The design and complexity of exact multi-layered perceptions

Citation for published version (APA):

Document status and date:
Published: 01/01/1991

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 author's 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.

Take down policy
If you believe that this document breaches copyright please contact us:
openaccess@tue.nl
providing details. We will immediately remove access to the work pending the investigation of your claim.
Memorandum COSOR 91-08

The Design and Complexity of
Exact Multi-Layered Perceptrons

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

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB Eindhoven
The Netherlands

ISSN: 0926-4493

Eindhoven, April 1991
The Netherlands
The Design and Complexity of Exact Multi-Layered Perceptrons

P.J. Zwietering¹, E.H.L. Aarts¹,² and J. Wessels¹

1. Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands
2. Philips Research Laboratories, P.O. Box 80.000, 5600 JA Eindhoven, the Netherlands

Abstract

We investigate the network complexity of multi-layered perceptrons for solving exactly a given problem. We limit our study to the class of combinatorial optimization problems. It is shown how these problems can be reformulated as binary classification problems and how they can be solved by multi-layered perceptrons.

Keywords: Combinatorial Optimization, Classification, Complexity, Exact Network Configurations, Multi-Layered Perceptrons, Neural Networks

Introduction

There is an increasing interest in the use of multi-layered perceptrons for various kinds of applications because of their adaptive and associative capabilities. However, to the best of our knowledge there does not exist a methodical approach for the design of multi-layered perceptrons. The main objective of this paper is the development of a methodical approach for the design of the network configuration of a multi-layered perceptron for a specific problem. Once the network configuration is defined, the weights can be found by one of the learning algorithms which are abundantly available. We distinguish three steps in the design of a multi-layered perceptron:

1. Specification of the number of inputs, the number of outputs and the required input-output behavior of the network.

2. Specification of the network configuration: the number of hidden layers and the number of units in each hidden layer.

3. Determination of the appropriate weights between the units such that the required input-output behavior is obtained.

In practice the first step of the design process is usually not very hard; for most problems the number of inputs and outputs and the desired input-output behavior follow directly
from the problem formulation. Usually, one chooses a network configuration by some rule of thumb and concentrates on the third stage. For the third stage all sorts of learning algorithms have become available that calculate an appropriate set of weights [9, 11]. Most of these algorithms produce a set of weights for which the input-output behavior approximates the specification. If the result is unsatisfactory, the configuration of the network is changed and a new set of weights is determined using the selected learning algorithm. In this way the designing becomes more or less a trial-and-error approach and it is not sure that the required input-output behavior is ever reached. In fact it even is not sure that the required behavior is reachable.

So far, the literature presents only few results regarding a methodical approach to the design of multi-layered perceptrons. Most papers study the capabilities of multi-layered perceptrons in general, without giving methods for finding the network configurations of perceptrons for solving a specific problem. Furthermore, many results concern the approximate classification capabilities of two-layered perceptrons. A number of authors show that a two-layered perceptron is capable of classifying a given subset within arbitrary precision (see Cybenko [2], Funahashi [4] and Hornik et al. [6]).

There are three reasons why we did not use these results. The main reason is that for obtaining the exact solution of combinatorial optimization problems one requires the exact solution of the corresponding classification problems. The second reason is that a learning algorithm can only produce weights which converge to the required weights if the problem is exactly solvable by a multi-layered perceptron of the given configuration. The third reason is that the papers mentioned above do not give insight in the precise number of hidden units that is needed for a specific problem. A recent paper by Cybenko [3] does shed some light on this matter, in the sense that an upper-bound on the necessary number of units is given, but this bound is too rough in many cases. Note also that the results obtained by Baum and others (see [1] and the references therein) about the exact capabilities of multi-layered perceptrons with binary inputs cannot be used here, since we consider networks with real valued inputs.

The aim of the present paper is to replace the usual trial-and-error approach by a methodical approach for the design of the exact network configuration that is needed for a specific problem. This implies that the iterative adaptation of the network configuration becomes superfluous. Moreover, if a learning algorithm is used to calculate the weights between the units in the exact network configuration, the resulting input-output behavior is directly attributed to the learning algorithm, which implies that the quality of the learning algorithm can be measured.

We limit ourselves in this paper to the class of combinatorial optimization problems, but the approach can also be applied to other kinds of problems. We have chosen the class of combinatorial optimization problems for our design study because of the following reasons:

- Many problems can be formulated as a combinatorial optimization problem.
The space and time complexity of a multi-layered perceptron that exactly solves a given combinatorial optimization problem can be compared with the space and time complexity of a traditional parallel or sequential algorithm, which gives insight in the quality of our approach.

The intention is to design multi-layered perceptrons, with real valued inputs and binary valued outputs, that exactly solve a given combinatorial optimization problem, i.e. the network must give the optimal solution for each instance of the problem. Such a multi-layered perceptron is called an exact multi-layered perceptron for a given combinatorial optimization problem.

Our strategy for finding an exact multi-layered perceptron for a given combinatorial optimization problem consists of three parts. To some extent these three parts correspond to the three steps in the general design procedure given above, and are treated in Sections 1, 2 and 3, respectively.

In the first part (Section 1) we reformulate a given combinatorial optimization problem as a 0-1-formulated classification problem. This reformulation is necessary since a multi-layered perceptron is essentially a classification device. A 0-1-formulated classification problem can be decomposed into a number of binary classification problems if the subsets that define the classification problem are disjoint. In the second part (Section 2), we discuss the general problem of exactly solving a binary classification problem by a multi-layered perceptron with one output only. The main issue is to find the exact configuration of the network. After a detailed description of the considered networks, we derive a necessary and sufficient condition for a subset of $\mathbb{R}^n$ such that the corresponding binary classification problem can be solved exactly by an $m$-layered perceptron, for $m \geq 3$. In the third part of the design (Section 3), the results of the previous two parts are combined. It is shown that a combinatorial optimization problem can be solved exactly by a three-layered perceptron if the cost function and feasibility constraints are piece-wise linear in the input data of the problem. In some cases a four-layered perceptron is proposed for efficiency reasons. Both the number of hidden units and the values of the weights can be easily found. In Section 4 we evaluate the exact multi-layered perceptrons found by studying the complexity of simulations of these exact multi-layered perceptrons on a PRAM. The paper ends by giving some concluding remarks and references.

1 Combinatorial Optimization And Binary Classification

In this section we present a first approach to the problem of finding an exact multi-layered perceptron for a given combinatorial optimization problem, a multi-layered perceptron that exactly solves that problem. The main idea is to reformulate the combinatorial optimization problem as a classification problem. To this end we make a distinction between a problem and an instance of a problem, for both combinatorial optimization problems and classification problems. In the reformulation, instances of the given combinatorial opti-
nization problem are mapped onto instances of the corresponding classification problem. The resulting classification problem is then solved exactly by a multi-layered perceptron.

A classification problem must satisfy some basic conditions to be exactly solvable by a multi-layered perceptron with real valued inputs and binary valued outputs. The latter is typical for the multi-layered perceptrons under consideration. Firstly, an instance must correspond to a fixed number of real variables and secondly, a solution must be representable using a 0–1-formulation. These requirements can be directly transferred to the combinatorial optimization problem at hand, which implies that it must be put in a 0–1-formulation.

In the second part of this section we show that in general a classification problem that is given in a 0–1-formulation can be decomposed into a number of binary classification problems, when the subsets that define the classification problem are disjoint. The resulting binary classification problems can then be solved independently, which will be discussed in Section 2.

A classification problem can be formalized as a pair \( (\Omega, \Gamma) \), where \( \Omega \) denotes the set of objects that must be classified, the set of instances of the classification problem, and \( \Gamma \) a collection of subsets of \( \Omega \): \( \Gamma = \{ \Omega_l \subseteq \Omega \mid l \in L \} \), for some set of labels \( L \). For each instance \( o \in \Omega \) the problem is to classify \( o \) as a member of one of the subsets \( \Omega_l \in \Gamma \), i.e we must find a label \( l \in L \) such that \( o \in \Omega_l \). We assume \( \Omega \subseteq \bigcup_{l \in L} \Omega_l \), which guarantees that for every instance \( o \in \Omega \) this problem has a solution. We say that \( (\Omega, \Gamma) \) is a binary classification problem if \( |L| = 2 \).

![Figure 1: A Classifier System represented as a black box.](image)

Assume for a moment that every given classification problem \( (\Omega, \Gamma) \) can be solved by a classifier function \( \delta : \Omega \rightarrow L \) represented by the black box in Figure 1. For each object \( o \in \Omega \) presented to the classifier function it yields the correct label \( l = \delta(o) \) satisfying \( o \in \Omega_l \). The idea now is to exploit such a classifier function (black box) for solving combinatorial optimization problems, by reformulating them as classification problems. The next step is then to put a multi-layered perceptron inside the black box. First we need a formal definition of a combinatorial optimization problem.

