case of an incorrect retrieval the Hamming distance equals 1 or 2 (see the Hamming-distance matrix given in figure 3b). This suggests associative capabilities. Furthermore, from figure 3a it is observed that the score matrix is not symmetric. This feature can not be explained from simple external quantities such as the Hamming distance but is related to internal quantities of the Boltzmann machine, i.e. the maximal consensus and the connection strengths. This feature is also observed in the next paragraph.

Association

To illustrate the associative capabilities of a Boltzmann machine the following experiment is discussed. Using the connection strengths obtained from the learning phase of the experiment described in the previous paragraph, the Boltzmann machine was tested by clamping inputs that have a large similarity with the inputs of the learning set. Again each input was clamped 100 times. The results are shown in the score matrix given in figure 4a. The corresponding Hamming-distance matrix is given in figure 4b. The results clearly demonstrate that the outputs obtained by the test

	0	1	2	3	4	5	6	7	8	9
0	82	-	-	-	-	-	18	-	-	-
1	-	100	-	-	-	-	-	-	-	-
2	-	-	100	-	-	-	-	-	-	-
3	-	-	2	56	-	42	-	-	-	-
4	-	-	-	-	97	-	-	-	-	3
5	30	-	-	7	-	61	2	-	-	-
6	7	-	-	-	-	-	93	-	-	-
7	-	1	-	-	-	-	-	99	-	-
8	4	-	49	35	-	-	-	-	12	-
9	-	26	-	-	-	-	-	64	-	10

(a)

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

(b)

Figure 4: *(a) Score matrix for the association test, (b) Hamming-distance matrix for the inputs.*

correspond to inputs of the learning set that have a large similarity with the tested inputs (i.e. a Hamming distance 1 or 2). Similar to the memory test different scores are observed for outputs whose corresponding inputs (in the learning set) have equal Hamming distances to the inputs used in the test. For instance the input corresponding to the 9-th row has a Hamming distance 1 to the inputs corresponding to the digits 2, 3 and 8. The scores, however, vary considerably for the corresponding outputs. Again, these features can only be explained from the associated internal quantities such as the connection strengths and the maximal consensus.

Induction

By induction we define the ability to discover an underlying pattern from a set of examples in such a way that afterwards inputs (not corresponding to or resembling the inputs of the learning set) can be classified according to this pattern. To illustrate that the Boltzmann machine exhibits restricted inductive capabilities the following experiment is discussed.

A Boltzmann machine is modelled consisting of 8 input, 12 hidden and 2 output vertices. The learning set consists of input-output combinations with an input of length 8 and an output of length 2. The output equals (1,0) if at least 2 successive components of the input equal 1, and (0,1) otherwise, e.g.

$$(1,0,1,1,0,0,1,0) \rightarrow (1,0)$$
$$(1,0,1,0,1,0,1,0) \rightarrow (0,1)$$

The goal of this experiment is to "learn" the Boltzmann machine to discover a '11'-pair in an input by using a learning set consisting of a subset of all the 256 possible environmental configurations.

By using another subset as test set the induction result can be tested.
The learning set consisted of 60 learning examples, 30 having a '11'-pair and 30 not. The test set consisted of 40 inputs, 23 having a '11'-pair and 17 not. During each learning cycle 20 randomly chosen learning examples were clamped into the environmental vertices. After a learning phase of 560 learning cycles the results obtained by the learning was tested. Each input belonging to the test set was clamped twice. In 76% of the tests a correct classification was obtained (61 out of 80). To demonstrate the significance of this result it is compared with a result obtained by randomly choosing an output (0,1) or (1,0) with a probability of 0.5 to choose a correct output. In this way the probability of choosing 61 (out of 80) correct outputs equals 10^{-6}. The calculation is based on a binomial distribution. It, therefore, may be concluded that the degree to which the Boltzmann machine has "learned" the underlying '11'-pattern is significant. Moreover, it should be taken into account that recognition of a '11'-pair by a Boltzmann machine is a difficult task as it can not use the advantage of topological information, e.g. the human brain can discriminate a '11'-pair from a string of binary numbers by using visual information.

