Neural networks for combinatorial optimization

Citation for published version (APA):

Document status and date:
Published: 01/01/1994

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 01. Apr. 2019
Memorandum COSOR 94-29

Neural networks
for combinatorial optimization

E.H.L. Aarts
P.H.P. Stehouwer
J. Wessels
P.J. Zwietering

Eindhoven, August 1994
The Netherlands
NEURAL NETWORKS FOR COMBINATORIAL OPTIMIZATION

Emile H.L. Aarts$^{1,2}$
Peter H.P. Stehouwer$^1$
Jaap Wessels$^{1,3}$
Patrick J. Zwietering$^4$

$^1$ Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
$^2$ Philips Research Laboratories, P.O. Box 80000, NL-5600 JA Eindhoven, The Netherlands
$^3$ International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria
$^4$ University of Twente, P.O. Box 217, NL-7500 AE Enschede, The Netherlands

1 INTRODUCTION

Recent advances in the design and manufacturing of integrated circuits have brought the construction of parallel computers, consisting of thousands of individual processing units, within our reach. A direct consequence of these technological advances is the growing interest in computational models that support the exploitation of massive parallelism. Connectionist models [Feldman & Ballard, 1982] are computational models that are inspired by an analogy with the neural network of human brains, in which massive parallelism is generally considered to be of great importance. The corresponding parallel computers are called neural networks and the field of research neural computing. The greatest potential of neural computing is in the areas where high computation rates are required and present computer systems perform poorly. However, the potential benefit of neural computing extends beyond purely technical advantages. Many models in neural computing have human-like capabilities such as association and learning, which are essential in areas such as speech and image processing [Kohonen, 1988]. Moreover, these capabilities provide a kind of robustness and fault tolerance, since they compensate for minor variations in input data and damages in the network components.

In general, a neural network consists of a network of elementary nodes that are linked through weighted connections. The nodes represent computational units, which are capable of performing a simple computation that consists of a summation of the weighted inputs of the node, followed by the addition of a constant called the threshold or bias, and the application of a non-linear response function. The result of the computation of a unit constitutes the output of the corresponding node. Subsequently, the output of a node is used as an input for the nodes to which it is linked through an outgoing connection. For a detailed review of the different neural network models, we refer the reader to the textbooks by Aarts & Korst [1989], Hecht-Nielsen [1990], Hertz, Krogh & Palmer [1991], and Kosko [1992].

Combinatorial optimization is concerned with the problem of finding an optimal solution among a finite, possibly large, number of alternative solutions. Over the past few decades, a
wide variety of such problems has emerged from such diverse areas as management science, computer science, engineering, VLSI design, etc. An important achievement in combinatorial optimization is the existence of two classes of related problems, $\mathcal{P}$ and $\mathcal{NP}$, and the fact that, unless $\mathcal{P} = \mathcal{NP}$, there exists a class of combinatorial optimization problems for which no algorithms exist that solve each instance of the problem to optimality within a running time that is polynomially bounded in the size of the instance. These problems are called \textit{NP-hard} [Garey & Johnson, 1979], and a direct consequence of the NP-hardness of a problem is that optimal solutions probably can not be obtained in reasonable amounts of computation time.

However, large NP-hard problems still must be solved and in constructing appropriate algorithms for combinatorial optimization problems, roughly speaking, there are two options. Either one goes for optimality at risk of very large, possibly impracticable running times, or one strives for more quickly obtainable solutions at the risk of sub-optimality. The first option constitutes the class of \textit{optimization algorithms}. Examples are enumeration methods using cutting plane, branch and bound, or dynamic programming techniques [Papadimitriou & Steiglitz, 1982]. The second option constitutes the class of \textit{approximation algorithms}; examples are local search and randomization algorithms.

During the past decade, a substantial amount of literature has been published in which neural networks are used for combinatorial optimization problems. The most important motivation for using neural networks is the potential speed up obtained by massively parallel computation. The first researchers that applied neural network techniques to combinatorial optimization problems were Hopfield & Tank [1985] and Baum [1986].

When applying a certain neural network model to a certain task, besides choosing the right number of units and the right connections, one has to choose the connection strengths such that the network performs the task. One can choose between two approaches, i.e., \textit{reproduction} and \textit{adaptation}. In reproduction the connection strengths are given and kept constant during the network execution. This embeds certain information into the network by design, which is reproduced during operation. In adaptation the connection strengths are iteratively adjusted until the neural network performs a given task accurately.

In most applications of neural network models to combinatorial optimization problems reproduction is used. Perhaps the only exception is the application of Kohonen's feature maps [Kohonen, 1982; Kohonen, 1988] to Euclidean instances of the travelling salesman problem; see [Aarts & Stehouwer, 1993; Potvin, 1994] and the references therein. In this paper we concentrate on reproduction approaches that apply to combinatorial optimization problems in general. In that context we deal with Boltzmann machines [Hinton & Sejnowski, 1983], Hopfield networks [Hopfield, 1982], and multi-layered perceptrons [Minsky & Papert, 1969]. For a more extensive overview of neural network methods for combinatorial optimization we refer the reader to [Looi, 1992].

The paper is organized as follows. In Section 2, we introduce our formulation of combinatorial optimization problems. Section 3 discusses Boltzmann machines and Hopfield networks and their relation with combinatorial optimization. The emphasis in this section is on Boltzmann machines. In Section 4 we show how combinatorial optimization problems can be reformulated as classification problems and how they can be solved by multi-layered perceptrons. Section 5 presents a discussion and some concluding remarks. The paper ends with some references.
2 COMBINATORIAL OPTIMIZATION

We start with presenting a formal description of a combinatorial optimization problem [Garey & Johnson, 1979].

Definition 2.1. A combinatorial optimization problem Π is either a minimization problem or a maximization problem and consists of

(i) a set \( \mathcal{D}_\Pi \) of instances,
(ii) for each instance \( I \in \mathcal{D}_\Pi \), a finite set \( \mathcal{S}_\Pi(I) \) of possible solutions, and
(iii) a cost function \( c_\Pi \) that assigns to each instance \( I \in \mathcal{D}_\Pi \) and each solution \( \sigma \in \mathcal{S}_\Pi(I) \) a positive rational number.