A combinatorial optimization problem can be formalized as a tuple \( (I, S, F, c) \), where \( I \) denotes a set of problem instances. An instance \( x \in I \) denotes the 'input data' and gives enough information to solve the problem; \( S(x) \) denotes the finite set of possible solutions for instance \( x \), \( F(x) \subseteq S(x) \) denotes the set of feasible solutions and \( c(\cdot; x) : S(x) \rightarrow \mathbb{R} \) denotes the cost function. Given an instance \( x \in I \), the problem is to find a solution \( y \in F(x) \) such that \( c(y; x) \leq c(z; x) \) for all \( z \in F(x) \). We assume \( F(x) \neq \emptyset \) for all \( x \in I \),...
which guarantees that for every instance \( x \) this problem has a solution. For later use we define the set of all feasible solutions \( F = \bigcup_{x \in I} F(x) \).

The following example describes the well-known traveling salesman problem using the above formulation.

**Example 1 (Traveling Salesman Problem (TSP))**
In an instance of the TSP we are given an integer \( n > 0 \) and an \( n \times n \) matrix representing the distance between every pair of the \( n \) cities. A solution is denoted by a permutation \( \pi \) of the set \( \{1, \ldots, n\} \), where \( \pi(j) \) is to be interpreted as the city that is visited immediately after city \( j \). A permutation \( \pi \) represents a tour if and only if \( \pi \) is a cyclic permutation. The problem is to find a tour with shortest length. Hence, we come to the following definition of the tuple \((I,S,F,c)\):

\[
I = \{(n,D) \mid n \text{ an integer}, D \text{ an } n \times n \text{ distance matrix}\},
\]

\[
S(n,D) = \{\pi \mid \pi \text{ is a permutation of } \{1, \ldots, n\}\},
\]

\[
F(n,D) = \{\pi \mid \pi \text{ is a cyclic permutation of } \{1, \ldots, n\}\},
\]

\[
c(\pi; n, D) = \sum_{j=1}^{n} D_{\pi(j),j}.
\]

For a given \((n,D) \in I\) the problem is to find a \( \pi \in F(n, D) \) that minimizes \( c(\pi; n, D) \).

We reformulate a combinatorial optimization problem given by the tuple \((I,S,F,c)\) as a classification problem by mapping the instances of the combinatorial optimization problem on the instances of the corresponding classification problem: \( \Omega = I \). The labels on the subsets that partition \( \Omega = I \) will correspond to the solution that is optimal for the instances in that subset, i.e. we let \( L = F = \bigcup_{x \in I} F(x) \) and \( \Gamma = \{\Omega_y \mid y \in F\} \), where the subset \( \Omega_y \subseteq \Omega \) is given by:

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

(1)

It is not hard to see that for this choice of \( \Omega \) and \( \Gamma \) the combinatorial optimization problem \((I,S,F,c)\) and the classification problem \((\Omega, \Gamma)\) are equivalent: for each instance \( x \in I \) the problem of finding a solution \( y \in F(x) \) such that \( c(y;x) \leq c(z;x) \) for all \( z \in F(x) \) is equivalent to finding a label \( y \in F \) such that \( x \in \Omega_y \). Notice also that \( \Omega = I = \bigcup_{y \in F} \Omega_y \), since \( F(x) \neq \emptyset \) for all \( x \in I \). Next, we consider the problem of finding a multi-layered perceptron that exactly solves a given classification problem. We must impose some conditions on a classification problem to be able to solve it exactly by a multi-layered perceptron. Since we reformulate a combinatorial optimization problem as a classification problem before solving it by a multi-layered perceptron, the same conditions must be imposed on the combinatorial optimization problems.

If we want to construct a multi-layered perceptron for exactly solving a given classification problem \((\Omega, \Gamma)\), we have to choose a type of perceptron first. We only consider multi-layered perceptrons with real valued inputs and binary valued outputs. This implies that an instance of the classification problem must be representable with a fixed number of
real variables: $\Omega \subseteq \mathbb{R}^N$ for some fixed $N \in \mathbb{N}$. Furthermore, the labels (solutions) of the classification problem must be representable with a fixed number of binary variables: $L \subseteq \{0,1\}^K$, for some fixed $K \in \mathbb{N}$. The last condition is usually satisfied: we may assume without loss of generality that $L \subseteq \{0,1,\ldots,r\}$ for some $r \in \mathbb{N}$ and obviously $\{0,1,\ldots,r\} \subseteq \{0,1\}^K$ for some $K \in \mathbb{N}$.

The above conditions have direct implications for the combinatorial optimization problem that is reformulated as a classification problem in order to solve it by a multi-layered perceptron. It follows that we must have $I \subseteq \mathbb{R}^N$ and $F \subseteq \{0,1\}^K$, for some fixed $N, K \in \mathbb{N}$. For the sake of convenience we assume that the set of solutions is independent of the instance at hand, i.e. $S(x) = S$ for all $x \in I$. Then to satisfy the above condition $F \subseteq \{0,1\}^K$, it is sufficient to have a 0-1-formulation of the combinatorial optimization problem with $S \subseteq \{0,1\}^K$ for some $K \in \mathbb{N}$.

Below we give four examples of combinatorial optimization problems in a 0-1-formulation. In the first three examples the set of feasible solutions $F(x)$ equals for all instances $x$, i.e. $F(x) = F$ for all $x \in I$, which will simplify the construction of the multi-layered perceptron that exactly solves the problem (see Section 3).

**Example 2 (Traveling Salesman Problem on $n$ Cities)**

*For every matrix $x \in \mathbb{R}^{n \times n}$, $x_{ij}$ represents the distance between city $i$ and city $j$. Let $y_{ij} \in \{0,1\}$ be a 0-1-variable, where $y_{ij} = 1$ if the tour visits city $i$ at the $j^{th}$ position and $y_{ij} = 0$ otherwise. Then the problem can be formulated with:*

\[
I = \{ x \in \mathbb{R}^{n \times n} \mid x \geq 0 \}, \\
S = \{0,1\}^{n \times n}, \\
F = \{ y \in S \mid \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} = \sum_{j=1}^{n} y_{ij} = 1 \}, \\
c(y; x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} x_{ij} y_{ik} y_{j(k+1)},
\]

*where for a given $x \in I$ the problem is to find a $y \in F$ that minimizes $c(y; x)$. Note that this formulation satisfies the above conditions with $N = K = n^2$.*

**Example 3 (Sorting of $n$ Numbers)**

*The objective is to sort the numbers $x_1, \ldots, x_n \in \mathbb{R}$ in ascending order, i.e. we must find a permutation $\pi$ such that $x_{\pi(j)} \leq x_{\pi(j+1)}$ for all $j = 1, \ldots, n - 1$. By representing a permutation as a 0-1-matrix, we come to the following formulation. Let $y_{ij} \in \{0,1\}$ be a 0-1-variable, where $y_{ij} = 1$ if the number $x_j$ is to be put on the $i^{th}$ position and $y_{ij} = 0$ otherwise. Then the problem can be formalized as a tuple $(I, S, F, c)$, with:*

\[
I = \mathbb{R}^n, \\
S = \{0,1\}^{n \times n}, \\
F = \{ y \in S \mid \sum_{i=1}^{n} y_{ij} = \sum_{j=1}^{n} y_{ij} = 1 \},
\]
\[ c(y; x) = - \sum_{i=1}^{n} \sum_{j=1}^{n} i x_i y_{ij}, \]

where for a given \( x \in I \) the problem is to find a \( y \in F \) that minimizes \( c(y; x) \) (see also [13]). This formulation satisfies the above conditions with \( N = n \) and \( K = n^2 \).

**Example 4 (0-1 Knapsack Problem with \( n \) Weights)**

The objective is to find a selection of \( n \) items with weights \( x_1, \ldots, x_n \in \mathbb{R}_+ \) that fill a knapsack of size 1 as good as possible. The variable \( y_i \in \{0, 1\} \) flags the presence of the \( i \)th weight in the knapsack. Hence, we come to the following definition of \((I, S, F, c)\):

\[
\begin{align*}
I &= \{ x \in \mathbb{R}^n | x \geq 0 \}, \\
S &= \{0, 1\}^n, \\
F(x) &= \{ y \in S | \sum_{i=1}^{n} x_i y_i \leq 1 \}, \\
c(y; x) &= - \sum_{i=1}^{n} x_i y_i.
\end{align*}
\]

where for a given \( x \in I \) the problem is to find a \( y \in F \) that minimizes \( c(y; x) \). Note that \( F(x) \neq \emptyset \) for all \( x \in I \), since 0 \( \in F(x) \) for all \( x \). The formulation satisfies the above conditions with \( N = n \) and \( K = n \).

