Solving Multiple Objective Programming Problems Using Feed-Forward Artificial Neural Networks: The Interactive FFANN Procedure

Sun, M., Stam, A. and Steuer, R.E.

IIASA Working Paper

WP-95-046

May 1995

Working Paper

Solving Multiple Objective Programming Problems Using Feed-Forward Artificial Neural Networks: The Interactive FFANN Procedure

Minghe Sun
Antonie Stam
Ralph E. Steuer

WP-95-46
May 1995

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Solving Multiple Objective Programming Problems Using Feed-Forward Artificial Neural Networks: The Interactive FFANN Procedure

Minghe Sun
Antonie Stam
Ralph E. Steuer

WP-95-46
May 1995

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Solving Multiple Objective Programming Problems Using Feed-Forward Artificial Neural Networks: The Interactive *FFANN* Procedure

Minghe Sun
Division of Management and Marketing
College of Business
University of Texas at San Antonio
San Antonio, Texas 78249 USA

Antonie Stam
International Institute for Applied Systems Analysis
A-2361 Laxenburg, Austria
and
Department of Management
Terry College of Business
University of Georgia
Athens, Georgia 30602 USA

Ralph E. Steuer
Faculty of Management Science
Brooks Hall
University of Georgia
Athens, Georgia 30602 USA

May 18, 1995

FOREWORD

The usefulness of any interactive multicriteria decision making methodology depends crucially on the accuracy with which it represents the decision maker's preference structure, and on its flexibility in its treatment of preference information elicited during the interactive process. As feed-forward artificial neural networks have been applied successfully to various complex pattern recognition problems, and a decision maker's preference structure may be viewed as a pattern, the idea of applying neural networks to multicriteria problems is intuitively appealing. The current paper explores the viability of using artificial neural networks within the framework of multicriteria optimization. In a systematic analysis, the authors show convincingly that, at least for the types of problems considered in their study, the neural network approach is more robust than the Tchebycheff Procedure, one of the leading interactive methods in the field. Hence, this working paper provides an interesting and useful contribution to both the theory and practice of interactive multicriteria optimization.

Solving Multiple Objective Programming Problems Using Feed-Forward Artificial Neural Networks: The Interactive *FFANN* Procedure

Abstract

In this paper, we propose a new interactive procedure for solving multiple objective programming problems. Based upon feed-forward artificial neural networks (*FFANN*s), the method is called the Interactive *FFANN* Procedure. In the procedure, the decision maker articulates preference information over representative samples from the nondominated set either by assigning preference "values" to the sample solutions or by making pairwise comparisons in a fashion similar to that in the Analytic Hierarchy Process. With this information, a *FFANN* is trained to represent the decision maker's preference structure. Then, using the *FFANN*, an optimization problem is solved to search for improved solutions. An example is given to illustrate the Interactive *FFANN* Procedure. Also, the procedure is compared computationally with the Tchebycheff Method (Steuer and Choo 1983). From the computational results, the Interactive *FFANN* Procedure produces good results and is robust with regard to the neural network architecture.

KEYWORDS: Multiple Objective Programming, Feed-Forward Artificial Neural Networks, Multiple Criteria Decision Making, Analytic Hierarchy Process, Interactive Procedures

1. Introduction

We propose a new procedure for solving multiple objective programming problems. Called the Interactive *FFANN* Procedure, it focuses on the elicitation, representation, and utilization of preference information obtained from a decision maker (DM) in a feed-forward artificial neural network (*FFANN*) framework. One advantage of the Interactive *FFANN* Procedure over existing procedures is that it takes the initiative in searching for improved solutions, rather than merely judging the discrete solutions generated by some sampling method. Another advantage is that the *FFANN* within the procedure makes it possible to represent various types of nonlinear preference structures.

During the last two decades, much progress has been made in the modeling of multiple objective programming problems. However, although many solution procedures have been proposed, these methods have generally not been fully satisfactory. The most effective methods have been interactive procedures, which typically include alternating phases of analysis – the solution generation phase and the solution evaluation phase. Examples of interactive multiple objective programming procedures include STEM (Benayoun, de Montgolfier, Tergny and Larichev 1971), the Geoffrion-Dyer-Feinberg Procedure (Geoffrion, Dyer and Feinberg 1972), the Visual Interactive Approach (Korhonen 1987b), the Tchebycheff Method (Steuer and Choo 1983; Steuer 1986), the Zionts-Wallenius Method (Zionts and Wallenius 1983), the Reference Point Method (Wierzbicki 1982), and others as summarized in Gardiner and Steuer (1994).

Whenever a multiple objective programming problem is solved interactively in practice, three issues must be addressed: (i) how to elicit preference information from the DM over the set of feasible solutions, (ii) how to capture and represent the DM's preference structure in a systematic manner, and (iii) how to use the DM's preference structure to guide the search for improved solutions. Many methods have been developed for eliciting preference information from the DM, but finding an effective device to capture preference information and use it effectively in the search for improved solutions has been problematic.

This research addresses these three issues as follows. The DM has the choice of articulating his or her preference information either by assigning "values" to trial solutions or by making comparisons between pairs of trial solutions. The preference information elicited is then used to train a *FFANN* so as to "store" the preference information. The trained *FFANN* then serves as an approximate representation of the DM's preference structure and is combined with nonlinear programming techniques to search for improved solutions.

The contribution of this research is twofold. From the perspective of those working in management science, this research may be viewed as a new paradigm for solving multiple objective optimization problems using artificial intelligence methods. From the perspective of those working in artificial intelligence, this research can be seen as a new application of artificial neural networks to problems in constrained optimization.

The remainder of this paper is organized as follows. We briefly review the topology and dynamics of a *FFANN* in Section 2. In Section 3, we introduce notation and discuss issues related to preference information elicitation and representation. The Interactive *FFANN* Procedure is detailed in Section 4, followed by an illustrative example in Section 5. Computational results are reported in Section 6, and concluding remarks are given in Section 7. The algorithm for training the *FFANN*s employed in this paper is presented in Appendix A.

2. Feed-Forward Artificial Neural Networks

An artificial neural network consists of a set of processing units, called nodes, connected by weighted arcs, where the weights represent the strength of connections. A *FFANN* is an artificial neural network where the nodes are organized into layers, and the weighted arcs only link nodes in lower layers to nodes in higher layers (Rumelhart, Hinton and Williams 1986; Wasserman 1989). Nodes in the input layer, called input nodes, accept input from the outside world and nodes in the output layer, called output nodes, generate output to the outside world. Nodes in the input layer are used to distribute inputs only and do not serve any processing or computational function. Nodes in layers between the input layer and the output layer are called hidden nodes, and these layers are called hidden layers.

Let the input layer also be known as layer 0 and let the number of layers aside from the input layer be m. Denote node k in layer i by v_k^i; the number of nodes in layer i by n_i; and the connectivity weight from v_r^j to v_k^i by w_{kr}^{ij}. If two nodes are not connected, the connectivity weight between them is 0. Associated with v_k^i is a node bias or threshold θ_k^i. Further, denote the set of connectivity weights and node biases by $W = \{w_{kr}^{ij}, \theta_k^i\}$.

Two examples of a *FFANN*, one without direct connections from the input layer to the output layer, the other fully connected, are given in Figures 1 and 2.

Figures 1 and 2 About Here

Mapping vectors from the input space $\Re^{n_0}$ to the output space $\Re^{n_m}$, a *FFANN* can be expressed as *FFANN*: $\Re^{n_0} \rightarrow \Re^{n_m}$. The mapping of an input vector to an output vector is a dynamic process, in which node inputs and outputs are updated sequentially from the input layer to the output layer. For $i > 0$, the input to v_k^i, denoted by z_k^i, is the weighted sum of the outputs of all nodes directly connected to it from all other lower layers plus θ_k^i, *i.e.*,

$$z_k^i = \sum_{j=0}^{i-1} \sum_{r=1}^{n_j} w_{kr}^{ij} w_r^j + \theta_k^i, \tag{2.1}$$

where w_r^j is the output of v_r^j.

Each node, except for the ones in the input layer, has an activation function which computes the node's output based upon its input. The most frequently used activation function, which is also used in this paper, is the logistic function, defined as

$$u_k^i = \left(1 + e^{-\frac{z_k^i}{T}}\right)^{-1}, \tag{2.2}$$

where the "temperature" T, a user-selected scalar, determines the steepness of the activation function.

A *FFANN* is usually trained to represent an unknown mapping by employing a training set (a collection of paired input and desired output vectors observed from the unknown mapping). The purpose in training a *FFANN* is to determine the values of the elements in W so that the *FFANN* can closely represent the unknown mapping.