For an instance \( I \in \mathcal{D}_\Pi \), if Π is a minimization (maximization) problem, the problem is to find a globally-optimal solution \( \sigma^* \in \mathcal{S}_\Pi(I) \) such that \( c_\Pi(\sigma^*) \leq c_\Pi(\sigma) \) (\( c_\Pi(\sigma^*) \geq c_\Pi(\sigma) \)), for all \( \sigma \in \mathcal{S}_\Pi(I) \).

In this paper we consider combinatorial optimization problems as minimization problems. This can be done without loss of generality since maximization is equivalent to minimization after simply reversing the sign of the cost function.

In most applications of neural networks to combinatorial optimization problems, the number of nodes and connections is fixed. Therefore, we resort to a definition of a combinatorial optimization problem where the size of an instance is fixed. Furthermore we make a distinction between solutions and feasible solutions. With a feasible solution we mean a solution that satisfies the constraints in the problem. The problem is then to find a feasible solution for which the cost is optimal and can be formulated as follows.

Definition 2.2. A combinatorial optimization problem is represented by a 4-tuple \( (\mathcal{D}, \mathcal{S}, \mathcal{F}, c) \), where (i) for some \( n \in \mathbb{N} \), \( \mathcal{D} \subseteq \mathbb{R}^n \) denotes a set of instance defining parameters, (ii) for some \( k \in \mathbb{N} \) and for all \( x \in \mathcal{D} \), \( \mathcal{S}(x) \subseteq \mathbb{R}^k \) denotes a finite set of solutions and \( \mathcal{F}(x) \subseteq \mathcal{S}(x) \) the set of feasible solutions for the instance defined by \( x \), and (iii) for all \( x \in \mathcal{D} \), \( c(\cdot; x) : \mathcal{S}(x) \rightarrow \mathbb{R} \) denotes a cost function on the set of solutions for the instance defined by \( x \). For a given instance \( (\mathcal{S}(x), \mathcal{F}(x), c(\cdot; x)) \) defined by some \( x \in \mathcal{D} \), the problem is to find a feasible solution with optimal cost, i.e., we must find an \( y \in \mathcal{F}(x) \), such that \( c(y; x) \leq c(z; x) \), for all \( z \in \mathcal{F}(x) \).

3 BOLTZMANN MACHINES AND HOPFIELD NETWORKS

Boltzmann machines have been introduced by Hinton & Sejnowski [1983] and can be viewed as an extension of discrete Hopfield networks [Hopfield, 1982] in the sense that they replace the greedy local search dynamics of Hopfield networks with a randomized local search dynamics. This extension is achieved through the use of stochastic computing elements.

Interest in Boltzmann machines extends over many disciplines, e.g., computer architectures, artificial intelligence, modelling of neural brain behavior, combinatorial optimization, and image and speech processing; for an overview we refer to [Aarts & Korst, 1989]. Here, we present a mathematical model and briefly discuss the theory of self-organization in Boltzmann machines. For a discussion about adaptation in Boltzmann machines we refer to [Aarts & Korst, 1989].
3.1 NETWORK STRUCTURE

Like in a Hopfield network, a Boltzmann machine consists of a number of two-state units, that are connected in some way. The network can be represented by a pseudograph $B = (\mathcal{U}, \mathcal{C})$, where $\mathcal{U}$ denotes a finite set of nodes and $\mathcal{C} \subseteq \mathcal{U} \times \mathcal{U}$ a set of symmetric connections. A connection $(u, v) \in \mathcal{C}$ joins the nodes $u$ and $v$. The set of connections may contain loops, i.e., $(u, u) \in \mathcal{C}$, for some $u \in \mathcal{U}$. If two nodes are connected, they are called adjacent. A node $u$ can be in one of two states, either in state “0” or state “1”.

Definition 3.1. A configuration $k$ is a global state which is uniquely defined by a sequence of length $|\mathcal{U}|$, whose $u^{th}$ component $k(u)$ denotes the state of node $u$ in configuration $k$. $\mathcal{R}$ denotes the set of all configurations. □

Definition 3.2. A connection $(u, v) \in \mathcal{C}$ is activated in a given configuration $k$ if both $u$ and $v$ have state “1”, i.e., if $k(u) \cdot k(v) = 1$. □

Definition 3.3. With each connection $(u, v) \in \mathcal{C}$ a connection strength $s_{(u,v)} \in \mathbb{R}$ is associated as a quantitative measure for the desirability that $(u, v)$ is activated. By definition, $s_{(u,v)} = s_{(v,u)}$. If $s_{(u,v)} > 0$, it is desirable that $(u, v)$ is activated; if $s_{(u,v)} < 0$, it is undesirable. Connections with a positive (negative) strength are called excitatory (inhibitory). □

Definition 3.4. The consensus function $C : \mathcal{R} \rightarrow \mathbb{R}$ assigns to each configuration $k$ a real number, called the consensus, which equals the sum of the strengths of the activated connections, i.e.,

$$C(k) = \sum_{(u,v) \in \mathcal{C}} s_{(u,v)} k(u)k(v). \quad (1)$$

Generally speaking, the consensus is large if many excitatory connections are activated, and it is small if many inhibitory connections are activated. The consensus is a global measure indicating to what extent the nodes in the network have reached a consensus about their individual states, subject to the desirabilities expressed by the individual connection strengths. Since the connection strengths impose local constraints, these networks are also often called constraint satisfaction networks [Hinton & Sejnowski, 1983].

3.2 NETWORK DYNAMICS

Self-organization in a Boltzmann machine is achieved by allowing nodes to change their states, from “0” to “1” or reverse. This is similar to the self-organization in Hopfield networks. However, in a Boltzmann machine the acceptance of a proposed state change is stochastic whereas in a Hopfield network it is deterministic. Let the network be in configuration $k$, then a state change of node $u$ results in a configuration $l$, with $l(u) = 1 - k(u)$ and $l(v) = k(v)$ for each $v \neq u$. Furthermore, let $\mathcal{C}_u$ denote the set of connections incident with node $u$, excluding $(u,u)$. Then the difference in consensus induced by a state change of node $u$ in configuration $k$, is given by

$$\Delta C_k(u) = (1 - 2k(u)) \left( s_{(u,u)} + \sum_{(u,v) \in \mathcal{C}_u} s_{(u,v)} k(v) \right). \quad (2)$$
The effect on the consensus, resulting from a state change of node $u$, is completely determined by the states of its adjacent nodes and the corresponding connection strengths. Consequently, each node can locally evaluate its state change since no global calculations are required.