The next condition we must impose on a classification problem \((\Omega, \Gamma)\) to be able to solve it exactly by a multi-layered perceptron, is that the classifying subsets \( \Omega_i \) are disjoint. If this condition is satisfied the classification problem can be decomposed into a number of binary classification problems, see Theorem 1. The following step is then to find an exact multi-layered perceptron for each of the resulting binary classification problems (see Section 2). The exact multi-layered perceptrons for each of the binary classification problems can be found independently. Finally, these individual multi-layered perceptron can be put together to form an exact multi-layered perceptron for the original classification problem.

**Theorem 1** Let \((\Omega, \Gamma)\) be a classification problem with \( L \subseteq \{0, 1\}^K \) for some \( K \in \mathbb{N} \). Define for \( i = 1, \ldots, K \) and \( q = 0, 1 \) the set \( V_q^{(i)} \) by:

\[ V_q^{(i)} = \bigcup_{l \in L, l_i = q} \Omega_l, \] (2)

and assume \( \Omega_l \cap \Omega_k = \emptyset \) for all \( l \neq k \).

Then \( o \in \Omega_l \) if and only if \( o \in V_q^{(i)} \) for all \( i = 1, \ldots, K \).

**Proof**

\((\Rightarrow)\) Assume \( o \in \Omega_l \) for some \( l \in L \), then by definition (2) it follows directly that \( o \in V_q^{(i)} \) for all \( i = 1, \ldots, K \).

\((\Leftarrow)\) Assume \( o \in V_q^{(i)} \) for some \( l_i \in \{0, 1\}, i = 1, \ldots, K \). Since \( o \in \Omega \) it follows that \( o \in \Omega_k \).
for some \( k \in L \). If \( k_i \neq i \) for some \( i \), then by using the only-if-part (\( \Rightarrow \)), we would have:

\[
o \in V_i^{(i)} \cap V_{k_i}^{(i)} \subseteq \left( \bigcup_{r \in L, r_i = i} \Omega_r \right) \cap \left( \bigcup_{s \in L, s_i = k_i} \Omega_s \right) \subseteq \bigcup_{r \neq s} (\Omega_r \cap \Omega_s) = \emptyset,
\]

which is obviously impossible. Hence, \( k = l \) and the result follows.

The above theorem shows that a classification problem \((\Omega, \Gamma)\) with \( L \subseteq \{0, 1\}^K \) for some \( K \in \mathbb{N} \) and disjoint subsets \( \Omega_i \), can be decomposed into \( K \) binary classification problems \((\Omega, \Gamma_i)\), \( i = 1, \ldots, K \), with \( \Gamma_i = \{V_i^{(i)}, V_{k_i}^{(i)}\} \). These binary classification problems are well-defined as \( V_0^{(i)} \cup V_1^{(i)} = \bigcup_{p \in L} \Omega_p \supseteq \Omega \) for all \( i \). Furthermore it follows from the proof of Theorem 1 that \( V_0^{(i)} \cap V_1^{(i)} = \emptyset \) for all \( i \), which makes each of these binary classification problems suited for solving by a multi-layered perceptron (see Section 2).

In general the subsets \( \Omega_i \) of a given classification problem \((\Omega, \Gamma)\) do not have to be disjoint. If we want to use the decomposition method described above, we must transform \((\Omega, \Gamma)\) into a classification problem \((\Omega, \tilde{\Gamma})\) with disjoint subsets. This can be done by introducing a preference measure \( p : L \rightarrow \mathbb{R} \), satisfying \( p(l) \neq p(k) \) for all \( l \neq k \). We use the preference measure \( p \) to define the classification problem \((\Omega, \tilde{\Gamma})\) by \( \tilde{\Gamma} = \{\Omega_l | l \in L\} \), where \( \Omega_l \) is given by:

\[
\tilde{\Omega}_l = \Omega_l \setminus \bigcup_{k \in L, p(k) > p(l)} \Omega_k.
\]

It can be shown that \( \bigcup_{p \in L} \tilde{\Omega}_l = \bigcup_{p \in L} \Omega_p \supseteq \Omega \), which implies that \((\Omega, \tilde{\Gamma})\) is a well-defined classification problem. Furthermore, an object \( o \in \tilde{\Omega} \) is classified as a member of \( \tilde{\Omega}_l \) if and only if \( o \) is a member of \( \Omega_l \) and \( o \) is not a member of any subset \( \Omega_k \) that is more preferred. One can easily verify that \( \tilde{\Omega}_l \cap \tilde{\Omega}_k = \emptyset \) if \( l \neq k \), which shows that \((\Omega, \tilde{\Gamma})\) satisfies the conditions of Theorem 1.

For classification problems that are obtained as a reformulation of a given combinatorial optimization problem the subsets \( \Omega_y \) defined by (1) usually are not disjoint. This is due to the fact that most combinatorial optimization problems have a set of instances in which some instances have multiple optimal solutions. Adding a preference measure to the classification problem, as described above, yields a disjoint classification problem. Alternatively one can add the preference measure directly to the combinatorial optimization problem, which is then reformulated as classification problem already having disjoint subsets.

Let \((I, S, F, c)\) be a combinatorial optimization problem and \( p : F \rightarrow \mathbb{R} \) a preference function, \( p(y) \neq p(z) \) for \( y \neq z \). The combinatorial optimization problem \((I, S, F, (c, p))\) is obtained from \((I, S, F, c)\) by the addition of \( p \) to the objective in the following way: for each instance \( x \in I \) the problem is to find a \( y \in F(x) \) such that:

(i) \( c(y; x) \leq c(z; x) \) for all \( z \in F(x) \),

(ii) \( p(y) > p(z) \) for all \( z \in F(x) \) with \( c(y; x) = c(z; x) \).
This means that if we have more than one optimal solution for a given instance of the problem, we choose the most preferred optimal solution. Reformulating \((I, S, F, (c, p))\) as a classification problem \((\Omega, \Gamma)\) yields \(\Gamma = \{\Omega_y \subseteq \Gamma | y \in F\}\) with \(\Omega_y\) given by:

\[
\Omega_y = \{x \in I | \forall z \in F(x) \setminus \{y\} : c(y; x) < c(z; x) \lor (c(y; x) = c(z; x) \land p(y) > p(z))\}. \tag{4}
\]

We end this section with some remarks on the choice of the preference measure. In case of a 0–1-formulated classification problem or combinatorial optimization problem, we have that the set of solutions (labels) satisfies \(S \subseteq \{0, 1\}^K\) for some \(K \in \mathbb{N}\). A suitable preference measure is then given by:

\[
\forall y \in \{0, 1\}^K : p(y) = \sum_{i=1}^{K} 2^{K-i} y_i. \tag{5}
\]

It maps every \(y \in \{0, 1\}^K\) to the value of \(y\) when viewed as a binary number and induces the lexicographic order on \(\{0, 1\}^K\). If the solutions are represented by a two-dimensional 0–1-vector (see Example 2 and Example 3), we have a choice between several variants of the above given preference measure, the most obvious ones being:

\[
\forall y \in \{0, 1\}^{n \times n} : p(y) = \sum_{i=1}^{n} \sum_{j=1}^{n} 2^{n^2 - (i-1) - j} y_{ij}, \tag{6}
\]

and

\[
\forall y \in \{0, 1\}^{n \times n} : p(y) = \sum_{i=1}^{n} \sum_{j=1}^{n} 2^{n^2 - i - (j-1)} y_{ij}. \tag{7}
\]

In the next section we derive conditions for exactly solving a binary classification problem by a multi-layered perceptron.

## 2 Binary Classification And Exact Multi-Layered Perceptrons

Consider a binary classification problem \((\Omega, \Gamma)\) with disjoint subsets. We assume that \(\Omega = \mathbb{R}^N\) for some fixed integer \(N\), which implies that \(\Gamma = \{V, V^*\}\) with \(V \subseteq \mathbb{R}^N\) and \(V^* = \mathbb{R}^N \setminus V\) (extending the results to the case where \(\Omega \subset \mathbb{R}^N\) is straightforward). We pose ourselves the following:

**Question 1** Can the classification problem \((\mathbb{R}^N, \{V, V^*\})\) be solved by an exact multi-layered perceptron \((m-LP)\), with \(m\) layers, \(N\) real valued inputs and 1 binary valued output. “Solving” means in this context that the \(m\)-LP, represented by the function \(f: \mathbb{R}^N \rightarrow \{0, 1\}\), has the property that \(f(x) = 1\) for all \(x \in V\) and \(f(x) = 0\) for all \(x \in V^*\).
In this paper we consider the standard architecture of an $m$-LP consisting of one output layer and $m - 1$ hidden layers (see also [9, 11]). Every layer can have a different number of nodes. There are connections only between nodes in subsequent layers and the subsequent layers are completely connected. The output of a node is the result of a computation determined by a summation of a bias and the weighted inputs of that node, which is then passed through a non-linear response function $\theta$. The output of a node is thus given by $\theta(\sum_i a_i x_i + b)$, where $a_i$, $x_i$ and $b$ are the weights, inputs and bias of that node, respectively.