The training of a *FFANN* is accomplished by (1) mapping input vectors from the training set by the current version of the *FFANN* to their computed output vectors, (2) comparing the computed output vectors with their respective desired output vectors in the training set, and then (3) adjusting the values of the components of W so as to reduce any differences between the computed and desired output vectors. After a number of training iterations, the connectivity weights and node biases of the *FFANN* will converge to a set of values that minimizes the differences between the computed and desired output vectors, and the *FFANN* will organize itself internally, constructing a model to represent the unknown mapping from the input space to the output space. Thus any new input vector presented to an appropriately trained *FFANN* will yield an output vector similar to the one that would have been given by the actual mapping. The training algorithm that we used in this paper is based upon the *error back-propagation* algorithm (see Rumelhart, Hinton and Williams 1986) as described in Sun (1992) and is presented in Appendix A.

Artificial neural networks have been applied to many real world problems, especially in classification and pattern recognition (Másson and Wang 1990; Zahedi 1991). Also, artificial neural networks have been applied to problems in combinatorial optimization (Hopfield and Tank 1985; Aarts and Korst 1989) and linear programming (Tank and Hopfield 1986; Wang and Chankong 1992). Recently, Wang and Malakooti (1992) and Malakooti and Zhou (1994) have used *FFANN*s to solve discrete multiple criteria decision making problems. Burke and Ignizio (1992) provide an overview of connections between artificial neural networks and operations research.

3. Notation and Preference Information Elicitation

As for notation and terminology, a multiple objective programming problem is written as

$$\begin{aligned} &max\ \{f_1(\mathbf{x}) = z_1\} \\ &\vdots \\ &max\ \{f_k(\mathbf{x}) = z_k\} \\ &s.t.\quad \mathbf{x} \in S, \end{aligned}$$

or equivalently as

$$\begin{aligned} &max\ \{f(\mathbf{x}) = \mathbf{z}\} \\ &s.t.\quad \mathbf{x} \in S, \end{aligned}$$

where k is the number of objectives, the z_i are criterion values, and $S \subset \Re^n$ is the feasible region in decision space. Let $Z \subset \Re^k$ be the feasible region in criterion space where $\mathbf{z} \in Z$ if and only if there exists an $\mathbf{x} \in S$ such that $\mathbf{z} = (f_1(\mathbf{x}), \cdots, f_k(\mathbf{x}))$. Criterion vector $\bar{\mathbf{z}} \in Z$ is *nondominated* if and only if there does not exists another $\mathbf{z} \in Z$ such that $z_i \geq \bar{z}_i$ for all i and $z_i > \bar{z}_i$ for at least one i. The set of all nondominated criterion vectors is designated N and is called the *nondominated set.* A point $\bar{\mathbf{x}} \in S$ is *efficient* if and only if its criterion vector $\bar{\mathbf{z}} = (f_1(\bar{\mathbf{x}}), \cdots, f_k(\bar{\mathbf{x}}))$ is nondominated. The set of all efficient points is designated E and is called the *efficient set.* If a multiple objective program is all linear, it will be referred to as an MOLP (multiple objective linear program).

Let V: $\Re^k \rightarrow \Re$ be a DM's value function. A $\mathbf{z}^{\mathbf{opt}} \in Z$ that maximizes V over Z is an *optimal criterion vector* and any $\mathbf{x}^{\mathbf{opt}} \in S$ such that $(f_1(\mathbf{x}^{\mathbf{opt}}), \cdots, f_k(\mathbf{x}^{\mathbf{opt}})) = \mathbf{z}^{\mathbf{opt}}$ is an *optimal solution* of the multiple objective program. Our interest in the efficient set E and the nondominated set N stems from the fact that if V is *coordinatewise increasing* (that is, more is always better than less of each criterion), $\mathbf{x}^{\mathbf{opt}} \in E$ and $\mathbf{z}^{\mathbf{opt}} \in N$. However, in interactive multiple objective programming, because of the difficulty in precisely locating the best nondominated criterion vector, we typically conclude the search for an optimal solution with a *final solution* $\mathbf{z}^{\mathbf{fin}} \in Z$ (a solution that is either optimal, or close enough to being optimal to satisfactorily terminate the decision process).

Because of difficulties in assessing a DM's value function (see for instance, Farquhar (1984), Fishburn (1974, 1984), Keeney and Raiffa (1976), and Yu (1985)), we have been intrigued by artificial neural networks because of their ability to represent complex mappings (linear or nonlinear, convex or nonconvex, continuous or discontinuous, differentiable or nondifferentiable). For instance, Hecht-Nielsen (1987) has shown that a *FFANN* with three layers can represent any continuous mapping from $\Re^{n_0}$ to $\Re^{n_m}$, and others have shown that *FFANN*s with two hidden layers can represent any set in $\Re^n$ (Cybenko 1989; Zwietering, Aarts and Wessels 1991). With this kind of potential, the strategy of this paper becomes clear -- to develop a *FFANN* approach that can capture a DM's preference structure well enough to enable the Interactive *FFANN* Procedure to quickly locate final solutions of top quality. Hence, in this paper, we are interested in employing a *FFANN*: $\Re^{n_0} \rightarrow \Re^{n_m}$ with $n_0 = k$ and $n_m = 1$

such that *FFANN*: $\Re^k \rightarrow \Re$. In other words, in this case, the *FFANN* input is a k-dimensional (rescaled) criterion vector, and the *FFANN* output is a single (rescaled) preference value.

In the Interactive *FFANN* Procedure, two different approaches are developed for evaluating the criterion vectors generated at each iteration so that they can be used for initially training and then re-training the *FFANN* for use on each iteration. One approach is for the DM to assign an interval-scale preference "value" to each criterion vector, higher "values" representing higher degrees of satisfaction. So as to anchor the scale, the *nadir* criterion vector $\mathbf{z^{nad}}$ ($z_i^{\text{nad}} = min\{f_i(\mathbf{x}) | \mathbf{x} \in E\}$) could be given a preference value of 0, and the *ideal* criterion vector $\mathbf{z^{max}}$ ($z_i^{\text{max}} = max\{f_i(\mathbf{x}) | \mathbf{x} \in S\}$) could be given a preference value of 100. One way to obtain $\mathbf{z^{nad}}$ would be to examine the criterion vectors of all efficient extreme points. If $\mathbf{z^{nad}}$ cannot be obtained in this way, for instance if the problem is too large to enumerate all efficient extreme points, $\mathbf{z^{nad}}$ can be estimated from the minimum values in the columns of a payoff table (Isermann and Steuer 1988; Korhonen, Salo and Steuer 1994). In this way, the preference value of every nondominated criterion vector should fall within the range of 0 to 100. Actually, the scale is not important. What really matters is the order of the preference values and the differences between them.

The other approach is to make pairwise comparisons between trial solutions. In this approach, the DM is asked questions similar to those posed in the Analytic Hierarchy Process (AHP) (Saaty 1988) and in its software implementation Expert Choice (Expert Choice 1992). The advantage of eliciting preference information by pairwise comparisons is that it is easier for many DMs to provide relative than absolute preference information. The pairwise comparisons result in a reciprocal comparison matrix. Saaty (1988) has shown that the principal eigenvector components of this matrix can be viewed as the priorities of the alternative solutions. In the Interactive *FFANN* Procedure, the components of this priority vector are used as the desired outputs when training the *FFANN*.

The AHP appears to be an easy and convenient methodology for eliciting preference information from the DM. In fact, it has been used to solve different types of real world discrete multiple criteria decision making problems. Recently, there have been authors that have used the AHP to elicit preference information from the DM in multiple objective programming and have incorporated the AHP into interactive solution procedures. In their interactive method, Arbel and Oren (1987) use the AHP to assess the relative preference of the current solution and adjacent solutions. Gass (1986) used the AHP to determine goal priorities and objective function weights in a linear goal programming formulation. Korhonen (1987a) discusses the use of the AHP to find reference directions, which are then used as search directions in his visual interactive approach. Korhonen and Wallenius (1990) use the AHP to determine objective coefficients and parameter values for an MOLP problem which is subsequently solved using their visual interactive package VIG. Kok and Lootsma (1985) propose using the AHP within the framework of the Reference Point Method (Wierzbicki 1982) to find the

weighting vector for use in an achievement scalarizing program that projects reference points onto N. Barzilai and Golany (1990) derive weights for additive value functions from a reciprocal comparison matrix.

A caveat of the AHP is that it has been criticized for several theoretical shortcomings (Dyer 1990), in spite of many successful applications. One problem is the phenomenon of rank reversal when new alternatives are added to, or old alternatives are removed from, the current set of alternative solutions. Another problem is that the questions DMs are asked about the pairwise relative importance of the criteria may be viewed as ambiguous. To date, proponents and critics have not yet fully resolved these issues (Winkler 1990; Schoner, Wedley and Choo 1992). However, in the Interactive *FFANN* Procedure rank reversal problems can be handled through the interactive nature of the procedure in that at any iteration previous solutions can be re-ranked to correct for any errors that may have been made earlier in the solution process.

4. Interactive *FFANN* Procedure

In this section we specify the Interactive *FFANN* Procedure followed by comments about its different steps.

Step 0: Determine $\mathbf{z}^{\max}$ and $\mathbf{z}^{\text{nad}}$ (if nadir values are not available, use the minimum values in the columns of a payoff table). Specify the number of criterion vectors P to be presented to the DM at each iteration and the number of iterations t the procedure is to run. Select a particular *FFANN* architecture to use. Generate P dispersed criterion vectors from the nondominated set.