**Definition 3.5.** The response in a Boltzmann machine of an individual state change of node $u$ to its adjacent nodes in a configuration $k$ is a stochastic function which is given by

\[
P_c(u|k) = \frac{1}{1 + \exp(-\Delta C_k(u)/c)},
\]

where $\Delta C_k(u)$ is given by (2). The probability of accepting a state change is controlled by the parameter $c$, and is given by (3). In most cases the algorithm is implemented such that initially the value of $c$ is large, in which case the probability of accepting deteriorations is large. Subsequently, the value of $c$ is decreased to eventually become 0, in which case no deteriorations are accepted anymore.

Implementation of state changes in a Boltzmann machine is done by simulated annealing, which is a randomized local search algorithm for solving combinatorial optimization problems. It originates from the simulation of physical annealing processes. Generally speaking, the algorithm allows the acceptance of state changes that deteriorate the consensus, in order to escape from poor locally maximal configurations. The probability of accepting a state change is controlled by the parameter $c$, and is given by (3). In most cases the algorithm is implemented such that initially the value of $c$ is large, in which case the probability of accepting deteriorations is large. Subsequently, the value of $c$ is decreased to eventually become 0, in which case no deteriorations are accepted anymore.

Simulated annealing can be mathematically modeled by using the theory of Markov chains; for a detailed description we refer to [Aarts & Korst, 1989]. This theory can also be used to describe the state changes of the nodes in a Boltzmann machine. To this end we distinguish between two models, viz., *sequential Boltzmann machines* and *parallel Boltzmann machines*.

**Sequential Boltzmann Machines.** In a sequential Boltzmann machine, nodes may change their states only one at a time. The resulting iterative procedure can be described as a sequence of Markov chains, where each chain consists of a sequence of trials and the outcome of a given trial depends probabilistically only on the outcome of the previous trial. A trial consists of the following two steps. Given a configuration $k$, then a neighboring configuration $k_u$ is generated, determined by a node $u \in U$ that proposes a state change. Next it is evaluated whether $k_u$ is accepted or not. More specifically, the outcome of the trial is $k_u$ with probability $P_c(u|k)$, and otherwise $k$. For the acceptance probability of (3), we obtain the following result.

**Theorem 3.1 (Aarts & Korst [1989]).** For a sequential Boltzmann machine with a response function given by (3) the following holds.

(i) The probability $q(c)$ of obtaining a configuration $k$ after a sufficiently large number of trials carried out at a fixed value of $c$ is given by

\[
q_k(c) = \frac{\exp(C(k)/c)}{\sum_{l \in \mathbb{R}} \exp(C(l)/c)}.
\]

(ii) For $c \downarrow 0$, (4) reduces to a uniform distribution over the set of configurations with maximum consensus.

The expression of (4) is often referred to as the *stationary distribution* of the corresponding Markov chain, and it is known in statistical physics as the Boltzmann distribution, which
explains the name "Boltzmann machine". The process of reaching the stationary distribution is called *equilibration*. The first part of the theorem states that configurations with a higher consensus have a larger probability of occurring than configurations with a lower consensus. The second part states that if \( c \) approaches 0 slowly enough to allow equilibration, the Boltzmann machine finds with probability one a configuration with maximum consensus. This result plays an important role in combinatorial optimization as is discussed in Section 3.3.

In practical implementations, the asymptoticity conditions cannot be attained and thus convergence to a configuration with maximum consensus is not guaranteed. In those cases the Boltzmann machine finds a locally optimal configuration, i.e., a configuration with consensus no worse than that of any of its neighboring configurations. The convergence of the Boltzmann machine is determined by a set of parameters, known as the *cooling schedule*, which determine the convergence of the simulated annealing algorithm. These parameters are: a start value of \( c \), a decrement rule to lower the value of \( c \), the length of the individual Markov chains and a stop criterion. In [Aarts & Korst, 1989], some cooling schedules tailored to Boltzmann machines are discussed.

**Parallel Boltzmann Machines.** To model parallelism in a Boltzmann machine, we distinguish between synchronous and asynchronous state changes.

*Synchronous state changes* are scheduled in successive trials, where each trial consists of a number of individual state changes. During each trial a node is allowed to propose a state change exactly once. Synchronous parallelism requires a global clocking scheme to control the synchronization. An extensive discussion of synchronously parallel Boltzmann machines is given by [Zwietering & Aarts, 1991]. Here, we only briefly summarize the most important results. The main result is a conjecture which states that under certain mild conditions a stationary distribution different from (4) is attained, which converges as \( c \) approaches 0 to a distribution over the set of configurations for which the so-called extended consensus is maximal. A proof of this conjecture is an open problem. However, for two special cases correctness can be proved. These are the cases of *limited parallelism*, where nodes may change their states simultaneously only if they are not adjacent, and *full parallelism*, where in each trial all nodes change their states simultaneously according to \( P_c(u|k) \).

*Asynchronous state changes* are evaluated concurrently and independently. Units generate state changes and accept or reject them on the basis of information that is not necessarily up-to-date, since the states of adjacent nodes may have changed in the meantime. Asynchronous parallelism does not require a global clocking scheme, which is of advantage in hardware implementations. However, this type of parallelism cannot be modelled by Markov chains and requires a completely different approach, and so far little progress has been made in this direction. A brief discussion on the subject can be found in [Aarts & Korst, 1989].

**Complexity issues.** Goles-Chacc, Fogelman-Soulie & Pellegrin [1985] showed that finding locally maximal configurations in a Boltzmann machine with a deterministic response function, i.e., with \( c = 0 \) in (3), requires pseudo-polynomial running times. This result has been generalized by Schaffer & Yannakakis [1991] who showed that the problem of finding locally maximal configurations in this type of network is PLS-complete. This result implies that it is unlikely that algorithms exist that find a locally maximal configuration in a Boltzmann machine with a worst-case running time that can be bounded by a polynomial in the size of the network.
Parberry & Schnitger [1989] have shown that synchronously parallel Boltzmann machines can be simulated by a standard unbounded fan-in threshold circuit with polynomially bounded size, and running time greater by a constant factor. Consequently, Boltzmann machines are equally powerful as threshold circuits, which can be viewed as a standard parallel machine model.