In this paper we examine the exact capabilities of an $m$-LP, and therefore only consider the hard limiting response function $\theta = \Theta$ defined as follows:

$$\Theta(\lambda) = \begin{cases} 0 & \text{if } \lambda < 0, \\ 1 & \text{if } \lambda \geq 0. \end{cases}$$

If an $m$-LP represented by the function $f : \mathbb{R}^N \rightarrow \{0, 1\}$ exactly solves a given classification problem $(\mathbb{R}^N, \{V, V^*\})$, it satisfies $f(x) = 1$ for all $x \in V$ and $f(x) = 0$ for all $x \in V^*$. Hence, the classification problem can be viewed as a functional specification of this $m$-LP. The network configuration and the values of the weights and biases that form this $m$-LP can be viewed as a structural specification of that $m$-LP.

**Example 5** In Figure 2 an example of a specification of a 2-LP is given. At the left the functional specification is presented both geometrically and algebraically. At the right the structural specification is given in two different ways, first topologically and second constructionally (see Section 2.1).

Before we can answer Question 1, we give a precise definition of the considered $m$-LPs.

### 2.1 The Structural Specification Of $m$-LPs

In order to investigate which binary classification problems can be exactly solved by an $m$-LP we have to characterize the possible $m$-LP functions.

Let $R_{m,N,K}$ denote the set of all vector functions from $\mathbb{R}^N$ to $\{0, 1\}^K$ that can be formed by constructing an $m$-LP with $N$ inputs and $K$ outputs, i.e. for which such a function is the structural specification of an $m$-LP.

It is convenient to define $R_{m,N,K}$ iteratively, we start with $R_{1,N,K}$:

$$R_{1,N,K} = \{ f : \mathbb{R}^N \rightarrow \{0, 1\}^K \mid f_i = \Theta \circ \tilde{f}_i, \tilde{f}_i \in A_N, i = 1, \ldots, K \}, \quad (8)$$

where $A_N$ denotes the set of all affine functions from $\mathbb{R}^N$ to $\mathbb{R}$:

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

Next, since an $(m + 1)$-LP can be constructed by putting a 1-LP on top of an $m$-LP, we define $R_{m+1,N,K}$ in terms of $R_{1,L,K}$ and $R_{m,N,L}$, where $L$ denotes the number of hidden nodes in the $m^{th}$ hidden layer:

$$R_{m+1,N,K} = \{ f : \mathbb{R}^N \rightarrow \{0, 1\}^K \mid f = g \circ h, g \in R_{1,L,K}, h \in R_{m,N,L}, L \in \mathbb{N} \}. \quad (9)$$
Figure 2: Four views to the specification of a two-layered perceptron.

Alternatively, one could obtain an \((m+1)\)-LP by putting an \(m\)-LP on top of a 1-LP. Hence we also have:

\[
R_{m+1,N,K} = \{ f : \mathbb{R}^N \to \{0,1\}^K \mid f = g \circ h, \ g \in R_{m,L,K}, \ h \in R_{1,N,L}, \ L \in \mathbb{N} \}. \tag{10}
\]

Moreover, the equivalence of (9) and (10) follows straightforwardly from the associativity of the composition operator \(\circ\).
2.2 Conditions For Exact m-LPs

We return to the problem of exactly classifying a subset \( V \subset \mathbb{R}^N \), for some fixed \( N \in \mathbb{N} \). Abbreviate \( R_{m,N,1} \) as \( R_m \), then \( V \) can be exactly classified by an \( m \)-LP if an \( f \in R_m \) exists such that \( f(x) = 1 \) if and only if \( x \in V \). Define for each function \( f : \mathbb{R}^N \rightarrow \{0,1\} \) the set \( \mathcal{J}(f) \subseteq \mathbb{R}^N \) by \( \mathcal{J}(f) = \{ x \in \mathbb{R}^N \mid f(x) = 1 \} \), then it follows that \( V \) can be classified by an \( m \)-LP if \( V = \mathcal{J}(f) \) for some \( f \in R_m \).

The collection of subsets of \( \mathbb{R}^N \) that can be exactly classified by an \( m \)-LP is denoted as \( C_m \). It is clear from the above that \( C_m = \{ \mathcal{J}(f) \mid f \in R_m \} \). In the remainder of this section and the following section we derive conditions for the exact classifiability of a subset \( V \subseteq \mathbb{R}^N \), which appear as characterizations of \( C_m \) \((m \in \mathbb{N})\). In this section we concentrate on a general condition, which holds for all values of \( m \).

Example 6 (One-Layered Perceptron)

\( V \subseteq \mathbb{R}^N \) is a closed linear half-space if it is of the form \( \{ x \in \mathbb{R}^N \mid a \cdot x + b \geq 0 \} \) for some \( a \in \mathbb{R}^N \) and \( b \in \mathbb{R} \). One can easily show that \( C_1 = \{ V \subseteq \mathbb{R}^N \mid V = \text{closed linear half-space} \} \).

From the example above we have that \( V \in C_1 \) implies that \( V^* \notin C_1 \). The following lemma shows that this property is exclusive for \( C_1 \).

**Lemma 1** If \( m \geq 2 \) then \( V \in C_m \) implies that \( V^* \notin C_1 \).

**Proof**

Suppose that \( V \in C_m \), \( m \geq 2 \), then \( V = \mathcal{J}(f) \) for some \( f \in R_m = R_{m,N,1} \). Then by (10), \( f = g \circ h \) with \( g \in R_{1,K,1} \) and \( h \in R_{m-1,N,K} \) for some \( K \in \mathbb{N} \). Let \( g = \Theta \circ \tilde{g} \), then since \( \{ h(x) \mid x \in \mathbb{R}^N \} \) is finite, there exists an \( \varepsilon > 0 \) such that for \( \tilde{g} = \tilde{g} + \varepsilon \) we have \( \tilde{g}(h(x)) \neq 0 \) and \( \tilde{g}(h(x)) > 0 \) if \( \tilde{g}(h(x)) \geq 0 \), for all \( x \in \mathbb{R}^N \). It follows that:

\[
1 - f(x) = 1 - \Theta[\tilde{g}(h(x))] = 1 - \Theta[\tilde{g}(h(x))] = \Theta[-\tilde{g}(h(x))],
\]

for all \( x \in \mathbb{R}^N \). Since \(-\tilde{g} \in A_K\) this implies that \( V^* = \mathcal{J}(1 - f) \subseteq C_m \).

Next we show that multi-layered perceptrons with more than one layer can exactly classify more complex subsets as the 1-LP. Clearly, this result has already been known for a long time [10, 9, 11].

**Lemma 2** Let \( \{ V_i \mid i = 1, \ldots, l \} \) be a collection of subsets with \( V_i \in C_m \) or \( V_i^* \in C_m \) for all \( i \), then \( \bigcap_{i=1}^{l} V_i \in C_{m+1} \).

**Proof**

Without loss of generality we assume that \( V_i \in C_m \) for \( i = 1, 2, \ldots, k \) and \( V_i^* \in C_m \) for \( i = k + 1, \ldots, l \) for some \( 0 \leq k \leq l \). Define \( f \in R_{m+1,N,1} \) by \( f = g \circ h \), where \( g \in R_{1,l,1} \) is given by \( g(x) = \Theta(\sum_{i=1}^{k} x_i - \sum_{i=k+1}^{l} x_i - k) \) and \( h \in R_{m,N,l} \) is given by \( \mathcal{J}(h_i) = V_i^* \). Then we have:

\[
x \in \mathcal{J}(f) \iff g(h(x)) = 1
\]
which completes the proof.

\[ \Lhi(x) - k? \leq 0, \quad \forall i = 1, \ldots, l : x \in \V_i, \]

which completes the proof.