Repeat for $h = 1, \cdots, t$:

Step 1: After presenting the P criterion vectors along with $\mathbf{z}^{\text{nad}}$ and $\mathbf{z}^{\max}$ to the DM, identify the best criterion vector seen so far. If $h = t$, or if the DM feels that the best criterion vector obviates the need for additional iterations, designate this criterion vector as the final criterion vector $\mathbf{z}^{\text{fin}}$ and stop. Otherwise, let the DM articulate his/her preference information either by directly assigning values to the criterion vectors or by making pairwise comparisons.

Step 2: Rescale the components of each of the P criterion vectors using the transformation

$$z_i' = \frac{z_i - z_i^{\text{nad}}}{z_i^{\max} - z_i^{\text{nad}}} .$$

Step 3: If pairwise comparisons are made, compute and normalize the principal eigenvector of the reciprocal comparison matrix so that its largest component is one. If preference values are

assigned, let $V(\mathbf{z})$ be the value assigned to $\mathbf{z}$. Then, for each of the P criterion vectors, compute a normalized preference value using

$$v(\mathbf{z}) = \frac{V(\mathbf{z}) - V(\mathbf{z}^{\mathbf{nad}})}{V(\mathbf{z}^{\mathbf{max}}) - V(\mathbf{z}^{\mathbf{nad}})} . \tag{4.1}$$

Step 4: Use the rescaled criterion vectors (from Step 2) with either their normalized assigned preference values or the components of the principal eigenvector of the reciprocal comparison matrix (from Step 3) to train (if $h = 1$) or re-train (if $h > 1$) the *FFANN*.

Step 5: With the most recently trained or re-trained *FFANN* as the objective function, solve the optimization problem

$$\begin{aligned} max \quad & FFANN(\mathbf{z}) \\ s.t. \quad & \mathbf{z} = f(\mathbf{x}) \\ & \mathbf{x} \in S \end{aligned}$$

to obtain a new solution $(\mathbf{z}^{(h)}, \mathbf{x}^{(h)})$.

Step 6: If $\mathbf{z}^{(h)}$ is different from any criterion vector previously presented to the DM, generate $P-1$ new dispersed criterion vectors. If $\mathbf{z}^{(h)}$ duplicates a previously seen criterion vector, generate P new dispersed criterion vectors for presentation to the DM on the next iteration.

End Repeat.

In Step 0 there are no specific guidelines as to what *FFANN* architecture to use, in terms of the numbers of hidden layers and hidden nodes. Fortunately, as evidenced by the computational tests in Section 6, the particular *FFANN* representation of the DM's preference structure is not very sensitive to the particular *FFANN* structure employed. Finally in Step 0 (and also in Step 6), we use the augmented weighted Tchebycheff program (Steuer and Choo 1983; and Steuer 1986) to generate the dispersed criterion vectors required at each iteration.

Although a *FFANN* can be easily modified to automatically scale the input vectors and outputs by introducing nodes with linear activation functions, we always recommend in Steps 2 and 3 that all input vectors and outputs be rescaled prior to their presentation to the *FFANN*. In this way, the number of elements in W is kept at its minimum so as to avoid using unnecessarily extra time in training the *FFANN*.

Saaty (1988) has shown that if the priorities of all trial solutions are known exactly and each pairwise comparison is made based on these priorities, the components of the principal eigenvector of the reciprocal comparison matrix are identical to these priorities. Saaty (1988) suggests several alternative methods for estimating the principal eigenvector of the comparison matrix. In Step 3, we use the *power* method (Burden and Faires 1989) for this purpose. The dimensions of the reciprocal

comparison matrix are $(P+2)\times(P+2)$ because of the P trial solutions plus $\mathbf{z}^{\mathbf{max}}$ and $\mathbf{z}^{\mathbf{nad}}$.

The training algorithms developed based on error back-propagation (Rumelhart, Hinton and Williams 1986) in Sun (1992) are used to train the *FFANN*s in Step 4. The details of one of these algorithms are provided in Appendix A. As we will see below, for multiple objective programming problems with a reasonable number of objectives the structure of the *FFANN* needs not to be very complicated and the training time is typically only a few seconds. After the first iteration, the procedure offers the option to continue the training in subsequent iterations with either a warm start, using the connectivity weights of the previous iteration as initial weights, or a cold start, using random initial weights. In the computational experiments, each iteration is started with random initial weights.

The objective function of the optimization problem in Step 5, in this case the trained *FFANN*, may be complicated, requiring nonlinear programming solution techniques. In the implementation, the GRG2 package (Lasdon and Waren 1989) is used for this purpose. The gradient of the trained *FFANN* with respect to the k criterion values at a specific solution is determined numerically. The following *three point formula* (Burden and Faires 1989) is used to estimate the partial derivative of the trained *FFANN*:

$$\frac{\partial FFANN(\mathbf{z})}{\partial z_i'} \approx \frac{FFANN(z_1', ..., z_i'+\epsilon_i, ..., z_k') - FFANN(z_1', ..., z_i'-\epsilon_i, ..., z_k')}{2\epsilon_i},$$

where ϵ_i is a small positive scalar. In the computational tests, we obtained similar results for various ϵ_i-values in the range from 0.001 to 0.01.

5. An Example

To illustrate how the Interactive *FFANN* Procedure works step-by-step, consider the following MOLP problem:

$$\begin{array}{lrrrrrrrl}
max & & 2x_2 & +5x_3 & +5x_4 & -2x_5 & +5x_6 & = z_1 \\
max & -x_1 & -2x_2 & & & +4x_5 & -x_6 & = z_2 \\
max & 5x_1 & +3x_2 & -2x_3 & & -x_5 & -x_6 & = z_3 \\
 & & & & & & & \\
s.t. & & & & 7x_4 & +2x_5 & +6x_6 & \leq 28 \\
 & 3x_1 & & & & & +4x_6 & \leq 23 \\
 & 4x_1 & & +4x_3 & +x_4 & & & \leq 23 \\
 & & x_2 & +6x_3 & +7x_4 & & +4x_6 & \leq 23 \\
 & 2x_1 & +5x_2 & +5x_3 & +5x_4 & +8x_5 & & \leq 29 \\
 & & & & & & x_j & \geq 0, \quad 1 \leq j \leq 6.
\end{array}$$

Let us assume a hypothetical DM has the following value function

$$V_4(\mathbf{z}) = 50 - \left(\sum_{i=1}^{3} [\lambda_i (z_i^{\max} - z_i)]^4 \right)^{\frac{1}{4}},$$

with $\boldsymbol{\lambda} = (0.319, 0.416, 0.265)$. Using GRG2 (Lasdon and Waren 1989), the optimal solution is found to be $\mathbf{z}^{\mathbf{opt}} = (16.517, -0.886, 18.970)$ with a hypothetical value function value $V_4(\mathbf{z}^{\mathbf{opt}}) = 42.42288$.

The vector-maximum code ADBASE (Steuer 1992) was then used to compute all efficient extreme points, from which the ideal criterion vector was found to be $\mathbf{z}^{\mathbf{max}} = (33.100, 14.500, 39.250)$ with $V_4(\mathbf{z}^{\mathbf{max}}) = 50.00000$ and the nadir criterion vector was found to be $\mathbf{z}^{\mathbf{nad}} = (-7.250, -16{,}412, -9.207)$ with $V_4(\mathbf{z}^{\mathbf{nad}}) = 33.07733$. Furthermore, the worst criterion vector in the nondominated set was found to be $\mathbf{z}^{\mathbf{worst}} = (-7.250, 14.500, -3.625)$ with $V_4(\mathbf{z}^{\mathbf{worst}}) = 35.50926$. The worst nondominated criterion vector, of course, is used as a benchmark to measure the quality of solutions only and is not used in the Interactive *FFANN* Procedure. Now let the number of solutions that are to be presented to the DM at each iteration be $P = 7$ and the number of iterations the procedure is to run be $t = 5$.

The augmented weighted Tchebycheff program (Steuer and Choo 1983; Steuer 1986) was then used to generate the seven dispersed nondominated solutions in the first iteration, as shown in Table 1. Together with $\mathbf{z}^{\mathbf{max}}$ and $\mathbf{z}^{\mathbf{nad}}$, the seven nondominated solutions are presented to the DM, who then evaluates them, either by directly assigning preference values or by making pairwise comparisons. In the example, the $V_4(\mathbf{z})$ values in Table 1 represent the preference information elicited from the hypothetical DM.

Tables 1, 2 and Figure 3 About Here

We train the *FFANN* shown in Figure 3, with one hidden layer comprised of two hidden nodes, using the rescaled criterion vectors in Table 2 as the inputs and the normalized preference values as the desired outputs. Thus, each line of Table 2 corresponds to one training pattern. The connectivity weights and node biases of the trained *FFANN* are shown in Figure 3. The connectivity weights between the nodes are given by the values on the arcs, while the node biases are indicated inside each node in the hidden and output layers. The temperature used in training this *FFANN* was $T = 10$.