3.3 COMBINATORIAL OPTIMIZATION

The ability of a Boltzmann machine to obtain configurations with a large consensus can be used to handle combinatorial optimization. Given a combinatorial optimization problem \((D, S, \mathcal{F}, c)\) and an instance \((S(x), \mathcal{F}(x), c(\cdot, x))\) of this problem defined by some \(x \in D\), the problem is to find a feasible solution \(y \in \mathcal{F}(x)\) with optimal cost.

A Boltzmann machine can be used to solve instances of combinatorial optimization problems by defining a correspondence between the configurations of the Boltzmann machine and the solutions of the combinatorial optimization problem in such a way that the cost function of the combinatorial optimization problem is transformed into the consensus function associated with the Boltzmann machine. In general this can be done by formulating the combinatorial optimization problem as a 0-1 integer programming problem in which the decision variables assume values equal to 0 or 1. The values of the 0-1 variables correspond to the states of the nodes. The cost function and the constraints that go with the combinatorial optimization problem are embedded into the Boltzmann machine by choosing the appropriate connections and their strengths. In this way, maximizing the consensus in the Boltzmann machine is equivalent to solving the corresponding instance of the combinatorial optimization problem. More specifically, it is often possible to construct a Boltzmann machine such that the following properties hold.

(i) Each locally maximal configuration of the Boltzmann machine corresponds to a feasible solution.

(ii) The higher the consensus of the corresponding configuration, the better the cost of the corresponding feasible solution.

These properties imply that feasible solutions can be obtained and that configurations with near-maximal values of the consensus function correspond one-to-one to near-optimal solutions of the combinatorial optimization problem. This feature enables Boltzmann machines to be used for approximation purposes, which we have demonstrated for several well-known problems, including TRAVELLING SALESMAN, GRAPH COLORING, INDEPENDENT SET, and others [Aarts & Korst, 1989]. It was found that for graph theoretical problems, such as GRAPH COLORING and INDEPENDENT SET the performance was good in the sense that high-quality solutions were obtained within small running times. The performance for TRAVELLING SALESMAN was poor.

4 MULTI-LAYERED PERCEPTRONS

Multi-layered perceptrons can be viewed as an extension of the single-layered perceptrons designed by Rosenblatt; see [Rosenblatt, 1958; Rosenblatt, 1962]. Rosenblatt showed that perceptrons can be used for adaptive pattern classification, by proving his famous perceptron convergence theorem. This theorem states that the perceptron convergence procedure finds the connection strengths of a one-layered perceptron that solves a given classification problem if such a solution exists. Among others, Minsky & Papert [1969] demonstrated the limitations of
one-layered perceptrons by showing that they can only classify sets that are linearly separable. Minsky & Papert suggested the use of multi-layered perceptrons to overcome these difficulties. After the convincing argument of Minsky & Papert and the lack of a convergence procedure for multi-layered perceptrons, interest in the perceptrons dropped to a modest level.

Recently, multi-layered perceptrons regained interest due to the discovery of suitable learning algorithms such as the back-propagation algorithm, that can be used to find the connection strengths that correspond to a given input-output behavior [Rumelhart, Hinton & Williams, 1986; Werbos, 1990]. However, despite many successful practical applications of the back-propagation algorithm there is still a large number of unresolved questions about the use of multi-layered perceptrons. The optimal choice of the number of layers and the number of nodes in each layer, are examples of such open problems. The number of layers and the number of nodes, minimally required to solve a given problems, are two of most important complexity measures considered for multi-layered perceptrons. Here we study the multi-layered perceptron complexity of combinatorial optimization problems, by viewing those problems as a special type of classification problems. This approach is mainly based upon the results presented in [Zwietering, 1994].

The outline of this section is as follows. After introducing the neural network model known as multi-layered perceptrons, we discuss some results about the classification capabilities of multi-layered perceptrons that use a hard-limiting response function. The section ends by describing how insight in the classification capabilities of multi-layered perceptrons can be used to investigate their capabilities for solving combinatorial optimization problems and presenting some preliminary results.

### 4.1 NETWORK STRUCTURE AND DYNAMICS

In a multi-layered perceptron (MLP) the nodes are arranged in layers, and the connections are not allowed to cross a layer, i.e., there are connections between the inputs of the network and the nodes in the first layer and between subsequent layers only. This implies that the inputs of a node in the first layer correspond to the inputs of the network, while the inputs of the nodes in a higher layer are the outputs of the nodes in the preceding layer. The outputs of the nodes in the highest layer form the outputs of the network. The nodes that are not output nodes are called the hidden nodes, and the corresponding layers the hidden layers.

The inputs and the connection strengths of a multi-layered perceptron are in general real valued. The outputs of the nodes can also be real valued, depending on the choice of the response functions that are used. Let $\Sigma \subseteq \mathbb{R} \rightarrow \mathbb{R}$ be a collection of response functions. Then we speak of $\Sigma$-MLPs, if the nodes all use a response function in $\Sigma$. The following definition identifies all possible functions computed by $\Sigma$-MLPs.

**Definition 4.1.** Let $m, N, K \in \mathbb{N}$ and let $A_N$ denote the set of all affine functions from $\mathbb{R}^N$ to $\mathbb{R}$ defined by

$$A_N = \{f : \mathbb{R}^N \rightarrow \mathbb{R} | f(x) = a \cdot x + b, x \in \mathbb{R}^N, a \in \mathbb{R}^N, b \in \mathbb{R}\}.$$ 

Then the set $\Sigma$-$R_{m,N,K} \subseteq \mathbb{R}^N \rightarrow \mathbb{R}^K$ of all functions that can be computed by a $\Sigma$-MLP with $m$ layers, $N$ inputs, and $K$ outputs, is defined recursively by

$$\Sigma$-$R_{1,N,K} = \{(\gamma_1 \circ f_1, \ldots, \gamma_K \circ f_K) | \gamma_i \in \Sigma, f_i \in A_N, i = 1, \ldots, K\},$$


and, for $m > 1$, 
\[
\Sigma R_{m,N,K} = \{g \circ h \mid g \in \Sigma R_{1,L,K}, h \in \Sigma R_{m-1,N,L}, L \in \mathbb{N}\}.
\]

In this paper we only consider MLPs that use the hard-limiting response function $\theta$, which are denoted by $\theta$-MLPs or simply MLPs. The output of the hard-limiting response function is 1, if its input is positive or zero, and 0 otherwise. Applications of multi-layered perceptrons usually consider a certain type of sigmoidal response function, which is often some kind of continuous approximation of the hard-limiting response function, such as the well-known logistic function.

