Neural networks and production planning

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

Memorandum COSOR 91-15

Neural Networks and Production Planning

P.J. Zwietering
M.J.A.L. van Kraaij
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, July 1991
The Netherlands

Neural Networks and Production Planning

P.J. Zwietering[1], M.J.A.L. van Kraaij[1], E.H.L. Aarts[1,2] and J. Wessels[1]

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

Because of the combination of classification, association, adaptation, and pattern recognition capabilities, neural networks are shown to be suitable for solving problems in production planning with uncertain and non-stationary demand. We demonstrate that a properly designed and trained multi-layered perceptron outperforms traditional algorithms for the rolling horizon version of the dynamic lotsizing problem. Formal arguments are supported by numerical experiments.

Keywords: Lotsizing, Multi-Layered Perceptrons, Neural Networks, Pattern Recognition, Production Planning, Uncertainty

1 Introduction

Production planning is a challenging problem area in the field of management sciences. It covers a wide spectrum of decisions in a manufacturing organization. The decisions can be long range, medium range or short range, depending on the length of time from planning to final execution of the decision. Long-range decisions concern the product, processes, plant, equipment and distribution planning. Medium-range decisions involve the master production scheduling (MPS), material requirements planning (MRP) and lotsizing, while short-range decisions deal with sequencing and shop floor control; see [1] for a more detailed discussion.

This paper focuses on the lotsizing problems, which are at the low end of the second level, i.e. the medium-range decisions with a planning horizon of 6–12 months. Lotsizing problems have been extensively studied in the past and there is a wealth of information available from the literature; see [1] for an overview. In a lotsizing problem one is asked to find the lotsizes that satisfy demands for the current planning interval at minimum production and holding costs. Lotsizing problems can be classified by the presence of one or more levels of production, one or more items that have to be produced, continuous or discrete replenishment points and constrained or unconstrained resources. A more formal problem formulation is given in Section 2.

Apart from a number of well-solvable special cases, the lot-sizing problems have been shown to be hard to solve: if there is more than one level of production, more than one item to be produced or there are constrained resources the problems is NP-hard; see [3, 7, 11]. Consequently, a large number of approximation algorithms has been proposed; see [1, 8, 11] and the references therein. In addition to the computational complexity of lotsizing there is a more fundamental problem, namely the uncertainty of its environment. Especially uncertainty in demand forecasts can be of great influence on the quality of the obtained solutions, but also uncertainty in prices and resource constraints, unreliable vendors, scrap losses and inventory record errors may affect the applicability of complex models in real life situations. Roughly speaking, one distinguishes between the following approaches to deal with uncertainty in demand.

The first approach is to assume that the demand is perfectly known for some short time horizon and nothing is known about the demand beyond that horizon. One then applies an optimal or approximation algorithm to find a production schedule for this short time horizon. The production schedule for the complete planning interval, which extends over a number of short time horizons, is then found by iteratively using the above procedure. After each decision the time horizon is shifted forward in time over the span of that decision until the end of the planning interval is reached; see [4, 9] for a discussion of the results. The advantages of this approach are its simple implementation and the absence of history requirements, which makes it suitable for use in practice. The disadvantages are a low performance and high system nervousness, especially if variances in demand are large and/or the demand process contains some periodic pattern.

The second approach assumes that the demand is the realization of a random process, possibly with unknown parameter values. The unknown parameters may characterize the noise-part of the demand process or some systematic trend. By using some estimation procedure the unknown parameters are computed from historical data of the demand process. Once these parameters are known they can be incorporated in lotsizing algorithms that reckon with future demand. Estimation and optimization may also be done simultaneously by viewing the problem as a Markov decision problem; see [5]. The results of this approach are mathematically attractive but practically of little use, due to the rather strict structure which is required for the demand process and the relatively complicated estimation and optimization procedures required. For that reason the gap between theory and practice is substantial.