To demonstrate the mapping of a *FFANN*, let $\mathbf{z}' = (0.78, 0.16, 0.76)$ be a given input vector of rescaled criterion values. It follows from (2.1) that $z_1^1 = 13.74(0.78) - 47.17(0.16) + 22.30(0.76) + 21.26 = 41.38$, and $z_2^1 = 38.58(0.78) - 53.90(0.16) + 53.97(0.76) - 61.23 = 1.26$. Hence, from (2.2) it follows that $u_1^1 = [1 + e^{-\frac{41.38}{10}}]^{-1} = 0.98$ and $u_2^1 = [1 + e^{-\frac{1.26}{10}}]^{-1} = 0.53$, respectively. Similarly, $z_1^2 = 40.55(0.78) + 9.17(0.16) + 46.03(0.76) - 30.55(0.98) - 28.32(0.53) - 29.09 = -5.96$, and $u_1^2 = [1 + e^{-\frac{-5.96}{10}}]^{-1} = 0.36$.

Solving the optimization problem in Step 5 of the procedure yields $\mathbf{z}^{(1)} = (19.16292, -4.44382, 24.18538)$ with $V_4(\mathbf{z}^{(1)}) = 41.80951$. At this point, one iteration has been completed. Solution $\mathbf{z}^{(1)}$ is

different from any of the solutions previously presented to the DM.

Along with $\mathbf{z}^{(1)}$, in the second iteration six new nondominated criterion vectors are presented to the DM for evaluation. The rescaled nondominated criterion vectors and the DM's normalized preference values are then used to re-train the *FFANN*. The re-trained *FFANN* is then used to search for improved solutions. This process is repeated for four more times. Table 3 lists the solutions obtained at each iteration.

Table 3 About Here

As seen, the best solution was found in Iteration 3. Thus, the final solution is $\mathbf{z}^{\mathbf{fin}} = (18.96599, -2.52878, 20.36740)$. This represents a 98.43% $\left(\frac{42.27592-33.07733}{42.42288-33.07733} \times 100\%\right)$ achievement of the DM's value function value from that of the nadir point $\mathbf{z}^{\mathbf{nad}}$ to that of the optimal point $\mathbf{z}^{\mathbf{opt}}$, and a 97.93% $\left(\frac{42.27592-35.50926}{42.42288-35.50926} \times 100\%\right)$ achievement from that of the worst nondominated point $\mathbf{z}^{\mathbf{worst}}$.

6. Computational Experiments

In this section, we conduct computational experiments in order to test the Interactive *FFANN* Procedure against the Tchebycheff Method, which has tested well in a previous study (Buchanan and Daellenbach 1987).

Similar to the previous section, for each problem we assumed a hypothetical DM with a particular value function. This is useful for test purposes because it enables us to determine an optimal solution for each problem ahead of time and helps us in providing preference information by acting as the DM. This is especially useful when two procedures are compared computationally because it provides the same preference information required by both of the procedures. In the tests, the value function, of course, is only used in the preference elicitation phase and not in the search for improved solutions. With this experimental design, the performance of the Interactive *FFANN* Procedure is measured along four dimensions: (1) solution quality, (2) problem size, (3) type of value function, and (4) *FFANN* architecture.

6.1 Test Problems

The MOLP test problems used in the experiments were generated using the random problem generation capability in ADBASE (Steuer 1992), the same capability used in other interactive multiple objective programming computational studies such as those reported in Reeves and Franz (1985), Steuer (1986), and Buchanan and Daellenbach (1987). The problem sizes, defined by $k \times m \times n$ (m is the number of linear constraints), used in the experiments are $3 \times 5 \times 6$, $5 \times 5 \times 10$, $5 \times 8 \times 15$, $5 \times 10 \times 20$ and $6 \times 50 \times 100$. ADBASE was used to find the criterion vectors of all efficient extreme points (see

Table 4) for all problem sizes except for those in the $6 \times 50 \times 100$ category. The difficulty with the $6 \times 50 \times 100$ category is that the tens of thousands of efficient extreme points that such problems are likely to have is beyond the capability of any currently existing code. For all problems except those in the $6 \times 50 \times 100$ category, $\mathbf{z}^{\max}$, $\mathbf{z}^{\text{nad}}$ and $\mathbf{z}^{\text{worst}}$ were obtained from the generation of all efficient extreme points. For the $6 \times 50 \times 100$ problems, the $\mathbf{z}^{\text{nad}}$ criterion vectors were estimated from their payoff tables.

Table 4 About Here

6.2 Value Functions

In the experiments we used four different value functions of L_p-metric form with $p = 1$, $p = 2$, $p = 4$ and $p = \infty$,

$$V_p(\mathbf{z}) = K - \left[\sum_{i=1}^{k} [\lambda_i (z_i^{\max} - z_i)]^p \right]^{\frac{1}{p}}, \tag{6.1}$$

where the λ_i are given by

$$\lambda_i = \frac{1}{z_i^{\max} - z_i^{\text{nad}}} \left[\sum_{j=1}^{k} \frac{1}{(z_j^{\max} - z_j^{\text{nad}})} \right]^{-1},$$

and K is a constant to ensure that all value function values are positive. When $p = \infty$ we note that (6.1) reduces to

$$V_\infty(\mathbf{z}) = K - \max_{1 \le i \le k} \{\lambda_i (z_i^{\max} - z_i)\},$$

in which case the value function is nondifferentiable. Thus, it will be interesting to study the performance of the Interactive *FFANN* Procedure when dealing with this potentially difficult mapping.

In the computational experiments, we set the number of iterations to $t = 5$, (except for with the $6 \times 50 \times 100$ problems in which case we used $t = 6$), and evaluated $P = 7$ nondominated criterion vectors at each iteration. Note that in the experiments the principal eigenvector of the reciprocal comparison matrix is identical to the normalized preference values since the hypothetical DM makes each preference judgment according to the pre-specified value function. Therefore, the test results will be the same regardless of which method is used for eliciting preference information (pairwise comparisons or direct assessment).

6.3 Solution Quality

In the computational tests, except for the $6 \times 50 \times 100$ problems, the solution quality of a $\mathbf{z} \in Z$ is measured by comparing its preference value against that of $\mathbf{z}^{\mathbf{opt}}$ relative to that of $\mathbf{z}^{\mathbf{worst}}$ by means of

$$\frac{V(\mathbf{z}) - V(\mathbf{z}^{\mathbf{worst}})}{V(\mathbf{z}^{\mathbf{opt}}) - V(\mathbf{z}^{\mathbf{worst}})} \times 100. \tag{6.2}$$

The convenience of this measure is that the quality of $\mathbf{z}^{\mathbf{worst}}$ is 0 and the quality of $\mathbf{z}^{\mathbf{opt}}$ is 100. For the $6 \times 50 \times 100$ problems, the quality of a $\mathbf{z} \in Z$ is measured by comparing its preference value against that of $\mathbf{z}^{\mathbf{opt}}$ relative to that of the estimated nadir point $\mathbf{z}^{\mathbf{nad}}$ by means of

$$\frac{V(\mathbf{z}) - V(\mathbf{z}^{\mathbf{worst}})}{V(\mathbf{z}^{\mathbf{opt}}) - V(\mathbf{z}^{\mathbf{nad}})} \times 100,$$

because $\mathbf{z}^{\mathbf{worst}}$ is not available.

6.4 Experimental Results

In the experiments, fifty (ten in each problem size category) MOLP test problems were employed. In testing the Interactive *FFANN* Procedure, different *FFANN* structures with different numbers of hidden nodes were employed. We report the computational results for four different *FFANN* structures with no, one, two or six hidden nodes in one hidden layer respectively. In Tables 5–8, we summarize the quality of the final solutions obtained from the Interactive *FFANN* Procedure as compared against those obtained from the Tchebycheff Method when using the L_1-, L_2-, L_4- and L_∞-metric value functions. More computational results with different quality measures are reported in Sun (1992).

In Table 5, the Interactive *FFANN* Procedure was run for the L_1-metric value function, with a neural network structure without any hidden nodes, resulting in higher quality solutions than with the Tchebycheff Method, in terms of average as well as best and worst qualities. In this case, the output node is the only node which performs a computational function. If this node were to have a linear activation function, the *FFANN* would reduce to a linear regression model and be able to represent the DM's linear value function exactly. However, in this case the nonlinear nature of the activation function of the output node introduces "imprecision" into the *FFANN*. Nevertheless, this imprecision hardly has an impact on the performance, as the Interactive *FFANN* Procedure correctly identifies the optimal solution, within five iterations, for 35 out of the 50 test problems, and approximates the optimal solution closely (within one to four percent) for the remaining problems.