Before we discuss the use of MLPs in combinatorial optimization we briefly discuss their classification capabilities.

### 4.2 CLASSIFICATION CAPABILITIES

The capabilities of MLPs ($= \theta$-MLPs) can be studied by considering them for solving classification problems. To this end we first examine the classification capabilities of MLPs with one output node.

**Definition 4.2.** A subset $V \subseteq \mathbb{R}^N$ is classified with a $\theta$-MLP with $N$ inputs and one output node represented by $f : \mathbb{R}^N \rightarrow \{0, 1\}$, if
\[
f(x) = \begin{cases} 1 & \text{if } x \in V, \\ 0 & \text{if } x \notin V. \end{cases}
\]

Thus $V$ is classified with an MLP represented by the function $f$, if the decision region $J(f)$ of this MLP equals $V$, where
\[
J(f) = \{x \in \mathbb{R}^N \mid f(x) = 1\}.
\]

The following definition introduces the collection of classifiable subsets, where we distinguish between the number of layers used in the network.

**Definition 4.3.** Let $N \in \mathbb{N}$ and $m \in \mathbb{N}$. Then the collection of subsets of $\mathbb{R}^N$ that can be classified with an MLP with $m$ layers (an $m$-LP) is denoted by $C_{m,N}$.

In short, $C_{m,N} = \{J(f) \mid f \in \theta R_{m,N,1}\}$. Usually, we abbreviate $C_{m,N}$ to $C_m$, except for those cases where $N$ is explicitly needed. In order to be able to present some results that characterize $C_m$ we introduce a number of collections of subsets of $\mathbb{R}^N$.

**Definition 4.4.** The collection of closed affine halfspaces $H$, the collection of open and closed affine halfspaces $\tilde{H}$, the collection of pseudo polyhedra $\tilde{P}$, and the collection of unions of pseudo polyhedra $\tilde{U}$, are defined by
\[
H = \{V \subseteq \mathbb{R}^N \mid \exists a \in \mathbb{R}^N \setminus \{0\} \exists b \in \mathbb{R} : V = \{x \in \mathbb{R}^N \mid a \cdot x + b \geq 0\}\},
\]
\[
\tilde{H} = \{V \subseteq \mathbb{R}^N \mid V \in H \lor V^+ \in H\},
\]
\[
\tilde{P} = \{V \subseteq \mathbb{R}^N \mid V = \bigcap_{i=1}^K W_i, W_i \in \tilde{H}, K \in \mathbb{N}_0\},
\]
\[
\tilde{U} = \{V \subseteq \mathbb{R}^N \mid V = \bigcup_{i=1}^L V_i, V_i \in \tilde{P}, L \in \mathbb{N}_0\},
\]
respectively.
A polyhedron \( V \in P \) is the intersection of a finite collection of closed affine halfspaces. Therefore, all its bounds, usually called faces, belong to the set. A pseudo polyhedron \( V \in \tilde{P} \) is the intersection of a finite collection of closed or open affine halfspaces and can have faces belonging to the set and faces belonging to its complement \( V^* \). The collection \( \tilde{U} \) can be viewed as the collection of all subsets of \( \mathbb{R}^N \) that have a finite number of piece-wise linear bounds.

We can now state the following result.

**Theorem 4.1 (Zwietering [1994]).** The classification capabilities of MLPs with \( N \) inputs and 1, 2, or \( m \) layers, \( m \geq 3 \) can be characterized by

\[
C_1 = H \cup \{\emptyset, \mathbb{R}^N\}, \quad (5)
\]
\[
\tilde{P} \subset C_2 \subset \tilde{U}, \quad (6)
\]
\[
C_m = \tilde{U}, \quad m \geq 3, \quad (7)
\]

respectively.

For a proof of Theorem 4.1 we refer to [Zwietering, 1994]. The same results can be obtained from [Gibson & Cowan, 1990] and [Gibson, 1993]. In [Zwietering, 1994] a more detailed characterization of \( C_2 \) is presented, by giving necessary conditions for a subset to be classifiable with a 2LP as well as sufficient conditions. The necessary conditions can be used to show that for instance the subsets presented in Figure 1 do not belong to \( C_2 \), i.e., they cannot be classified with a 2LP. On the other hand, using the sufficient conditions, one can show that the subsets presented in Figure 2 are members of \( C_2 \). For the details we refer to [Zwietering, 1994].

![Figure 1: Three subsets in that cannot be classified with a 2LP.](image1)

![Figure 2: Three subsets that can be proved to be in \( C_2 \).](image2)
Next, we consider more general classification problems. In particular we study classification problems that allow for the labels to be represented by Boolean variables. We call these problems combinatorial classification problems, defined as follows.

**Definition 4.5.** A *combinatorial classification problem* is given by a 3-tuple \((\Omega, L, \Gamma)\), where, (i) for some \(N \in \mathbb{N}\), \(\Omega \subseteq \mathbb{R}^N\) denotes a set of objects that must be classified, (ii) for some \(K \in \mathbb{N}\), \(L \subseteq \{0, 1\}^K\) denotes a set of labels, and (iii) \(\Gamma\) denotes a collection of subsets of \(\Omega\), one for each label, expressed by \(\Gamma = \{\Omega_l \subseteq \Omega \mid l \in L\}\). For a given object \(x \in \Omega\), the problem is to find a label \(l \in L\) such that \(x \in \Omega_l\). It is assumed that \(\bigcup_{l \in L} \Omega_l = \Omega\), which guarantees that for each \(x \in \Omega\) this problem can be solved. \(\square\)

The following definition formalizes what we understand by solving a combinatorial classification problem with an MLP.

**Definition 4.6.** A combinatorial classification problem given by \((\Omega, L, f)\) with \(\Omega \subseteq \mathbb{R}^N\) and \(L \subseteq \{0, 1\}^K\), for some \(N, K \in \mathbb{N}\), is solved by a \(B\)-MLP with \(N\) inputs and \(K\) outputs represented by the function \(f : \mathbb{R}^N \rightarrow \{0, 1\}^K\), if \(f(x) \subseteq L\) and \(x \in \bigcap_{l \in L} \Omega_l\), for all \(x \in \Omega\). \(\square\)

The classification of a single subset \(V \subseteq \mathbb{R}^N\) discussed above can be viewed as a special case of the general combinatorial classification problem by taking \(\Omega = \mathbb{R}^N\), \(L = \{0, 1\}\), and \(\Gamma = \{\Omega_0, \Omega_1\}\), with \(\Omega_1 = V\) and \(\Omega_0 = V^* = \mathbb{R}^N \setminus V\). In Theorem 4.2 below we essentially show that every combinatorial classification problem can be decomposed into a number of single subset classification problems.