Our hope of finding a good lotsizing method is based on the idea of using a kind of hybrid approach that exploits the positive aspects of both the approaches given above. Reports on such hybrid approaches are rare in literature; see [5]. To be successful such an approach should be able to

- solve a finite horizon lotsizing problem,
- forecast future demand based on demand history (when available) and account for these future demands in short term decisions,

- recognize patterns in the demand process if present, and
- adapt to changes in the demand process structure.

Neural networks and more specifically multi-layered perceptrons are serious candidates for handling these tasks effectively. This can be substantiated with the following arguments.

- Multi-layered perceptrons can solve combinatorial optimization problems. This capability is based on the classification capabilities of multi-layered perceptrons; see [22]. Since any finite lotsizing problem can be formulated as a combinatorial optimization problem, multi-layered perceptrons are capable of solving finite horizon lotsizing problems.
- The capabilities of multi-layered perceptrons for predicting the future behavior of time series have been known for some time; see for instance [17]. Recently, their capabilities for predicting real-world time series are convincingly demonstrated in [20].
- Pattern recognition is commonly accepted as one of the prime application fields of neural networks. There exists a vast amount of literature on this subject; see for instance [18].
- The adaptation capabilities of neural networks are accounted for by their (re-)learning capabilities. Most papers in literature discuss the learning of a static input-output behavior for neural network; see [19]. Dynamic input-output behavior can be obtained by the use of relearning techniques similar to those used for speech recognition.

For these reasons neural networks may be successful in solving problems in production planning with uncertain and non-stationary demand. In this paper we investigate this hypothesis by designing a multi-layered perceptron for the rolling horizon version of the dynamic lotsizing problem and comparing the results with the results of some traditional algorithms. Although we restrict ourselves to the use of multi-layered perceptrons, Hopfield networks may be used for solving problems in production planning as well; see also [14].

The remainder of the paper is organized as follows. In Section 2 we present the dynamic lotsizing problem, give a mathematical formulation and discuss some of the standard techniques for solving both the finite and rolling horizon case. In Section 3 the problem is presented in a 0–1-formulation. Furthermore, we discuss the network configurations needed to solve the problem with a three-layered perceptron or two-layered perceptron, again for both the finite and rolling horizon case. In Section 4 we compare the results of a two-layered perceptron that is obtained with the backpropagation learning algorithm and the traditional methods, when applied to the problem with different demand structures. The paper ends with some concluding remarks and references.

2 Problem Formulation and Traditional Algorithms

In this section we describe the dynamic lotsizing problem (DLP) firstly stated by Wagner and Whitin ([13]). It is a lotsizing problem with one level of production, one item that has to be produced, discrete replenishment points and unconstrained resources. We have selected this problem for testing the neural net approach since:

(*i*) it is a classical problem that, although it has been studied extensively, still attains much attention in literature,

(*ii*) it often appears as a subproblem in more complex lotsizing problems,

(*iii*) its solution techniques are used as heuristics for other lotsizing problems, and

(*iv*) the problem lends itself to a neural implementation (see Section 3).

We will consider two versions of the problem, the finite horizon case and the rolling horizon case (cf. [8, 9]). The difference between these two cases is that in the finite horizon case the demand is assumed to be known for the complete planning interval, whereas in the rolling horizon case at every replenishment point the future demand is known for only a fixed number of periods —the horizon— which is strictly less than the length of the planning interval.

2.1 The finite horizon case of the dynamic lotsizing problem

The finite horizon case of the DLP is the problem of satisfying at minimum cost the known demands for a specific commodity in a number of consecutive periods. It is possible to store units of this commodity to satisfy demands in later periods, but backlogging is not allowed. The costs consist of production and holding costs. The production costs consist of two components: a cost per unit produced (marginal production cost) and a fixed setup cost that is incurred whenever production occurs in that particular period. The holding costs are linear in the inventory level at the end of the period. Both the inventory at the beginning and at the end of the planning horizon are assumed to be zero.