In Table 6, three different neural network structures were used for the L_2-metric value function. From this table, we see that superior results were obtained using the Interactive *FFANN* Procedure, as long as at least one hidden node is used. In Tables 7 and 8, generally better results were obtained

using the Interactive *FFANN* Procedure than with the Tchebycheff Method, but at least two hidden nodes were required because of the more difficult L_4- and L_∞-metric value functions. Particularly for the L_∞-metric value function, the difference in performance becomes more favorable for the Interactive *FFANN* Procedure as the problem size increases.

Keeping in mind the fact that the test results were generated by the progenitors of the new procedure, we nevertheless feel that the results are very encouraging and that the possibilities for embedding artificial neural network technology in the interactive procedures of multiple objective programming are promising.

Tables 5–8 About Here

6.5. Computational Effort

The time required to train a *FFANN* depends on several factors, such as the number of patterns in the training set, the number of inputs (*i.e.* k, the number of objectives), the number of hidden nodes in the *FFANN*, the stopping criteria, and the complexity of the mapping the *FFANN* is to represent. In order to fully assess the usefulness of the *FFANN* approach, we report the average computational effort required to train *FFANN*s for ten $6 \times 50 \times 100$ MOLP problems in Table 9. Smaller size problems were trained within a few seconds.

Table 9 About Here

From Table 9 we see that, as expected, the average computational effort increases as the number of nodes in the hidden layer and the number of patterns in the training set increase. Nevertheless, even for *FFANN* configurations with 6 hidden nodes and for training sets with 44 patterns the training times are reasonable.

7. Concluding Remarks

In this paper, we present an Interactive *FFANN* Procedure for solving multiple objective programming problems using feed-forward artificial neural networks. In the procedure, the DM has the option of articulating his or her preference information either by directly assigning a preference value to each new solution or by making pairwise comparisons in a way similar to the AHP. Since preference structures may be very complex, a *FFANN* is used in the procedure because of its ability to capture and represent complicated mappings. Because the DM's aspirations may evolve over the course of the solution process, the *FFANN* has the chance to adapt to any such changes as the *FFANN* is re-trained

at each iteration.

From the computational results, it is evident that good solutions have been obtained, at least for the test problems and value functions used. Also, the procedure is relatively robust in that similar solutions are obtained when different *FFANN* structures are employed. Computer-time-wise, because of the re-training of the *FFANN* at each iteration, the Interactive *FFANN* Procedure can be expected to take more time than other interactive procedures. However, in an era of rapidly decreasing computer costs, solution quality may be the most important issue for many users. Typically, *FFANN*s used in the Interactive *FFANN* Procedure can be trained within a few seconds.

All of the computations performed in this paper were conducted on the University of Georgia IBM ES 9000 Model 720 computer.

References

Aarts, E. H. L. and J. H. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, Wiley, New York, 1989.

Arbel, A. and S. Oren, "Priority-Based Interactive Multicriteria Optimization Algorithm." In Y. Sawaragi, K. Inoue and H. Nakayama (eds.), *Toward Interactive and Intelligent Decision Support Systems*: Volume 1, *Lecture Notes in Economics and Mathematical Systems*, Vol. 285, Springer-Verlag, Berlin, 163–171, 1987.

Bazaraa, M. S. and C. M. Shetty, *Nonlinear Programming, Theory and Algorithms*, Wiley, New York, 1979.

Barzilai, J. and B. Golany, "Deriving Weights from Pairwise Comparison Matrices: The Additive Case," *Operations Research Letters*, **9** (6), 407–410, 1990.

Benayoun, R., J. de Montgolfier, J. Tergny and O. Larichev, "Linear Programming with Multiple Objective Functions: Step Method (STEM)," *Mathematical Programming*, **1** (3), 366–375, 1971.

Buchanan, J. T. and H. G. Daellenbach, "A Comparative Evaluation of Interactive Solution Methods for Multiple Objective Decision Models," *European Journal of Operational Research*, **29** (3), 353–359, 1987.

Burden, R. L. and J. D. Faires, *Numerical Analysis*, Fourth Edition, PWS-Kent, Boston, 1989.

Burke, L. I. and J. P. Ignizio, "Neural Networks and Operations Research: An Overview," *Computers & Operations Research*, **19** (2), 179–189, 1992.

Cybenko, G., "Approximations by Superpositions of a Sigmoidal Function," Technical Report No. 856, University of Illinois, Champaign-Urbana, 1989.

Dyer, J. S., "Remarks on the Analytic Hierarchy Process," *Management Science*, **36** (3), 249–258, 1990.

Expert Choice, *Expert Choice, Version 8.0*, Expert Choice, Inc., Pittsburgh, Pennsylvania, 1992.

Farquhar, P. H., "Utility Assessment Methods," *Management Science*, **30** (11), 1283–1300, 1984.

Fishburn, P. C., "Lexicographic Orders, Utilities and Decision Rules: A Survey," *Management Science*, **20** (11), 1442–1471, 1974.

Fishburn, P. C., "Multiattribute Nonlinear Utility Theory," *Management Science*, **30** (11), 1301–1310, 1984.

Gardiner, L. R. and R. E. Steuer, "Unified Interactive Multiple Objective Programming," *European Journal of Operational Research*, **74** (3), 391–406, 1994.

Gass, S. I., "A Process for Determining Priorities and Weights for Large-Scale Linear Goal Programmes," *Journal of the Operational Research Society*, **37** (8), 779–785, 1986.

Geoffrion, A. M., J. S. Dyer and A. Feinberg, "An Interactive Approach for Multicriterion Optimization, with an Application to the Operation of an Academic Department," *Management Science*, **19** (4), 357–368, 1972.

Hecht-Nielsen, R., "Kolmogorov's Mapping Neural Network Existence Theorem," *Proceedings of the IEEE First International Conference on Neural Networks*, **1**, 11–14, 1987.

Hopfield, J. J. and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, **52**, 141–152, 1985.

Keeney, R. L. and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley, New York, 1976.

Isermann, H. and R. E. Steuer, "Computational Experience Concerning Payoff Tables and Minimum Criterion Values over the Efficient Set," *European Journal of Operational Research*, **33** (1), 91–97, 1988.

Kok, M. and F. Lootsma, "Pairwise-Comparison Methods in Multiple Objective Programming with Applications in a Long-Term Energy-Planning Model," *European Journal of Operational Research*, **22** (1), 44–55, 1985.

Korhonen, P., "The Specification of a Reference Direction Using the Analytic Hierarchy Process," *Mathematical Modelling*, **9** (3–5), 361–368, 1987a.

Korhonen, P., "VIG – A Visual Interactive Support System for Multiple Criteria Decision Making," *Belgian Journal of Operations Research*, **27** (1), 3–15, 1987b.

Korhonen, P., S. Salo, and R. E. Steuer, "A Heuristic for Estimating Nadir Criterion Values in Multiple Objective Linear Programming," Working Paper, Helsinki School of Economics, Helsinki, Finland, 1994.

Korhonen, P. and J. Wallenius, "Using Qualitative Data in Multiple Objective Linear Programming," *European Journal of Operational Research*, **48** (1), 81–87, 1990.

Lasdon, L. S. and A. D. Waren, "GRG2 User's Guide," University of Texas, Austin, 1989.

Luenberger, D. G., *Linear and Nonlinear Programming*, Second Edition, Addison-Wesley, Reading, Massachusetts, 1984.

Malakooti, B. and Y. Zhou, "An Adaptive Feedforward Artificial Neural Network with Application to Multiple Criteria Decision Making," *Management Science*, **40**, forthcoming, 1994.

Másson, E. and Y. J. Wang, "Introduction to Computation and Learning in Artificial Neural Networks," *European Journal of Operational Research*, **47** (1), 1–28, 1990.

Polak, E., *Computational Methods in Optimization*, Academic Press, New York, 1971.

Reeves, G. R. and L. S. Franz, "A Simplified Interactive Multiple Objective Linear Programming Procedure," *Computers & Operations Research*, **12** (6), 589–601, 1985.

Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation." In *Parallel Distributed Processing, Volume* 1: *Foundations*, D. E. Rumelhart and J. L. McClelland and the PDP Research Group (eds.), MIT Press, Cambridge, Massachusetts, 318–362, 1986.

Rumelhart, D. E., J. L. McClelland and the PDP Research Group (eds.), *Parallel Distributed Processing, Volume* 1: *Foundations*, MIT Press, Cambridge, Massachusetts, 1986.

Saaty, T. L., *Multicriteria Decision Making: The Analytic Hierarchy Process (Revised Edition)*, RWS Publications, Pittsburgh, Pennsylvania, 1988.

Saaty, T. L., "An Exposition of the AHP in Reply to the Paper 'Remarks on the Analytic Hierarchy Process,'" *Management Science*, **36** (3), 259–268, 1990.

Schoner, B., W. C. Wedley and E. U. Choo, "A Rejoinder to Forman on AHP, With Emphasis on the Requirements of Composite Ratio Scales," *Decision Sciences*, **23** (2), 509–517, 1992.

Steuer, R. E., *Multiple Criteria Optimization: Theory, Computation, and Application*, Wiley, New York, 1986.

Steuer, R. E., "Manual for the ADBASE Multiple Objective Linear Programming Package," Faculty of Management Science, University of Georgia, Athens, 1992.