5 Conclusions

The model of Boltzmann machines combines typical features of neural networks and connectionist models. Salient features of the model are its massively parallel architecture of simple stochastic computing elements and the distribution of knowledge. It is concluded that Boltzmann machines can be effectively used to solve combinatorial optimization problems. For two distinct 0-1 integer formulations of the traveling salesman problem (as an assignment problem and as a quadratic assignment problem) it is shown that near-optimal solutions for these problems can be obtained by a network with $O(n^2)$ computing elements and $O(n^3)$ connections.
Furthermore, it is concluded that stochastic concepts can be successfully applied to the modelling of learning in parallel networks such as the Boltzmann machine. By "programming" the Boltzmann machine through a set of learning examples it exhibits memory, associative and inductive functions. Especially the latter two functions are interesting new features.
Asynchronous parallelism can be effectively exploited by the Boltzmann machine. Future research on the Boltzmann machine will have to concentrate on massively parallel hardware implementations. This is to a large extent an unexplored field of interest which will require considerable research efforts.
As an overall conclusion we state that the Boltzmann machine is promising for a number of complex computational tasks such as combinatorial optimization and learning. However, more analysis, large-scale simulations and practical experience are needed to judge the Boltzmann machine on its true merits.

References

[1] Aarts, E.H.L. and P.J.M. van Laarhoven, Statistical Cooling: A General Approach to Combinatorial Optimization Problems, *Philips Journ. Res.*, 40(1985)193.

[2] Aarts, E.H.L. and J.H.M. Korst, Simulation of Learning in Parallel Networks Based on the Boltzmann Machine, *Proc. 2nd Eur. Simul. Congr.*, Antwerp, September 1986, p. 391.

[3] Aarts, E.H.L. and J.H.M. Korst, Solving Traveling Salesman Problems with a Boltzmann Machine, *The European Journal of Operations Research*, to be submitted.

[4] Ackley, D.H., G.E. Hinton and T.J. Sejnowski, A Learning Algorithm for Boltzmann Machines, *Cognitive Science*, 9(1985)147.

[5] Feldman, J.A. and D.H. Ballard, Connectionist Models and Their Properties, *Cognitive Science*, 6(1982)205.

[6] Fahlman, S.E., G.E. Hinton and T.J. Sejnowski, Massively Parallel Architectures for AI: NETL, Thistle and Boltzmann Machines, *Proc. Nat. Conf. on AI, AAAI-83*, August 1983, p. 109.

[7] Fahlman, S.E and G.E. Hinton, Connectionist Architectures for Artificial Intelligence, *Computer*, January, 1987, p. 100.

[8] Garfinkel, R.S., Motivation and Modeling, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), *The Traveling Salesman Problem*, Wiley, Chichester, 1985, p. 17.

[9] Hillis, W.D., *The Connection Machine*, MIT-press, Cambridge (MA), 1985.

[10] Hinton G.E. and T.J. Sejnowski, Optimal Perceptual Inference, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Washington DC, June 1983, p. 448.

[11] Hinton, G.E., T.J. Sejnowski and D.H. Ackley, Boltzmann Machines: Constraint Satisfaction Machines that Learn, *Tech. Rep.* CMU-CS-84-119, Carnegie-Mellon University, 1984.

[12] Hopfield, J.J., Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proc. Nat. Academy Sciences*, USA, 79(1982)2554.

[13] Hopfield, J.J. and D.W. Tank, Neural Computation of Decisions in Optimization Problems, *Biological Cybernetics*, 52(1985)141.

[14] Kirkpatrick, S., C.D. Gelatt Jr. and M.P. Vecchi, Optimization by Simulated Annealing, *Science*, 220(1983)671.

[15] Korst, J.H.M. and E.H.L. Aarts, Solving Combinatorial Optimization Problems with Massively Parallel Computer Architectures Based on the Boltzmann Machine, *Operations Research*, to be submitted.

[16] Kullbach, S., *Information Theory and Statistics*, Wiley, New York, 1959.

[17] Laarhoven, P.J.M. van and E.H.L. Aarts, *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1987, in press.

[18] Prager R.G. *et al.*, Boltzmann Machines for Speech Recognition, *Computer Speech and Language*, Vol. 1, 1986.

[19] *Proceedings NASI on Pattern Recognition: Theory and Applications*, P.A. DeVijver and J. Kittler (eds.), Springer, 1987 (and references therein).

[20] Sejnowski, T.J. and C.R. Rosenberg, NETtalk: A Parallel Network that Learns to Speak Aloud, John Hopkins University, *Technical Report*, JHU/EECS-86/01, 1986.

[21] Stevens, J.K., Reverse Engineering the Brain, *Byte*, 10(4)(1985)287.

[22] Traherne, J.F. *et al.*, Speech Processing with a Boltzmann Machine, *Proc. IEEE ICASSP-86*, Tokyo, 1986, p. 725.

[23] Treleaven, P.C., A.N. Refenes, K.J. Lees and S.C. McCabe, Computer Architectures for Artificial Intelligence, *Lecture Notes*, University of Reading, Reading, 1985.

[24] Waldrop, M.M., Artificial Intelligence in Parallel, *Science*, 225(1985)608.