To get a better understanding of which sets \( V \subseteq \mathbb{R}^N \) are exactly classifiable, we define three collections of subsets of \( \mathbb{R}^N \):

\[
\begin{align*}
\hat{C}_1 &= \{ V \subseteq \mathbb{R}^N \mid V \in C_1 \lor V^* \in C_1 \}, \\
\tilde{P} &= \{ V \subseteq \mathbb{R}^N \mid V = \bigcap_{i=1}^{k} V_i, V_i \in \hat{C}_1, k \in \mathbb{N} \}, \\
U &= \{ V \subseteq \mathbb{R}^N \mid V = \bigcup_{i=1}^{l} V_i, V_i \in \tilde{P}, l \in \mathbb{N} \}.
\end{align*}
\]

\( \hat{C}_1 \) is the collection of open and closed linear half-spaces. The collection \( \tilde{P} \) can be viewed as the collection of all pseudo-polyhedra. A genuine polyhedron (i.e., an intersection of a finite number of closed linear half-spaces) has all faces belonging to the set, whereas a pseudo-polyhedron can have faces either belonging to the set or to the complement of the set. Finally, the collection \( U \) is the collection of all subsets of \( \mathbb{R}^N \) that can be represented as a union of a finite number of pseudo-polyhedra. \( U \) is the collection of subsets that have piece-wise linear bounds. Note that \( \hat{C}_1 \subseteq \tilde{P} \subseteq U \).

It follows directly from Lemma 2 that \( \tilde{P} \subseteq C_2 \). Combining Lemma 1 and Lemma 2 once more yields \( U \subseteq C_3 \). Since \( C_m \subseteq C_{m+1} \) (special case of Lemma 2), we have proven the following theorem.

**Theorem 2** Let \( V \subseteq \mathbb{R}^N \), then \( V \) can be exactly classified by an \( m \)-LP if \( V \in U \) and \( m \geq 3 \).

The following theorem shows that \( V \in U \) is a necessary condition for a subset \( V \subseteq \mathbb{R}^N \) to belong to any \( C_m \) for arbitrary \( m \in \mathbb{N} \).

**Theorem 3** Let \( V \subseteq \mathbb{R}^N \), then \( V \) can be exactly classified by an \( m \)-LP only if \( V \in U \).

**Proof**

We show that \( C_m \subseteq U \) for all \( m \geq 1 \). For \( m = 1 \) this result follows from \( C_1 \subseteq \hat{C}_1 \subseteq U \).

Suppose \( V \in C_m, m \geq 2 \), then \( V = \mathcal{F}(f) \) for some \( f \in R_m = R_{m,N,1} \). Hence by (10), \( f = g \circ h \) with \( g \in R_{m-1,K,1} \) and \( h \in R_{1,N,K} \) for some \( K \in \mathbb{N} \). For all \( x \in \mathbb{R}^N \) we define the set \( T_x \subseteq \mathbb{R}^N \) by:

\[ T_x = \{ y \in \mathbb{R}^N \mid h(y) = h(x) \}. \]

The proof is completed by showing the following results.

(i) There exist \( l \in \mathbb{N} \) and \( x_1, x_2, \ldots, x_l \in \mathbb{R}^N \) such that:

\[ V = \bigcup_{j=1}^{l} T_{x_j}, \]
(ii) For every \( x \in \mathbb{R}^N \) we have \( T_x \in \tilde{P} \).

The proof of (i) starts by considering the set \( B = \{ z \in h(\mathbb{R}^N) \mid g(z) = 1 \} \). Since \( h(\mathbb{R}^N) \subseteq \{0,1\}^K \) is finite, \( B \) is finite and hence, \( B = \{ h(x_1), h(x_2), \ldots, h(x_l) \} \) for some \( l \in \mathbb{N} \) and \( x_1, x_2, \ldots, x_l \in \mathbb{R}^N \). It remains to show that these \( x_j \) satisfy (15), which follows straightforwardly:

\[
\begin{align*}
x \in V & \iff f(x) = g(h(x)) = 1 \\
& \iff h(x) \in B \\
& \iff \exists 1 \leq j \leq l : h(x) = h(x_j) \\
& \iff \exists 1 \leq j \leq l : x \in T_{x_j}.
\end{align*}
\]

To prove the second part we first note that \( \mathcal{J}(h_i), \mathcal{J}^*(h_i) \in \tilde{C}_1 \). Then the result follows by showing that:

\[
\begin{align*}
y \in T_x & \iff h(y) = h(x) \\
& \iff \forall 1 \leq i \leq K : h_i(x) = h_i(y) \\
& \iff \forall i, h_i(x) = 1 : y \in \mathcal{J}(h_i) \land \forall i, h_i(x) = 0 : y \in \mathcal{J}^*(h_i) \\
& \iff y \in \bigcap_{h_i(x) = 1} \mathcal{J}(h_i) \cap \bigcap_{h_i(x) = 0} \mathcal{J}^*(h_i).
\end{align*}
\]

This completes the proof of (i) and (ii).

**Note:** Theorem 1 can also be proved by viewing \( g \) as a binary function. By interpreting binary variables as boolean variables, we can use the well-known result that every boolean function can be put in conjunctive normal form (CNF) to obtain a similar expression for \( V \) as given above.

Combining Theorem 2 and Theorem 3 we have that \( V \subseteq U \) is necessary and sufficient for the corresponding binary classification problem to be solvable by an exact 3-LP, which can be formulated as the following corollary.

**Corollary 1** \( C_m = U \) for all \( m \geq 3 \).

An important conclusion is that in theory there is never a need for more than three layers, since the capabilities of the \( m \)-LP are equal for all \( m \geq 3 \). However, it is possible that adding a fourth layer reduces the total number of nodes needed for a specific problem (see Section 3). In [12] it is demonstrated that the condition given of Theorem 3 is not sufficient for solving a binary classification problem by an exact 2-LP, i.e. there exist binary classification problems that can be solved by an exact 3-LP but cannot be solved by an exact 2-LP, which implies that \( C_2 \) is a proper subset of \( C_3 \).
3 Combinatorial Optimization And Exact Multi-Layered Perceptrons

Consider a combinatorial optimization problem \((I, S, F, (c, p))\), with \(I = \mathbb{R}^N\) and \(S = \{0, 1\}^K\). We show —using the results of the previous sections— that this problem can be solved by an exact 3-LP with \(N\) inputs and \(K\) outputs, if \(F\) and \(c\) satisfy some mild conditions. To this end the combinatorial optimization problem is firstly transformed into the classification problem \((\Omega, \Gamma)\) with \(\Omega = I, \Gamma = \{\Omega_y | y \in F\}\) and \(\Omega_y\) given by Equation 4 of Section 1. This classification problem can be solved exactly by simultaneously solving the binary classification problems \((n, V_{\Omega}^{(i)}, V_{\Gamma}^{(i)})\), \(i = 1, \ldots, K\), with \(V_{\Omega}^{(i)}\) defined by Equation 2 of Section 1. The final step is then to show that \(V_{\Omega}^{(i)} \in C_3\), which proves that each of the binary classification problems can be solved exactly by a 3-LP. This implies that \((\Omega, \Gamma)\) can be solved by an exact 3-LP with \(N\) inputs and \(K\) outputs. If we have \(F(x) = F\) for all \(x \in I\), we complete this final step by showing that \(V_{\Omega}^{(i)} \in U \subseteq C_3\), otherwise we show that \(V_{\Omega}^{(i)} = \bigcap_j V_j\) for some \(V_j \in U \subseteq C_3\), and use Lemma 2 of Section 2 to conclude \(V_{\Omega}^{(i)} \in C_4 = C_3\).

If we want to complete the final step of the above procedure, we must show that \(V_{\Omega}^{(i)} \in C_m\) for some \(m \in \mathbb{N}\). It is therefore necessary that \(V_{\Omega}^{(i)} \in U\), see Theorem 3 of Section 2. Hence, the subsets \(V_{\Omega}^{(i)}\) are completely determined by the bounds of the subsets \(\Omega_y\), it follows that the subsets \(\Omega_y\) must have piece-wise linear bounds too. This directly imposes a condition on the feasible sets \(F(x)\) and the cost function \(c(y; x)\). It is sufficient if both \(F(x)\) and \(c(y; x)\) are linear functions of \(x\), where \(F(x)\) is said to be a linear function of \(x\) if all the constraints that define \(F(x)\) are linear in \(x\). Notice that in all four examples of combinatorial optimization problems put in a 0-1-formulation given in Section 1, both \(F(x)\) and \(c(y; x)\) are linear functions of \(x\).