**Theorem 4.2 (Zwietering [1994]).** Let \((\Omega, L, \Gamma)\) be a combinatorial classification problem with \(\Omega \subseteq \mathbb{R}^N\), for some \(N \in \mathbb{N}\), \(L \subseteq \{0, 1\}^K\), for some \(K \in \mathbb{N}\), and \(\Gamma = \{\Omega_l \subseteq \Omega \mid l \in L\}\). Let \(m \in \mathbb{N}\). Then an MLP with \(m\) layers, \(N\) inputs, and \(K\) outputs, represented by \(f \in \Theta\)-R\(_{m,N,K}\), solves \((\Omega, L, \Gamma)\), if and only if there exists a combinatorial classification problem represented by \((\tilde{\Omega}, \tilde{L}, \tilde{\Gamma})\), with \(\tilde{\Gamma} = \{\tilde{\Omega}_l \subseteq \tilde{\Omega} \mid l \in \tilde{L}\}\), that satisfies the following conditions.

(i) \(\bigcup_{l \in \tilde{L}} \tilde{\Omega}_l = \tilde{\Omega} = \Omega\).

(ii) \(\tilde{L} \subseteq L\).

(iii) \(\tilde{\Omega}_l \subseteq \Omega_l\), for all \(l \in \tilde{L}\).

(iv) \(\tilde{\Omega}_l \cap \tilde{\Omega}_k = \emptyset\), for all \(l, k \in \tilde{L}, l \neq k\).

(v) \(\tilde{V}_i^{(q)} = f_i(\tilde{V}_i) \cap \tilde{\Omega}_l\), for all \(i = 1, \ldots, K\), where for all \(i = 1, \ldots, K\) and \(q = 0, 1\), the set \(\tilde{V}_i^{(q)}\) is defined by

\[
\tilde{V}_i^{(q)} = \bigcup_{l \in \tilde{L}, l_i = q} \tilde{\Omega}_l.
\]

The set \(\tilde{V}_i^{(q)}\) denotes the set of all objects in \(\tilde{\Omega}\) that can be uniquely labeled, with respect to \(\tilde{\Gamma}\), with a label that has a \(q \in \{0, 1\}\) on its \(i\)-th position.

Combining Theorem 4.2 and Theorem 4.1 yields a necessary and sufficient condition for a combinatorial classification problem to be solvable by an MLP.

**Corollary 4.1.** Let \((\Omega, L, \Gamma)\) be a combinatorial classification problem with \(\Omega \subseteq \mathbb{R}^N\), for some \(N \in \mathbb{N}\), \(L \subseteq \{0, 1\}^K\), for some \(K \in \mathbb{N}\), and \(\Gamma = \{\Omega_l \subseteq \Omega \mid l \in L\}\). Then there exists
an MLP that solves \((\Omega, L, \Gamma)\), if and only if there exists a combinatorial classification problem represented by \((\hat{\Omega}, \hat{L}, \hat{\Gamma})\), with \(\hat{\Gamma} = \{\hat{\Omega}_l \mid l \in \hat{L}\}\), that satisfies the Conditions (i), (ii), (iii), and (iv), given in Theorem 4.2, and such that for all \(i = 1, \ldots, K\), there exists a subset \(V \in \hat{U}\) with \(\hat{V}^{(i)}_0 \subseteq V \subseteq (\hat{V}^{(i)}_0)^* = \mathbb{R}^N \setminus \hat{V}^{(i)}_0\), where for all \(i = 1, \ldots, K\) and \(q = 0, 1\), the set \(\hat{V}^{(i)}_q\) is defined by (8).

Although of theoretical interest, the result of Corollary 4.1 is practically of little use, due to the fact that the conditions posed on \((\hat{\Omega}, \hat{L}, \hat{\Gamma})\) are not easily verified. Below we present a sufficient condition that is much easier to verify.

**Corollary 4.2.** Let \((\Omega, L, \Gamma)\) be a combinatorial classification problem with \(\Omega \subseteq \mathbb{R}^N\), for some \(N \in \mathbb{N}\), \(L \subseteq \{0, 1\}^K\), for some \(K \in \mathbb{N}\), \(\Gamma = \{\Omega_l \subseteq \Omega \mid l \in L\}\), and \(\Omega_l \in \hat{U}\), for all \(l \in L\). Then there exists a 3LP represented by an \(f \in \theta \cdot \mathcal{R}_{3,N,K}\) that solves \((\Omega, L, \Gamma)\).

### 4.3 COMBINATORIAL OPTIMIZATION

In this section we show that MLPs can be used to solve combinatorial optimization problems of the type presented in Definition 2.2. The idea is to translate a combinatorial optimization problem into a classification problem, and then to use the results of the previous section. The proposed translation is straightforward.

**Definition 4.7.** Let \((\mathcal{D}, \mathcal{S}, \mathcal{F}, c)\) be a combinatorial optimization problem. Then the combinatorial classification problem corresponding to this combinatorial optimization problem is given by \((\Omega, L, \Gamma)\), with \(\Omega, L,\) and \(\Gamma\) defined by

(i) \(\Omega = \mathcal{D}\), i.e., the set of objects corresponds to the set of instance defining parameters,

(ii) \(L = \bigcup_{x \in \mathcal{D}} \mathcal{F}(x)\), i.e., the set of labels corresponds to the set of all feasible solutions, and

(iii) \(\Gamma = \{\Omega_y \mid y \in L\}\), where for each \(y \in L\), the subset \(\Omega_y \subseteq \mathcal{D}\) denotes the set of instance defining parameters for which \(y\) is a feasible, optimal solution, and is given by

\[
\Omega_y = \{x \in \mathcal{D} \mid y \in \mathcal{F}(x) \land \forall z \in \mathcal{F}(x) : c(y; x) \leq c(z; x)\}.
\]