Let n be the length of the planning horizon and let d_i, p_i, r_i and h_i denote demand, marginal production cost, setup cost and unit holding cost in period i, $i = 1, \ldots, n$, respectively. Then using the variables

$$
\begin{array}{lcl}
u_i & : & \text{number of units produced in period } i \\
s_i & : & \text{number of units in stock at the end of period } i \\
y_i & : & \begin{cases} 0 & \text{if a setup occurs in period } i \\ 1 & \text{otherwise}^1 \end{cases}
\end{array}
$$

[1] We use the reverse definition compared to the usual, to facilitate the 0–1-formulation given in Section 3.

the problem can be formulated as

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}(p_i u_i + r_i(1-y_i) + h_i s_i) \\ \text{s.t.} \quad & \\ & s_{i-1} + u_i - d_i = s_i && i = 1,\ldots,n \\ & s_0 = s_n = 0 \\ & u_i \leq (1-y_i)\textstyle\sum_{j=i}^{n} d_j && i = 1,\ldots,n \\ & u_i \geq 0,\ s_i \geq 0,\ y_i \in \{0,1\} && i = 1,\ldots,n \end{aligned} \tag{1}$$

The above formulation is essentially a mixed-integer linear program formulation, but the problem is not hard to solve (see [6] for an efficient $\mathcal{O}(n^2)$ implementation and [12] for an $\mathcal{O}(n \log n)$ implementation). The problem becomes much harder if some kind of resource constraints like $u_i \leq c_i$ $(i = 1,\ldots,n)$ are added (see [3, 7, 11] for an analysis of the complexity of the problem in the various cases).

We will focus in this paper on the special case of the above problem that was also treated by Wagner and Whitin (see [13]). They assumed identical marginal production costs $(p_i = p)$ and non-negative unit holding costs $(h_i \geq 0)$. For simplicity we will additionally assume that setup costs and holding costs are also identical $(r_i = r,\ h_i = h)$. In this special case the problem can be solved in linear time (see [12]), but we do not consider this result. Instead, we examine some of the properties originally stated by Wagner and Whitin, since these will enable us to obtain the 0–1-formulation discussed in the following section.

The first property is that $\sum_{i=1}^{n} u_i = \sum_{i=1}^{n} d_i$, which follows by summing the first constraint of (1) and using that $s_0 = s_n = 0$. Substitution in the object function then shows (using $p_i = p$) that the variables u_i can be eliminated from the formulation. This yields

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}(r(1-y_i) + h s_i) \\ \text{s.t.} \quad & \\ & 0 \leq s_i + d_i - s_{i-1} \leq (1-y_i)\textstyle\sum_{j=i}^{n} d_j && i = 1,\ldots,n \\ & s_0 = s_n = 0 \\ & s_i \geq 0,\ y_i \in \{0,1\} && i = 1,\ldots,n \end{aligned} \tag{2}$$

The next property proven by Wagner and Whitin is that in an optimal solution of (2) a setup occurs in a period only if the inventory level at the end of the previous period is zero: $(1-y_i)s_{i-1} = 0$. This implies that in every period $s_i = \sum_{j=i+1}^{k} d_j$ for some $k \geq i$, which is the key observation to obtain a dynamic programming algorithm for (2). Wagner and Whitin showed that this algorithm solves the problem in $\mathcal{O}(n^2)$ time (see [13]).

2.2 The rolling horizon case of the dynamic lotsizing problem

In the rolling horizon case of the DLP one is faced with a planning interval in which there is demand for a specific commodity, that extends over a number of periods (say N periods).

However, at any moment demand is known (visible) for only a fixed and limited number of consecutive periods (say $n \ll N$) extending in the future, which defines a finite horizon. Beyond the horizon nothing is known about the (invisible) demand (see Figure 1).