Although sufficient, linearity of \(F(x)\) and \(c(y; x)\) is not necessary for the sets \(\Omega_y\) to have piece-wise linear bounds. A more tight condition can be expressed in terms of the sets \(\Phi_y\) and \(\Psi_{y,z}\), which for all \(y, z \in S\) are defined by:

\[
\Phi_y = \{x \in I | y \in F(x)\},
\]

and

\[
\Psi_{y,z} = \{x \in I | c(y; x) \leq c(z; x)\},
\]

respectively. The conditions we pose upon the feasible sets \(F(x)\) and the cost function \(c(y; x)\) of the combinatorial optimization problem \((I, F, S, (c, p))\) are that \(\Psi_{y,z} \in C_1\) and \(\Phi_y \in P\) for all \(y, z \in S\). We show that an exact 3-LP for this combinatorial optimization problem exists if these conditions are satisfied. For notational convenience we sometimes use the order \(\leq\) that \(p\) induces on \(F\), instead of \(p\) itself. This order is defined by:

\[
\forall y, z \in F : y \leq z \Leftrightarrow p(y) \leq p(z).
\]
We start by considering the special case where \( F(x) = F \) for all \( x \in I \); the general case is dealt with below. If \( F(x) = F \) then \( \Phi_y = I = \mathbb{R}^N \) if \( y \in F \) and \( \Phi_y = \emptyset \) if \( y \notin F \), which implies \( \Phi_y \in P \) for all \( y \in S \). If furthermore \( \Psi_{y,z} \in C_1 \) for all \( y, z \), the combinatorial optimization problem \((I, S, F, (c, p))\) can be solved by a 3-LP, as is shown in the following theorem.

**Theorem 4** Let \((I, S, F, (c, p))\) be a combinatorial optimization problem with \( I = \mathbb{R}^N \) and \( S = \{0, 1\}^K \). If \( F(x) = F \) for all \( x \in I \) and \( \Psi_{y,z} \in C_1 \) for all \( y, z \in F \), then the problem can be solved by an exact 3-LP with \( N \) inputs and \( K \) outputs.

**Proof**
First note that \( F(x) = F \) implies that the subsets \( \Omega_y \ (y \in F) \) given by equation 4 of Section 2 can be written as:

\[
\Omega_y = \{ z \in I \mid \forall z \in F, p(z) < p(y) : c(y; x) \leq c(z; x) \land \forall z \in F, p(z) > p(y) : c(y; x) < c(z; x) \}.
\]  
(18)

Using the subsets \( \Psi_{y,z} \) given by (17), the following equivalent formulation can be obtained straightforwardly:

\[
\Omega_y = \bigcap_{z \in F, z < y} \Psi_{y,z} \cap \bigcap_{z \in F, z > y} \Psi_{z,y}.
\]  
(19)

Hence, using the assumption that \( \Psi_{y,z} \in C_1 \) and the definition of \( P \), it directly follows that \( \Omega_y \in P \) for all \( y \). Since \( V_q^{(i)} = \bigcup_{y \in F, y = q} \Omega_y \), this implies that \( V_q^{(i)} \in U = C_3 \). Combining this with the result of Theorem 1 the proof is completed.

We can give an explicit description of the 3-LP mentioned in Theorem 4. Construct a 3-LP with \( N \) inputs, \( \frac{1}{2}|F|(|F| - 1) \) nodes in the first hidden layer labeled \((y, z)\) for all \( y, z \in F \) with \( y > z \), \( |F| \) nodes in the second hidden layer labeled \((y)\) for all \( y \in F \) and \( K \) nodes in the output layer labeled \((i)\) for \( i = 1, \ldots, K \). For every \( y, z \in F \) with \( y > z \), the node \((y, z)\) in the first layer classifies \( \Psi_{y,z} \). For every \( y \in F \), the node \((y)\) in the second layer classifies \( \Omega_y \) and finally, for each \( i = 1, \ldots, K \), the node \((i)\) in the output layer classifies \( V_q^{(i)} \). The corresponding structural specification of this exact 3-LP is given by the following definition of \( f \in R_3 \). \( f : \mathbb{R}^N \to \{0, 1\}^K \) is the function \( f = g^{(3)} \circ g^{(2)} \circ g^{(1)} \), with \( g^{(l)} = \Theta \circ \tilde{g}^{(l)} \) and \( \tilde{g}^{(l)} \) for \( l = 1, 2, 3 \) defined by:

\[
\begin{align*}
\forall y, y' \in F, y > y' : \quad \tilde{g}^{(1)}_{y, y'}(x) &= c(y'; x) - c(y; x), \\
\forall y \in F : \quad \tilde{g}^{(2)}_y(w) &= \sum_{z, z' \in F, z > z'} (1_{z = y} - 1_{z' = y}) w_{z, z'} - \sum_{z \in F} 1, \\
\forall i = 1, \ldots, K : \quad \tilde{g}^{(3)}_i(v) &= \sum_{z \in F} z_i v_z - 1,
\end{align*}
\]  
(20)

where \( 1_{\{\cdot\}} \) denotes the true-false indicator \( (1_{\text{true}} = 1, 1_{\text{false}} = 0) \). From (20) the values of the weights and the biases can be easily read off. Usually most non-zero weights have a
value of \( \pm 1 \). The weights in the first hidden layer can be found from the definition of the cost function \( c \) (see the examples given in Section 1).

In the next example we present an exact 3-LP that solves the problem of sorting 3 numbers in full detail. Note that the set of feasible solutions of the sorting problem is equal to the set of feasible solutions of the traveling salesman problem given in Example 2. This implies that the network configuration of the exact 3-LP given for the sorting problem below is the same as the network configuration of the exact 3-LP that solves the traveling salesman problem, except for the first hidden layer which requires some minor modifications.

**Example 7 (An Exact 3-LP That Solves The Sorting Of 3 Numbers)**

Consider Example 3 with \( n = 3 \). There are 6 feasible solutions, which we present in the order induced by the preference function given by Equation 7 of Section 1.

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix}.
\]

We number these solutions 1, 2, 3, 4, 5 and 6, respectively. The cost attached to each of these solutions is:

\[
c(1; x) = -x_1 - 2x_2 - 3x_3, \quad c(2; x) = -x_1 - 3x_2 - 2x_3,
\]
\[
c(3; x) = -2x_1 - x_2 - 3x_3, \quad c(4; x) = -2x_1 - 3x_2 - 1x_3,
\]
\[
c(5; x) = -3x_1 - x_2 - 2x_3, \quad c(6; x) = -3x_1 - 2x_2 - 1x_3.
\]

In Figure 3 we present an exact 3-LP for this problem, which can be found using expression (20). At the left-hand side the network configuration is shown, with some of the connections (not all). At the right-hand side the complete weight matrices are given. For \( n \) numbers this solution needs a total of \( \frac{1}{2}(n!)(n! + 1) + n^2 \) nodes, whereas in [13] we show that also an exact 3-LP exists that uses \( \frac{1}{2}n(5n - 1) \) nodes.

Next we consider the case that \( F(x) \neq F \) for some \( x \in I \). In this case the combinatorial optimization problem \((I, S, F, (c, p))\) can still be solved by a 3-LP, as long as the conditions \( \Psi_{y,x} \in C_1 \) and \( \Phi_y \in P \) are satisfied. To this end we first derive an exact 4-LP. Since \( C_4 = C_3 \) it then follows that an exact 3-LP must exist. We can in fact construct an exact 3-LP, but it needs approximately \( 2|F| \) hidden nodes, which is much more that the exact 4-LP, which requires only \( O(|F|^2) \) hidden nodes. We therefore propose to use the exact 4-LP in this case.

**Theorem 5**

Let \((I, S, F, (c, p))\) be a combinatorial optimization problem with \( I = \mathbb{R}^N \) and \( S = \{0,1\}^K \). If \( \Psi_{y,x} \in C_1 \) and \( \Phi_y \in P \) for all \( y, z \in S \) then the problem can be solved by a 3-LP with \( N \) inputs and \( K \) outputs.

**Proof**

In general the subsets \( \Omega_y (y \in F) \) given by Equation 4 of Section 2 can be written as:

\[
\Omega_y = \{ x \in I \mid y \in F(x) \land \forall z \in F, z < y : z \notin F(x) \lor c(y; x) \leq c(z; x) \\
\land \forall z \in F, z > y : z \notin F(x) \lor c(y; x) < c(z; x) \}.
\]  

(21)
Figure 3: The network configuration and weight matrices of a 3-LP that exactly solves the problem of sorting 3 numbers. The matrix $A_i$, $i = 1, 2, 3$, denotes the weight-matrix of the $i$-th layer, with dimensions as shown and the numbers corresponding to the numbers of the nodes. The bias of every node in the first hidden layer and the output layer equals 0 and -1, respectively. The bias of node $i$ in the second hidden layer equals $1 - i$, $i = 1, \ldots, 6$.