One can verify that if the combinatorial optimization problem \((\mathcal{D}, \mathcal{S}, \mathcal{F}, c)\) satisfies the requirements of Definition 2.2, its corresponding combinatorial classification problem \((\Omega, L, \Gamma)\) satisfies the requirements of Definition 4.5. Furthermore, the combinatorial optimization problem and its corresponding combinatorial classification problem are equivalent, in the sense that for each instance defining parameter \(x \in \mathcal{D}\), the problem of finding a feasible solution \(y \in \mathcal{F}(x)\), such that \(c(y; x) \leq c(z; x)\), for all \(z \in \mathcal{F}(x)\), is equivalent to finding a label \(y \in L\), such that \(x \in \Omega_y\). Consequently, both the considered combinatorial optimization problem and its corresponding combinatorial classification problem have the same MLP-complexity, with respect to the required number of layers and nodes. This implies that it suffices to study the MLP-complexity of the combinatorial classification problem corresponding to a given combinatorial optimization problem, which can be done using the results of the previous section. The first result is a sufficient condition for the existence of an MLP that solves the combinatorial optimization problem at hand, which is obtained from Corollary 4.2. Recall that \(P, \hat{P},\) and \(\hat{U}\), denote the set of all polyhedrons, the set of all pseudo polyhedrons, and the set of all unions of pseudo polyhedrons, respectively, as defined in Definition 4.4.
Theorem 4.3 (Zwietering [1994]). Let $(\mathcal{D}, \mathcal{S}, \mathcal{F}, c)$ be a combinatorial optimization problem with $\mathcal{D} \subseteq \mathbb{R}^N$ and $\mathcal{F}(\cdot) \subseteq \{0, 1\}^K$, for some $N, K \in \mathbb{N}$, and let $(\mathcal{Q}, \mathcal{L}, \Gamma)$ be its corresponding combinatorial classification problem. Let the subsets $\Phi_y \subseteq \mathcal{D}$ and $\Psi_{y,z} \subseteq \mathcal{D}$ be defined by

$$\Phi_y = \{x \in \mathcal{D} \mid y \in \mathcal{F}(x)\}, \quad \Psi_{y,z} = \{x \in \mathcal{D} \mid c(y; x) \leq c(z; x)\},$$

for all $y \in \mathcal{L}$ and $y, z \in \mathcal{L}$, respectively, and let $\Phi_y \subseteq \mathcal{U}$ and $\Psi_{y,z} \subseteq \mathcal{U}$, for all $y, z \in \mathcal{L}$. Then there exists a 3LP that solves $(\mathcal{D}, \mathcal{S}, \mathcal{F}, c)$.

The condition given in Theorem 4.3 is not a necessary condition. One can easily find artificially constructed combinatorial optimization problems that do not satisfy the condition of Theorem 4.3, and that are still solvable by an MLP. A necessary and sufficient condition can be derived from Corollary 4.1, but this leads to a condition that is not easily verifiable. The reason for giving the formulation used in Theorem 4.3 is that we do not expect that there exist real-world combinatorial optimization problems that do not satisfy the condition of Theorem 4.3 and that are still solvable by an MLP.

Theorem 4.3 defines a class of combinatorial optimization problems that can be solved by a 3LP. However, the theorem does not yield the minimal number of required layers and nodes, nor does it present a construction of a 3LP that solves the problem at hand. Zwietering [1994] addresses these issues for a subclass of combinatorial optimization problems, which contains all problems that satisfy the following conditions.

**Condition 1.** The set of feasible solutions is parameter independent, i.e., $\mathcal{F}(x) = \mathcal{F}(x')$, for all $x, x' \in \mathcal{D}$. In the remainder we use $\mathcal{F}$ to denote the set of feasible solutions.

**Condition 2.** For all $y \in \mathcal{F}$, $c(y; \cdot) : \mathbb{R}^N \to \mathbb{R}$ is an affine function, i.e., $c(y; x) = a(y)^T x + b(y)$, for some $a(y) \in \mathbb{R}^N$, $b(y) \in \mathbb{R}$, and all $x \in \mathbb{R}^N$.

**Condition 3.** The set of instance defining parameters forms a polyhedron in $\mathbb{R}^N$, i.e., $\mathcal{D} \subseteq \mathcal{P}$.

**Condition 4.** For each feasible solution, there exists a ball of parameters that all define instances for which this solution is strictly optimal, i.e.,

$$\forall y \in \mathcal{F} \exists x \in D^o \forall z \in \mathcal{F} \setminus \{y\} : c(y; x) < c(z; x).$$

It is easily verified that each combinatorial optimization problem that satisfies the Conditions 1, 2, 3, and 4, also satisfies the conditions of Theorem 4.3. Furthermore, by choosing $\mathcal{D} = \mathbb{R}^N$ and $b(y) \equiv 0$ we obtain the class of discrete linear optimization problems distinguished by Savage and others; see also Savage [1973] and Papadimitriou & Steiglitz [1982].

The results presented in [Zwietering, 1994] concern a general construction of a 3LP that solves a combinatorial optimization problem satisfying the Conditions 1, 2, 3, and 4, and which is minimal with respect to the number of first-layer nodes. Furthermore, we have derived necessary conditions for the existence of a 2LP that solves a combinatorial optimization problem of the introduced subclass. Again these results were based on the translation of combinatorial optimization problems into combinatorial classification problems, using that $\mathcal{Q}_y$, defined by (9) is a full-dimensional polyhedron in case that the Conditions 1, 2, 3, and 4 are satisfied, and exploiting some concepts from local search.

In case that Condition 1 does not hold, the considered combinatorial optimization problem may still be solvable by a 3LP, but the construction is far less straightforward and not necessarily
minimal with respect to the number of first-layer nodes [Zwietering, Aarts & Wessels, 1991].

Note that Condition 1 may exclude some interesting combinatorial optimization problems, for instance the knapsack problem discussed in Zwietering, Aarts & Wessels [1991].

In a strict sense, Condition 2 is not essential, since most results may be extended to the case that the subsets \( \{ x \in \mathbb{R}^N \mid c(y; x) \leq c(z; x) \} \) correspond to closed affine halfspaces, but this does not seem to be a serious extension, in the sense that the set of admissible combinatorial optimization problems is enlarged significantly. Condition 2 is required because most of the analysis is based on affine halfspaces. However, we are aware of the fact that it is a restrictive condition, and for instance excludes combinatorial optimization problems in which the cost function contains the maximum function or the absolute value function.