Steuer, R. E., and E.–U. Choo, "An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming," *Mathematical Programming*, **26** (1), 326–344, 1983.

Sun, M., "Interactive Multiple Objective Programming Procedures via Adaptive Random Search and Feed-Forward Artificial Neural Networks," Ph.D. dissertation, Terry College of Business, University of Georgia, Athens, GA, 1992.

Sun, M., A. Stam and R. E. Steuer, "Solving Interactive Multiple Objective Programming Problems Using Feed-Forward Artificial Neural Networks," Working Paper, Terry College of Business, University of Georgia, 1992.

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

Wang, J. and V. Chankong, "Recurrent Neural Networks for Linear Programming: Analysis and Decision Principles," *Computers & Operations Research*, **19** (2), 297–311, 1992.

Wang, J. and B. Malakooti, "A Feedforward Neural Network for Multiple Criteria Decision Making," *Computers & Operations Research*, **19** (2), 151–167, 1992.

Wasserman, P. D., *Neural Computing, Theory and Practice*, Van Nostrand Reinhold, New York, 1989.

Wierzbicki, A. P., "A Mathematical Basis for Satisficing Decision Making," *Mathematical Modelling*, **3**, 391–405, 1982.

Winkler, R. L., "Decision Modeling and Rational Choice: AHP and Utility Theory," *Management Science*, **36** (3), 247–248, 1990.

Yu, P.–L., *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions*, Plenum Press, New York, 1985.

Zahedi, F., "An Introduction to Neural Networks and a Comparison with Artificial Intelligence and Expert Systems," *Interfaces*, **21** (2), 25–38, 1991.

Zionts, S. and J. Wallenius, "An Interactive Multiple Objective Linear Programming Method for a Class of Underlying Nonlinear Utility Functions," *Management Science*, **29** (5), 519–529, 1983.

Zwietering, P. J., E. H. L. Aarts and J. Wessels, *The Classification Capabilities of Exact Two-Layered Perceptrons, Memorandum COSOR 91–09*, Eindhoven University of Technology, 1991.

Appendix A: A *FFANN* Training Algorithm

This appendix presents an algorithm for training *FFANN*s with multiple layers, which we use in our Interactive *FFANN* Procedure. The algorithm is developed based on the error back-propagation algorithm (Rumelhart, Hinton and Williams 1986), and uses unconstrained nonlinear optimization techniques. Specifically, the algorithm uses a combination of the Golden Section Method and a "doubling and halving" line search strategy, and the Polak and Ribiere conjugate gradient direction. In the following, we discuss the mathematical details of the training algorithm, present the line search procedure, and outline the training algorithm.

A1. Mathematical Details

In the training process, the node biases, θ_k^i, are treated the same as other connectivity weights. Actually, by adding a single node $v_{n_0+1}^0$ to the input layer, connecting it to all nodes in all other layers, and assigning $v_{n_0+1}^0$ an input value of 1, the connectivity weight $w_{k,\,n_0+1}^{i0}$ is the bias θ_k^i of node v_k^i, *i.e.*,

$$\theta_k^i = w_{k,\,n_0+1}^{i0}.$$

Suppose that $\mathbf{z}_q \in \Re^{n_0}$ is the q^{th} input vector and $\mathbf{t}_q \in \Re^{n_m}$ is the associated desired output vector in the training set. The compound vector $(\mathbf{z}_q,\, \mathbf{t}_q) \in \Re^{n_0+n_m}$ is called a training pattern. Let the number of patterns in the training set be denoted by Q.

When $\mathbf{z}_q$ is presented to the network, the *FFANN* maps it to an output vector $\mathbf{u}_q$ based on (2.1–2.2). The error measure E_q for the q^{th} training pattern is defined as

$$E_q = \tfrac{1}{2} \sum_{j=1}^{n_m} (t_{qj} - u_{qj})^2. \tag{A.1}$$

Thus, E_q is the sum over all output nodes of the squared differences between the computed and desired outputs. In our application, $n_m = 1$. For a given topology of the *FFANN* and a given set of training patterns, E_q is a function of the connectivity weights in W and can be written as $E_q(W)$. Summing over all training patterns, the overall error measure over all Q training patterns is given by

$$E(W) = \sum_{q=1}^{Q} E_q(W) = \tfrac{1}{2} \sum_{q=1}^{Q} \sum_{j=1}^{n_m} (t_{qj} - u_{qj})^2, \tag{A.2}$$

When a *FFANN* is trained, we try to adjust the values of the components of W so as to minimize $E(W)$. The partial derivative of E_q with respect to the connectivity weight w_{kr}^{ij} is given by

$$\frac{\partial E_q(W)}{\partial w_{kr}^{ij}} = -\delta_{qk}^i u_{qr}^j, \quad 0 < i < m, \tag{A.3}$$

where δ_{qk}^i is the error signal of node v_k^i and u_{qr}^j is the computed output of node v_r^j for the q^{th} training

pattern, respectively. If $i = m$, δ_{qk}^{m} is determined by

$$\delta_{qk}^{m} = (t_{qk} - u_{qk}^{m})\, f'(z_{qk}^{m}), \tag{A.4}$$

and, if $0 < i < m$, δ_{qk}^{i} is computed recursively in terms of the error signals of all the nodes to which it directly connects as shown in

$$\delta_{qk}^{i} = f'(z_{qk}^{i}) \sum_{s=i+1}^{m} \sum_{t=1}^{n_s} \delta_{qt}^{s} w_{tk}^{si}, \quad 0 < i < m, \tag{A.5}$$

where $f'(z_{qk}^{i})$ is the first derivative of the node activation function of v_k^i evaluated at z_{qk}^{i} and z_{qk}^{i} is determined by (2.1) for the q^{th} training pattern. The first derivative of the logistic node activation function in (2.2) is given by

$$f'(z_{qk}^{i}) = \tfrac{1}{T}\, u_{qk}^{i}\, (1 - u_{qk}^{i}). \tag{A.6}$$

Denote the gradient of $E(W)$ with respect to W by G, *i.e.* $G = \nabla E(W) = \{g_{kr}^{ij}\}$, for $i = 1, ..., m$; $j = 0, ..., m-1$; $k = 1, ..., n_i$, and $r = 1, ..., n_j$, then g_{kr}^{ij} is given by

$$g_{kr}^{ij} = \frac{\partial E(W)}{\partial w_{kr}^{ij}} = \sum_{q=1}^{Q} \frac{\partial E_q(W)}{\partial w_{kr}^{ij}} = -\sum_{q=1}^{Q} \delta_{qk}^{i} u_{qr}^{j}\,. \tag{A.7}$$

The connectivity weights are updated according to the following rule

$$W_{h+1} = W_h + \eta D_h, \tag{A.8}$$

where h is the iteration counter, sometimes called learning time, η is the learning rate, D_h is the search direction at iteration h, and W_h is the set of connectivity weights at the beginning of iteration h. Letting the set of values of G at iteration h be denoted by G_h, the search direction D_h is determined by

$$D_h = -G_h + \alpha_h D_{h-1}. \tag{A.9}$$

In our training algorithm, α_h is determined by a combination of the Polak and Ribiere gradient conjugate direction (Polak 1971; Luenberger 1984) and a momentum factor. In the Polak and Ribiere gradient conjugate direction, α_h is determined by

$$\alpha_h = \frac{(G_h - G_{h-1}) \cdot G_{h-1}}{\| G_{h-1} \|^2}\,. \tag{A.10}$$

A2. Line Search Procedure

For a given training set and W_h, the error measure E at iteration h is a function of the search direction D_h and the learning rate η. For a given search direction, E becomes a function of the learning rate η. Let us denote this error measure by $E(W_h + \eta D_h)$. There are many line search methods to determine a value η^* for η at which $E(W + \eta D_h)$ is approximately minimized along D_h. In the following line search procedure, we use a "doubling and halving" strategy to locate the initial interval of uncertainty, and the Golden Section method (Bazaraa and Shetty 1979; Luenberger 1984) to find η^*.

Initialization:

Step 0. Let $\epsilon > 0$ be small. Let $\zeta = E(W_h)$ and compute $\zeta_1 = E(W_h + \eta D_h)$. If $\zeta_1 > \zeta$, execute Step 0A; otherwise execute Step 0B.

Step 0A. Let $d = -1.0$, $\eta = \eta - \Delta\eta$, and compute $\zeta_2 = E(W_h + \eta D_h)$.

Step 0B. Let $\eta = \eta + \Delta\eta$ and compute $\zeta_2 = E(W_h + \eta D_h)$. If $\zeta_2 > \zeta_1$, then let $d = -1.0$, $\omega = \zeta_1$, $\zeta_1 = \zeta_2$, $\zeta_2 = \omega$, and $\eta = \eta - \Delta\eta$; otherwise let $d = 1.0$.