By using the subsets $\Psi_x$ and $\Phi_x$ introduced above one can easily verify that:

$$\Phi_x = \Phi_x^{\phi} \cup \bigcup_{x \in F} (\Phi_x^{\psi} \cup \Phi_x^{\psi^*})$$

or equivalently,

$$\Omega_y = \Phi_x \cup \bigcup_{x \in F} (\Phi_x^{\psi} \cup \Phi_x^{\psi^*}).$$

One easily shows that $\Phi_x \in P$ implies that $\Phi_x^{\phi} \in \Omega_y$. Furthermore, using that $\Psi_y \in C_1$, we can also show that:

$$\Phi_x = \Phi_x^{\phi} \cup \bigcup_{x \in F} (\Phi_x^{\psi} \cup \Phi_x^{\psi^*})$$

or equivalently,

$$\Omega_y = \Phi_x \cup \bigcup_{x \in F} (\Phi_x^{\psi} \cup \Phi_x^{\psi^*}).$$
we find that $\Phi_z \cap \Psi_{y,z} \in P$ and $\Phi_z \cap \Psi_{z,y} \in P$, which yields $\Omega_y^* \in U = C_3$. Using that
\[
V_q^{(i)} = (V_{l-q}^{(i)})^* = \left( \bigcup_{y \in F^i, y = 1 - q} \Omega_y \right)^* = \bigcap_{y \in F^i, y = 1 - q} \Omega_y^*,
\]
and applying the result of Lemma 2 we directly obtain $V_q^{(i)} \in C_4 = C_3$. Finally we use Theorem 1 to complete the proof.

We give an explicit description of the exact 4-LP described in the proof of Theorem 5 above. Since $\Phi_y \in P$ we assume
\[
\Phi_y = \bigcap_{j=1}^k \mathcal{J}(h_{y,j}) \cap \bigcap_{j=k+1}^l \mathcal{J}^*(h_{y,j}),
\]
with $h_{y,j} = \Theta \circ \tilde{h}_{y,j} \in R_1$, for some $k, l \in \mathbb{N}, 0 \leq k \leq l$. $l$ equals the number of constraints that define $F(x)$. We construct the exact 4-LP with $N$ inputs, $\frac{1}{2}|F|(|F| - 1 + 2l)$ nodes in the first hidden layer, labeled $(y, z)$ for all $y, z \in F$ with $y \succ z$ and $(y, j)$ for all $y \in F$ and $j = 1, \ldots, l$, $|F|^2$ nodes in the second hidden layer, labeled $(y, z)$ for all $y, z \in F$, $|F|$ nodes in the third hidden layer, labeled $(y)$ for all $y \in F$ and $K$ nodes in the output layer, labeled $(i)$ for $i = 1, \ldots, K$.

For every $y, z \in F$, the node $(y, z)$ in the first hidden layer classifies $\Psi_{y,z}$ and for every $y \in F$ and $j = 1, \ldots, l$, the node $(y, j)$ classifies $\mathcal{J}(h_{y,j})$. In the second layer a node $(y, z)$ classifies $\Phi^*_{y} \cup \Psi_{y,z} = \bigcup_{j=1}^k \mathcal{J}^*(h_{x,j}) \cup \bigcup_{j=k+1}^l \mathcal{J}(h_{x,j}) \cup \Psi_{y,z}$ if $y \succ z$, classifies $\Phi^*_{y} \cup \Psi_{y,z}$ if $y \prec z$ and classifies $\Phi^*_{y}$ if $y = z$. For every $y \in F$, the node $(y)$ in the third layer classifies $\Omega_y$ and finally, for each $i = 1, \ldots, K$, the node $(i)$ in the output layer classifies $V_i^{(i)}$. The corresponding structural specification of this exact 4-LP is given by the following definition of $f \in R_4, f : \mathbb{R}^N \rightarrow \{0, 1\}^K$ is the function $f = g^{(4)} \circ g^{(3)} \circ g^{(2)} \circ g^{(1)}$, with $g^{(i)} = \Theta \circ \tilde{g}^{(i)}$ and $\tilde{g}^{(i)}$ for $i = 1, 2, 3, 4$ defined by:

\[
\forall y, y' \in F^i, y \succ y': \tilde{g}^{(1)}_{y,y'}(x) = c(y'; x) - c(y; x),
\]
\[
\forall y \in F, i = 1, \ldots, l: \tilde{g}^{(1)}_{y,i}(x) = \tilde{h}_{y,i}(x),
\]
\[
\forall y, y' \in F: \tilde{g}^{(2)}_{y,y'}(w) = \sum_{z, z' \in F \atop z \succ z'} (1_{z = y} 1_{z' = y'} - 1_{z = y'} 1_{z' = y})w_{z, z'}
\]
\[
+ \sum_{z \in F, j = 1}^l (1_{z = y} (1_{j > k} - 1_{j \leq k})w_{z,j} + k - 1_{y \succ y'}),
\]
\[
\forall y \in F: \tilde{g}^{(3)}(v) = \sum_{z, z' \in F} (1_{z = y} 1_{z' \neq y} - 1_{z' = y})v_{z, z'} - |F| + 1,
\]
\[
\forall i = 1, \ldots, K: \tilde{g}^{(4)}(u) = \sum_{z \in F} z_i u_z - 1.
\]
Since $C_4 = C_3$, it is also possible to construct an exact 3-LP that solves the problem. We present a sketch of a possible construction and show why this solution is rejected. First note that $\Omega_y$ given by (22) is of the form $\bigcap_{i=1}^{|F|} \bigcup_{j=1}^{l+1} A_{ij}$, for some $A_{ij} \in \mathcal{C}_1$, depending on $y$ and with $l$ defined above. To be able to show that $V_q^{(i)} = \bigcup_{y \in F, \psi = q} \Omega_y = \bigcup_{j} B_j$ for some $B_j \in \mathcal{P}$, it is necessary to exchange the intersection operator and the union operator in the expression for $\Omega_y$. Using elementary set theory one can show:

$$\bigcap_{i=1}^{|F|} \bigcup_{j=1}^{l+1} A_{ij} = \bigcup_{\alpha \in \{1, \ldots, l+1\}^{|F|}} \bigcap_{i=1}^{|F|} A_{\alpha i}.$$ 

Generally the dimensions of the above formulation cannot be reduced. This would lead to an exact 3-LP having roughly $(l + 1)^{|F|}$ nodes in the second hidden layer, which compares unfavorably to the exact 4-LP that uses $O(|F|^2)$ nodes.

4 The Complexity Of Exact Multi-layered Perceptrons

In this section we study the complexity of the exact multi-layered perceptrons that we have constructed for solving combinatorial optimization problems. This is done by studying the time and space complexity of simulations of these multi-layered perceptrons on a PRAM. After a short description of the PRAM model, we show how and with what time complexity a multi-layered perceptron can be simulated on a PRAM. Next, we apply these results to the exact $m$-LPs constructed for solving combinatorial optimization problems in the previous sections. We conclude this section by discussing the obtained results within the scope of (parallel) complexity theory, which has implications for the class of problems that can be solved by an exact multi-layered perceptron.

The PRAM (Parallel Random Access Machine) is an idealized model of parallel computation which can be viewed as the parallel analogue of the sequential RAM (see [5]). A PRAM consists of an unlimited number of sequential processors, each with its own private memory, communicating with one another through a global memory. In one unit of time, each processor can read one global or local memory location, execute a single RAM instruction and write into one global or local memory location. PRAMs are classified according to their restrictions on global memory access (see [7] for an overview). For our purpose we consider the CREW PRAM, in which simultaneous reads are allowed but simultaneous writes are forbidden. However, this choice has no dramatic effect on the forthcoming complexity results. We start the discussion of simulations of multi-layered perceptrons on a PRAM by considering the simulation of one-layered perceptrons.

**Theorem 6** Let $f \in R_{1,N,1}$ represent a 1-LP with $N$ inputs and 1 output. Then $f$ can be simulated on a PRAM in $O(\log N)$ time using $O(N)$ processors.

**Proof**

Let $f(x) = \Theta(a \cdot x + B) = \Theta(\sum_{k=1}^N a_k x_k + B)$, for some $a \in \mathbb{R}^N$, $b \in \mathbb{R}$ and all $x \in \mathbb{R}^N$. 

20
Assign a PRAM-processor to every input of the perceptron, then the first step in the parallel evaluation of $f(x)$ for a given $x \in \mathbb{R}^N$ is the simultaneous calculation of the products $a_k x_k$ ($k = 1, \ldots, N$). Next, $\lceil \log N \rceil$ steps are required to calculate $\sum_{k=1}^{N} a_k x_k$ using the well-known parallel cascaded summation method. Addition of $b$ and applying the activation function $\Theta$ complete the evaluation of $f(x)$ in two more steps. □

The result of Theorem 6 is easily extended to a result about the simulation of $m$-LPs, which is given in Corollary 2 below. Note that $m \in \mathbb{N}$ is considered to be small and fixed. Furthermore, the number of inputs of a node not in the first layer equals the number of non-zero connections of that node.