In case that Condition 3 does not hold, because \( D \notin \mathcal{P} \), one might consider solving \((D', \mathcal{S}, \mathcal{F}, c)\), with \( D' \supseteq D \) such that \( D' \in \mathcal{P} \). For instance, if \( D \in \bar{U} \), then \( D' = \text{conv.hull}(D) \) indeed satisfies \( D' \supseteq D \) and \( D' \in \mathcal{P} \); see Zwietering [1994]. Obviously, any MLP that solves \((D', \mathcal{S}, \mathcal{F}, c)\), solves \((D, \mathcal{S}, \mathcal{F}, c)\) also. However, enlarging the set of instance defining parameters may affect the complexity of the problem with respect to the required number of hidden nodes and the required number of layers, respectively.

In case that Condition 4 does not hold, because \( D^0 = \emptyset \), one determines the smallest, in dimension, affine subspace that contains \( D \), and considers the problem with respect to that subspace. Obviously, \( D^0 \neq \emptyset \) with respect to this determined subspace. In case that Condition 4 does not hold, although \( D^0 \neq \emptyset \), one might consider solving \((D, \mathcal{S}, \mathcal{F}', c)\), with \( \mathcal{F}' = \{ y \in \mathcal{F} \mid \exists x \in D^0 \forall z \in \mathcal{F}\setminus \{y\} : c(y; x) < c(z; x) \} \), since it is easily shown that any MLP that solves \((D, \mathcal{S}, \mathcal{F}', c)\) also solves \((D, \mathcal{S}, \mathcal{F}, c)\).

We have applied the above mentioned results to five combinatorial optimization problems, all member of the subclass defined by the Conditions 1, 2, 3, and 4: SORTING, MINIMUM COST SPANNING TREE, SHORTEST NETWORK PATH, SHORTEST NETWORK ROUTE, and DISCRETE DYNAMIC LOTSIZING; see Zwietering [1994]. The minimal number of first-layer nodes of any MLP that solves SORTING was shown to be polynomial in the number of inputs. Furthermore, we presented a construction of an 3LP with a polynomial number of nodes for SORTING and proved that the minimal number of layers required by any MLP that solves SORTING is three; see also Zwietering, Aarts & Wessels [1994]. In [Zwietering, Aarts & Wessels, 1994], we discuss the possibility that there exists a 2LP for INTEGER SORTING, if one considers the case where the numbers to be sorted are taken from a bounded set of integers.

Similarly, we found a construction of an MLP that solves MINIMUM COST SPANNING TREE, has a minimal sized first layer, and has a total number of nodes that is polynomial in the number of inputs. For the other three problems we could show that the minimal number of nodes was exponential in the number of inputs; see Zwietering [1994]. Finally, we have shown that three layers is minimal for solving DISCRETE DYNAMIC LOTSIZING with an MLP; see also Zwietering, Van Kraaij, Aarts & Wessels [1991].

5 DISCUSSION

The Boltzmann machine is a stochastic neural network model that can be used to handle combinatorial optimization problems. Compared to Hopfield networks, Boltzmann machines have
the advantage that they can escape from poor locally optimal configurations. A disadvantage is
the slow convergence of the self-organization.

The importance of Boltzmann machines for combinatorial optimization lies in its significance
as a massively parallel approach to simulated annealing. Simulations support this significance
since a substantial speed up can be achieved by implementations on multiprocessor systems.
The significance is further increased when Boltzmann machines are implemented on special-
purpose hardware or general neurocomputers. Examples of VLSI implementations are given by Hirai [1993]. An optoelectronic implementation is given by Lalanne, Rodier, Chavel, Belhaire
& Garda [1993]. Recent research on extensions of the model concentrates on asynchronous and
asymmetric Boltzmann machines; see [Ferscha & Haring, 1991] and [Apolloni & De Falco,
1991], respectively. Extensions of the binary states to multivalued states [Lin & Lee, 1991] and
even continuous states [Beiu, Ioan, Dumbrava & Robciuc, 1992] are also being investigated.

We have shown that in theory multi-layered perceptrons can deterministically solve combinatorial
optimization problems. However, already for relatively easy problems the minimal required
size of such a network is exponential in the number of inputs. This makes this approach not
very practical. Promising are the results of adaptation in multi-layered perceptrons to solve a
combinatorial optimization problem with uncertainties; see Zwietering, Van Kraaij, Aarts &
Wessels [1991].

References

AARTS, E.H.L., AND J. KORST [1989], Simulated Annealing and Boltzmann Machines, John Wiley &
Sons, New York.

AARTS, E.H.L., AND H.P. STEHOUWER [1993], Neural networks and the travelling salesman problem,
in: S. Gieie and B. Kappen (eds.), Proceedings of the International Conference on Artificial
Neural Networks, Springer-Verlag, 950–955.

APOLLONI, B., AND D. DE FALCO [1991], Learning by asymmetric parallel Boltzmann machines, Neural
Computation 3, 402–408.

BAUM, E.B. [1986], Towards practical neural computation for combinatorial optimization problems, in:
J.S. Denker (ed.), Neural Networks for Computing (Snowbird 1986), 53–58.

BEIU, V., D.C. IOAN, M.C. DUMBRAVA, AND O. ROBCIUC [1992], Physical Fields Determination Using

FELDMAN, J.A., AND D.H. BALLARD [1982], Connectionist models and their properties, Cognitive
science 6, 205–254.

FERSCHA, A., AND G. HARING [1991], Asynchronous parallel Boltzmann machines for combinatorial

NP-Completeness, W.H. Freeman and co.

GIBSON, G.J. [1993], A combinatorial approach to understanding perceptron capabilities, IEEE Trans-
actions on Neural Networks 4, 989–992.

GIBSON, G.J., AND C.F.N. COWAN [1990], On the decision regions of multilayer perceptrons, Proceed-
ings of the IEEE 78, 1590–1594.

GOLES-CHACC, E., F. FOGELMAN-SOULIE, AND D. PELLEGRIN [1985], Decreasing energy functions as

HECHT-NIELSEN, R. [1990], Neurocomputing, Addison-Wesley, New York.

HERTZ, J., A. KROGH, AND R.G. PALMER [1991], Introduction to the Theory of Neural Computation,
Addison-Wesley.


HIRAI, Y. [1993], Hardware implementation of neural networks in Japan, Neurocomputing 5, 3–16.


PARBERRY, I., AND G. SCHNITGER [1989], Relating Boltzmann machines to conventional models of computing, Neural Networks 2, 59–67.


ROSENBLATT, R. [1958], The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review 65, 386–408.