Doubling and Halving:

Step 1. Let $\Delta\eta = 2\Delta\eta$. If both $\Delta\eta > \eta$ and $d = -1.0$, then let $\eta_1 = 0$, $\Delta\eta = 0.5\Delta\eta$, $\eta_2 = \eta + \Delta\eta$ and go to Step 4; otherwise let $\eta = \eta + d\Delta\eta$ and compute $\zeta_3 = E(W_h + \eta D_h)$.

Step 2. If $\zeta_3 < \zeta_2$, let $\zeta_1 = \zeta_2$, $\zeta_2 = \zeta_3$, and go to Step 1.

Step 3. Let $\Delta\eta = 0.5\Delta\eta$ and $\eta = \eta - d\Delta\eta$, and compute $\zeta_4 = E(W_h + \eta D_h)$. If $\zeta_4 \leq \zeta_2$, then let $\eta_1 = \eta - \Delta\eta$, $\eta_2 = \eta + \Delta\eta$; otherwise let $\eta_1 = \eta - 2\mathrm{d}\Delta\eta$, $\eta_2 = \eta$. If $d = -1.0$, then let $\eta_t = \eta_1$, $\eta_1 = \eta_2$, $\eta_2 = \eta_t$.

Golden Section:

Step 4. Let $\eta_3 = \eta_2 - 0.618(\eta_2 - \eta_1)$ and $\eta_4 = \eta_1 + 0.618(\eta_2 - \eta_1)$. Compute $\zeta_3 = E(W_h + \eta_3 D_h)$ and $\zeta_4 = E(W_h + \eta_4 D_h)$.

Step 5. If $\zeta_4 \leq \zeta_3$, execute Step 6. Otherwise execute Step 7.

Step 6. Let $\eta_1 = \eta_3$, $\eta_3 = \eta_4$ and $\zeta_3 = \zeta_4$. If $(\eta_2 - \eta_1) < \epsilon$, then go to Step 8. Otherwise let $\eta_4 = \eta_1 + 0.618(\eta_2 - \eta_1)$ and compute $\zeta_4 = E(W_h + \eta_4 D_h)$. Go to Step 5.

Step 7. Let $\eta_2 = \eta_4$, $\eta_4 = \eta_3$ and $\zeta_4 = \zeta_3$. If $(\eta_2 - \eta_1) < \epsilon$, then go to Step 9. Otherwise let $\eta_3 = \eta_2 - 0.618(\eta_2 - \eta_1)$ and compute $\zeta_3 = E(W_h + \eta_3 D_h)$. Go to Step 5.

Step 8. Let $\eta = \eta_3$, $\eta^* = \eta_3$ and $\zeta = \zeta_3$. Stop.

Step 9. Let $\eta = \eta_4$, $\eta^* = \eta_4$ and $\zeta = \zeta_4$. Stop.

The values of η and $\Delta\eta$ need to be initialized for the first iteration. For the following iterations, the ending values of the previous iteration are used as the beginning values of the current iteration.

A3. The Training Algorithm

Step 0. Initialize the connectivity weights W_1 to small positive values. Let $\epsilon_1 > 0$ and $\epsilon_2 > 0$ be small. Let $\alpha_0 > 0$ be a pre-determined constant. Set the iteration counter to $h = 1$.

Step 1. Compute G_h according to (A.1–A.7), let the search direction be $D_h = -G_h$.

Step 2. Perform a one-dimensional search to minimize $E(W_h + \eta D_h)$ with the line search procedure discussed above. Let η^* be the value of η corresponding to the minimum of $E(W_h + \eta D_h)$. Update the connectivity weights by setting $W_{h+1} = W_h + \eta^* D_h$. If $E(W_h) - E(W_{h+1}) < \epsilon_1$, then Stop. Let $h = h + 1$. If (h mod $|W|$) = 0, where $|W|$ is the cardinality of W, go to Step 1.

Step 3. Compute G_h according to (A.1–A.7). If $\| G_h \| < \epsilon_2$, then Stop. Otherwise, compute the value of α according to (A.10). If $\alpha > \alpha_0$, then let $\alpha = \alpha_0$. Let the new search direction be $D_h = -G_h + \alpha D_{h-1}$. Go to Step 2.

Table 1. Criterion Vectors of the First Iteration for the Example Problem

Solution	z_1	z_2	z_3	$V_4(\mathbf{z})$
1	24.35460	−11.54862	27.64540	39.13516
2	−5.69318	14.18864	−3.93636	35.80484
3	22.86093	2.25756	−7.88655	37.39273
4	−4.61749	7.45757	14.19598	37.69865
5	29.56935	−9.20832	6.82426	38.94388
6	2.32488	−6.27676	34.03545	38.96402
7	−3.17575	1.33950	27.90013	38.28441
$\mathbf{z}^{\max}$	33.10000	14.00000	39.25000	50.00000
$\mathbf{z}^{\text{nad}}$	−7.25000	−16.41200	−9.20700	33.07733

Table 2. Normalized Criterion Vectors of the First Iteration for the Example Problem

Solution	z'_1	z'_2	z'_3	$v_4(\mathbf{z})$
1	0.78326	0.15733	0.76052	0.35797
2	0.03858	0.98993	0.10877	0.16117
3	0.74624	0.60396	0.02725	0.25501
4	0.06524	0.77218	0.48296	0.27308
5	0.91250	0.23304	0.33083	0.34667
6	0.23730	0.32787	0.89239	0.34786
7	0.10097	0.57426	0.76577	0.30770
$\mathbf{z}^{\max}$	1.00000	1.00000	1.00000	1.00000
$\mathbf{z}^{\text{nad}}$	0.00000	0.00000	0.00000	0.00000

Table 3. Iteration by Iteration Solutions for the Example Problem

Iteration (h)	$z_1^{(h)}$	$z_2^{(h)}$	$z_3^{(h)}$	$V_4(\mathbf{z}^{(h)})$
1	19.16292	−4.44382	24.18538	41.80951
2	18.81381	−4.06534	23.91694	41.92090
3	18.96599	−2.52878	20.36740	42.27592
4	18.70442	−3.13536	22.33424	42.17731
5	18.70442	−3.13536	22.33424	42.17731
$\mathbf{z}^{\mathbf{opt}}$	**16.51700**	**−0.88555**	**18.97000**	**42.42288**

Table 4. Number of Efficient Extreme Points over the Ten Test Problems for Each Problem Size

Problem Size	Number of Efficient Extreme Points		
	Minimum	Maximum	Average
3 × 5 × 6	5	19	11.9
5 × 5 × 10	6	83	27.7
5 × 8 × 15	29	554	158.0
5 × 10 × 20	226	1131	417.7

Table 5. Final Solution Quality with L_1-Metric Value Function

Problem Size	Interactive *FFANN* Procedure (No Hidden Nodes in the *FFANN*)			Tchebycheff Method		
	Worst	Best	Average	Worst	Best	Average
3 × 5 × 6	98.18	100.00	99.82	94.95	99.95	98.48
5 × 5 × 10	100.00	100.00	100.00	90.50	99.98	97.81
5 × 8 × 15	97.09	100.00	99.71	84.89	100.00	97.19
5 × 10 × 20	95.97	100.00	99.26	91.12	99.59	97.69
6 × 50 × 100	99.62	99.97	99.81	93.40	100.00	96.98

Table 6. Final Solution Quality with L_2-Metric Value Function

MOLP Problem Size	Interactive *FFANN* Procedure									Tchebycheff Method		
	Number of Hidden Nodes in the *FFANN*											
	0			1			2					
	Worst	Best	Average	Worst	Best	Average	Worst	Best	Average	Worst	Best	Average
3 x 5 x 6	61.42	100.00	94.14	90.23	100.00	98.19	97.68	100.00	99.48	92.35	99.95	97.52
5 x 5 x 10	83.35	100.00	92.94	85.98	100.00	98.67	89.39	100.00	99.00	93.46	100.00	98.64
5 x 8 x 15	96.46	99.56	98.27	97.10	99.96	98.76	97.60	99.90	99.01	93.57	99.98	97.89
5 x 10 x 20	92.32	99.99	97.59	95.41	100.00	98.63	95.41	100.00	98.77	92.05	98.87	97.00
6 x 50 x 100	98.29	99.84	99.27	98.28	99.84	99.27	97.29	99.84	99.30	89.03	98.87	96.39

Table 7. Final Solution Quality with L_4-Metric Value Function

Problem Size	Interactive *FFANN* Procedure						Tchebycheff Method		
	Number of Hidden Nodes in the *FFANN*								
	2			6					
	Worst	Best	Average	Worst	Best	Average	Worst	Best	Average
3 x 5 x 6	93.88	99.98	98.13	93.71	99.98	98.13	95.00	100.00	98.78
5 x 5 x 10	86.63	100.00	97.70	84.05	100.00	97.21	74.16	99.67	94.15
5 x 8 x 15	94.45	99.96	97.61	93.06	99.96	97.15	74.44	99.70	95.15
5 x 10 x 20	95.59	99.77	98.35	92.89	99.91	98.11	81.39	98.87	95.36
6 x 50 x 100	93.53	99.06	97.13	97.07	99.78	99.02	93.19	98.86	96.24