**Corollary 2** Let $f \in R_{m,N,K}$ represent an $m$-LP with $N$ inputs and $K$ outputs. Let $n_i$ denote the number of nodes in layer $i$ ($i = 1, \ldots, m$) and let $\tau_{ij}$ denote the number of inputs of node $j$ ($j = 1, \ldots, n_i$) in layer $i$. Then $f$ can be simulated on a PRAM in $O(\max_{i=1,\ldots,m} n_i \log \tau_{ij})$ time using $O(\max_{i=1,\ldots,m} 2^j \tau_{ij})$ processors.

We apply the above result to the exact 3-LP for the combinatorial optimization problem $(I, S, F, (c,p))$ with $I = \mathbb{R}^N$, $S \subseteq \{0,1\}^K$ and $F(x) = F$ for all $x \in I$, given by (20). One can easily verify that $n_1 = \frac{1}{2} |F|(|F| - 1)$, $n_2 = |F|$ and $n_3 = K$. Furthermore, we have that $\tau_{1j} = N$, $\tau_{2j} = |F| - 1$ and $\tau_{3j} = \sum_{z \in F} z_j \leq |F|$. Hence, using Corollary 2, it follows that this exact 3-LP can be simulated on a PRAM in $O(\log |F|)$ time using $O(N|F|^2)$ processors. Since we have assumed $F \subseteq S \subseteq \{0,1\}^K$, the simulation requires in worst case $O(K)$ time and $O(N2^K)$ processors.

For the exact 4-LP given by (24) that solves $(I, S, F, (c,p))$ in the more general case where not always $F(x) = F$, we find that $n_1 = \frac{1}{2} |F|(|F| - 1 + 2l)$, $n_2 = |F|^2$, $n_3 = |F|$ and $n_4 = K$. The number of inputs per node are $\tau_{ij} = N$, $\tau_{2j} = l + 1$, $\tau_{3j} = |F|$ and $\tau_{4j} \leq |F|$, respectively. By assuming that $l < |F|$ and using Corollary 2, we conclude that this 4-LP can be simulated on a PRAM with same complexity bounds as the exact 3-LP above.

In order to discuss the above results, we have to make some informal remarks on parallel complexity theory, following the lines of [8]. Parallel complexity theory is related to sequential complexity theory by a hypothesis known as the parallel computation thesis: time bounded parallel machines are polynomial related to space bounded sequential machines. Formally, for any (reasonable) function $T$ of the problem size $n$, the class solvable by a machine with unbounded parallelism in time $T(n)^{O(1)}$ is equal to the class of problems solvable by a sequential machine in space $T(n)^{O(1)}$. This thesis has been proved for the PRAM - RAM combination provided that $T(n) \geq \log n$ (see [5]).

Hence, the class of problems solvable by a PRAM in polynomial time is equal to the class of problems solvable on a RAM using polynomial workspace, which has been denoted as $\mathcal{P}$-SPACE. Similarly, the class of problems solvable by a PRAM in polylogarithmical time is equal to the class of problems solvable on a RAM using polylogarithmical workspace, which is denoted as $\mathcal{P}$-LOGSPACE. It has been shown that $\mathcal{P}$-LOGSPACE $\subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{P}$-SPACE and $\mathcal{P}$-LOGSPACE $\not\subseteq$ $\mathcal{P}$-SPACE. $\mathcal{P}$ and $\mathcal{NP}$ are the well-known classes of problems solvable, respectively, verifiable, on a RAM in polynomial time. Similarly as the well-
known conjecture $\mathcal{NP} \not\subseteq \mathcal{P}$, it is conjectured that $\mathcal{PSPACE} \not\subseteq \mathcal{NP}$ and $\mathcal{P} \not\subseteq \mathcal{POLYLOGSPACE}$.

Like $\mathcal{NP}$, $\mathcal{PSPACE}$ has its complete members to which all problems in $\mathcal{PSPACE}$ can be reduced in polynomial time. More interestingly are the $\mathcal{P}$-complete problems, to which all problems in $\mathcal{P}$ can be reduced by a transformation that requires a logarithmical amount of workspace. It is unlikely that a $\mathcal{P}$-complete problem is in $\mathcal{POLYLOGSPACE}$, since this would imply $\mathcal{P} \subseteq \mathcal{POLYLOGSPACE}$. Hence, it is unlikely that a $\mathcal{P}$-complete problem can be solved on a PRAM in polylogarithmical time. Finally, $\mathcal{NC} \subseteq \mathcal{POLYLOGSPACE}$ is defined as the class of problems that can be solved on a PRAM in polylogarithmical time using a polynomial number of processors. It is obvious that $\mathcal{NC} \subseteq \mathcal{P}$.

We now evaluate the exact 3-LP and 4-LP designed in this paper for solving combinatorial optimization problems that satisfy the conditions of Theorem 4 or Theorem 5, assuming that $\text{IFI} = 2^{2^K}$. This is done by viewing the complexity bounds, $O(K)$ for time and $O(N2^{2K})$ for space, of the PRAM simulation in light of the above remarks on parallel complexity theory. The bounds are expressed in $K$, which is bounded by a polynomial in the problem size $N$ (see the examples in Section 1).

If the considered problem is $\mathcal{NP}$-complete, it is unlikely that the problem can be solved in $O(K)$ time with a number of processors bounded by a polynomial in $N$. Hence, for these kind of problems the results are acceptable. If the problem is $\mathcal{P}$-complete it is unlikely to be in $\mathcal{NC}$ and hence it is unlikely that the problem can be solved on a PRAM in an amount of time essentially less than $O(K)$. However, the number of processors is very large if one considers the fact that the problem can be solved in polynomial time on 1 processor.

If the problem is in $\mathcal{POLYLOGSPACE}$, the results are clearly unacceptable, in a sense that the implementations are not efficient, neither in time nor in space requirements. Especially for these kind of problems it is worth-while to search for exact multi-layered perceptrons that use a smaller number of nodes than the exact 3-LP/4-LP presented in this paper. For the sorting problem, which is in $\mathcal{POLYLOGSPACE}$, an exact 3-LP can be found with $O(N^2)$ nodes (see [13]). This solution can be simulated on a PRAM in $O(\log N)$ time using $O(N^3)$ processors ($O(N^2)$ by a tailored simulation), which is acceptable.

**Concluding Remarks**

In this paper we have presented results about the design and complexity of the network configuration of an exact multi-layered for solving combinatorial optimization problems. The design strategy is summarized as follows. The first step is to choose a 0-1-formulation for the combinatorial optimization problem at hand, which is then reformulated as a classification problem. In general our approach for exactly solving a given 0-1-formulated classification problem is by decomposing it into a number of binary classification problems, which are then solved. However, this decomposition is only possible if the classifying subsets are disjoint. If the classifying subsets are not disjoint, which is the case if the op-
timal solution of the corresponding combinatorial optimization problem is not unique, we need a tie-breaking mechanism. This is constructed by introducing a preference measure to the set of possible outcomes.

If the classifying subsets of a binary classification problem have piece-wise linear bounds, the problem can be solved exactly by a three-layered perceptron. Augmenting the three-layered perceptron with a fourth layer does not increase its classifications capabilities. For combinatorial optimization problems we have shown that piece-wise linearity of both the cost function and the feasibility constraints is sufficient for exactly solving the problem by a three-layered perceptron. We conjecture that these conditions are also essentially necessary. If the set of feasible solutions is not the same for all considered instances, a four-layered solution needs in general much less hidden units than a three-layered solution.

We used the class of combinatorial optimization problems to obtain mathematically founded results about the network configuration of a multi-layered perceptron designed for a specific problem. At the same time we obtained an exact three-layered perceptron solution for any given 0-1-formulated combinatorial optimization problem satisfying some mild conditions. This solution, which is essentially an enumerative one, can be viewed as a massively parallel algorithm for solving instances of the problem. Because of its enumerative nature, the space requirements of the designed three-layered perceptron are in most cases unacceptable if the number of feasible solutions of the problem is large. Clearly, this holds for combinatorial optimization problems that are \( \mathcal{NP} \)-complete, for which the number of feasible solutions is exponential in the size of the instance, since for these kind of problems the conjecture \( \mathcal{NP} \not\subseteq \mathcal{P} \) implies that a polynomial number of hidden units cannot be sufficient for exactly solving the problem. Arguments obtained from parallel complexity theory show that all also for the class of \( \mathcal{P} \)-complete problems an exponential number of nodes is to be expected. Only if a problem is in \textsc{polylogspace} one can expect to find an exact multi-layered perceptron with a polynomial number of nodes that exactly solves the problem.

References