Table 8. Final Solution Quality with L_∞-Metric Value Function

Problem Size	Interactive *FFANN* Procedure						Tchebycheff Method		
	Number of Hidden Nodes in the *FFANN*								
	2			6					
	Worst	Best	Average	Worst	Best	Average	Worst	Best	Average
3 x 5 x 6	88.35	100.00	96.28	84.21	100.00	92.83	92.08	99.00	95.94
5 x 5 x 10	85.65	97.55	92.13	83.16	96.70	92.55	52.27	95.34	86.75
5 x 8 x 15	72.42	98.26	91.63	79.81	99.71	92.28	68.34	98.49	87.98
5 x 10 x 20	88.59	98.20	94.84	85.16	98.42	91.94	75.75	98.64	87.14
6 x 50 x 100	72.09	93.09	76.63	69.23	96.68	84.88	49.39	91.19	71.47

Table 9. Average Time to Train Six-Input Node *FFANN*s for $6 \times 50 \times 100$ MOLP

Number of Hidden Nodes	Number of Training Patterns					
	9	16	23	30	37	44
	L_1-Metric Value Function					
0	2.09	0.56	0.75	1.00	1.21	1.55
2	4.91	3.39	12.06	22.50	67.32	105.67
	L_2-Metric Value Function					
0	2.14	0.72	0.90	1.15	1.39	1.77
1	2.34	2.24	8.86	13.25	26.85	31.29
2	3.72	9.76	23.95	48.10	60.65	77.76
4	8.27	15.64	48.74	71.71	139.10	164.23
6	18.20	52.26	80.95	167.13	233.43	208.33
	L_4-Metric Value Function					
0	2.16	0.94	1.02	1.33	1.76	1.95
2	5.33	15.54	45.20	78.60	94.58	110.81
4	8.25	49.83	81.02	123.87	164.09	187.17
6	9.10	66.28	138.97	188.52	228.70	271.43
	L_∞-Metric Value Function					
0	2.16	0.81	0.93	1.19	1.47	1.69
2	14.77	47.27	66.10	74.61	99.32	94.28
4	20.39	76.30	89.59	118.05	169.86	199.23
6	29.27	105.59	131.77	183.46	221.38	239.69

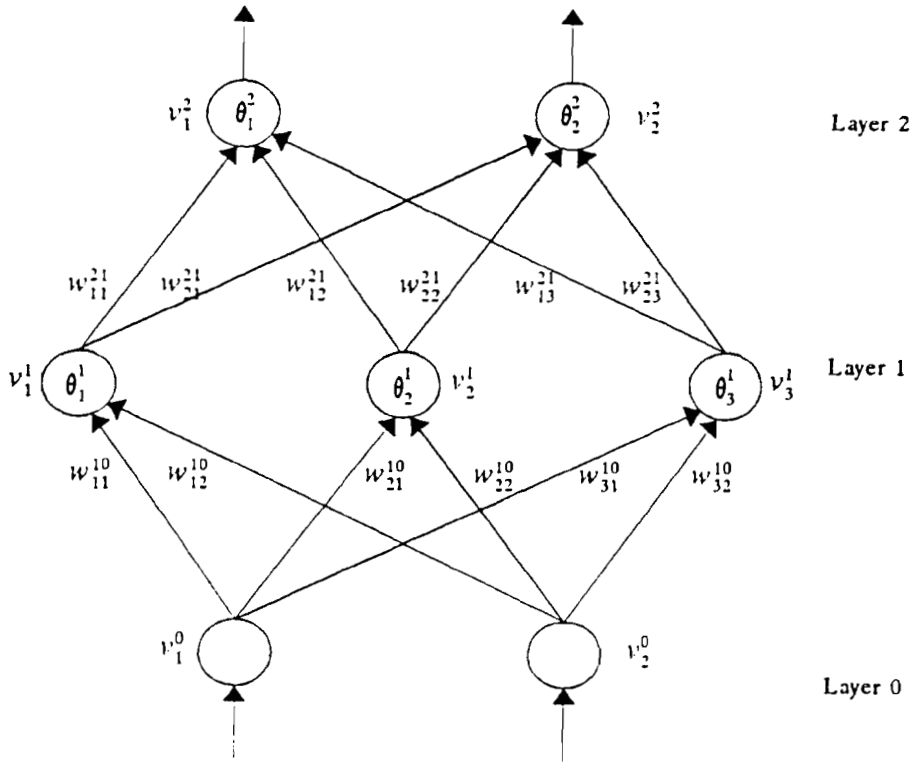

Figure 1: A *FFANN* without Direct Connections from the Input Layer to the Output Layer.

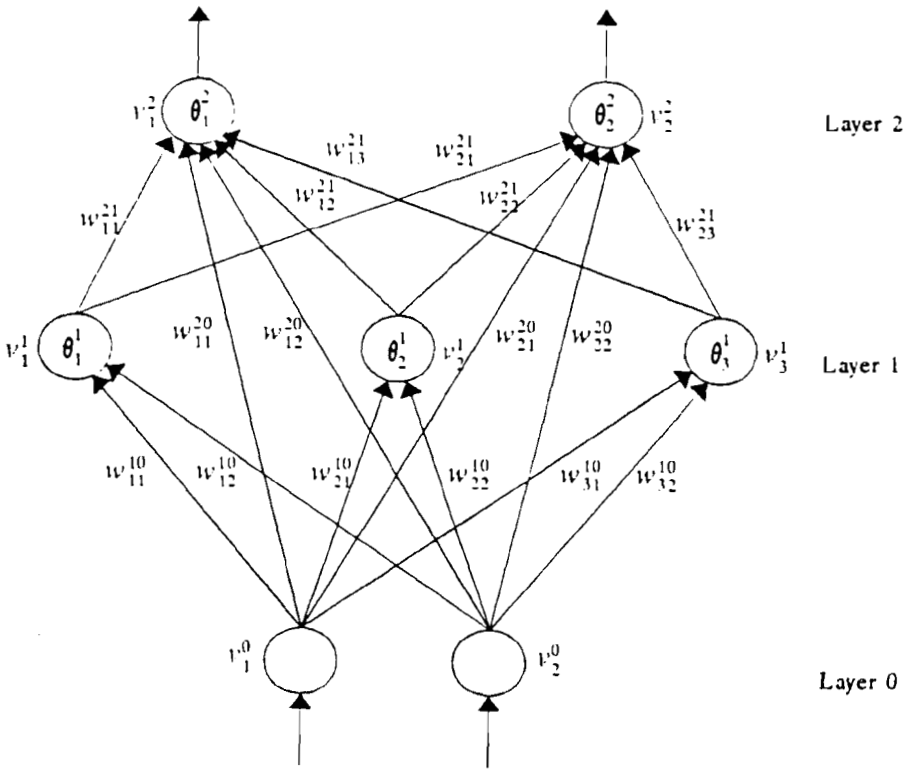

Figure 2: A Fully Connected *FFANN*.

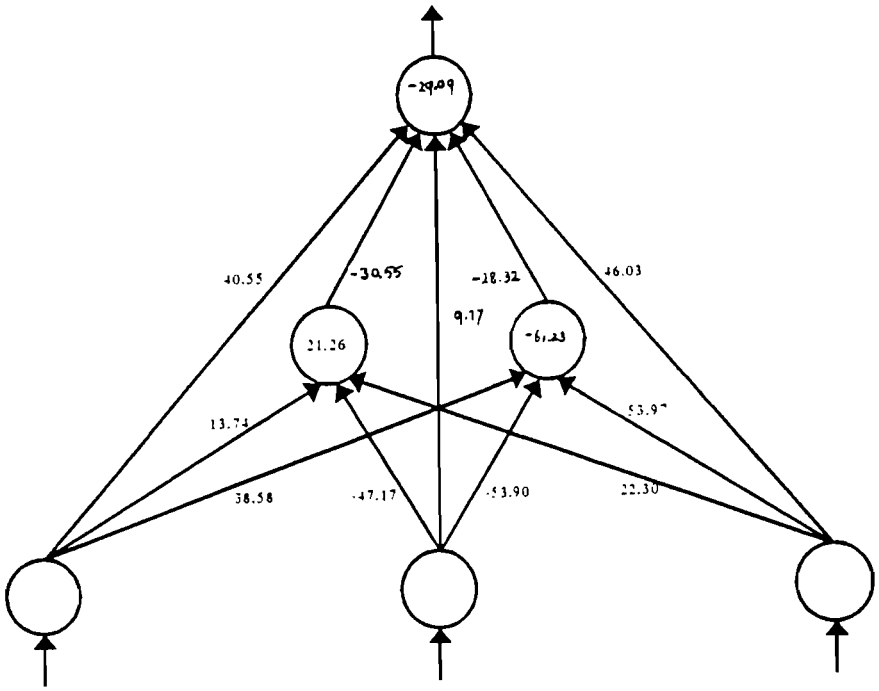

Figure 3: *FFANN* Architecture and Connectivity Weights for the Example Problem.