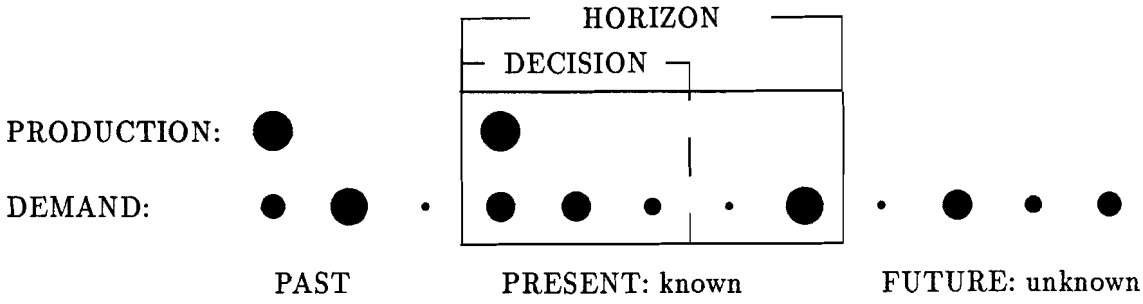

Figure 1: The interpretation of a rolling horizon, $N = 12$ and $n = 5$.

After each decision about the next production lot the horizon is shifted forward in time the same number of periods as satisfied by that lot. This method is applied repeatedly until the end of the planning interval is reached. The objective is to find a strategy for determing the lotsizes such that the total production and holding costs are minimal for the complete planning interval. An obvious approach is to calculate at any decision moment the optimal lotsizes for the encountered finite horizon problem using one of the algorithms described in the previous section and implement the first lotsize. However, this might yield a production schedule for the complete planning interval that is far from optimal, due to horizon effects.

Instead of using the somewhat complicated —and not necessarily optimal— algorithms designed for the finite horizon case, people have looked for simpler rules for determining lotsizes in the rolling horizon case that perform at least as good. One such rule is the Silver and Meal heuristic, which incorporates demand from future periods in the current lotsize until further demand additions would cause the average cost per period to increase; see [10]. It can be shown that this heuristic has a worst case error bound which can be arbitrarily bad when applied to the finite case problem described in the previous section, see [2] and the references therein. Nevertheless, the heuristic has been shown to perform well in practical situations and computational studies when applied to the rolling horizon case; see for instance [4].

In this paper we use the Silver and Meal heuristic in two ways. Firstly, one can easily show that it can be implemented using a one-layered perceptron. This implies that a well-designed two-layered perceptron should have no difficulty in beating the Silver and Meal heuristic, see also the next section. Secondly, we use the Silver and Meal heuristic for comparison with the numerical results in Section 4.

3 Neural Modeling

In this section we discuss the solving of the DLP introduced in the previous section by a multi-layered perceptron. We start by showing in Section 3.1 that the finite horizon case can be put in a 0–1-formulation. A 0–1-formulation is necessary since we want to solve the problem with a multi-layered perceptron. In Section 3.2 we investigate the possibility of exactly solving the finite case with a three- and two-layered perceptron, respectively. Finally, in Section 3.3 we discuss the configuration needed for solving the rolling case of the DLP.

3.1 A 0–1-Formulation

In the previous section we mentioned the zero-inventory property ($(1-y_i)s_{i-1}=0$) of an optimal solution of the problem given by (2), which implies that $s_i = \sum_{j=i+1}^{k} d_j$ for some $k \geq i$. A more detailed analysis shows that in fact we have:

$$s_i = \sum_{j=i+1}^{n} y_{i+1}y_{i+2}\cdots y_j d_j. \tag{3}$$

One can easily verify that (3) satisfies the constraints of (2) provided that $y_1 = 0$. Substitution of (3) in the object function of (2) yields:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\left(r(1-y_i) + hd_i\sum_{j=1}^{i} y_j y_{j+1}\cdots y_i\right) \\ \text{s.t.} \quad & \\ & y_1 = 0 \\ & y_i \in \{0,1\} \qquad\qquad i = 2,\ldots,n \end{aligned} \tag{4}$$

Finally, we write the above problem in the form introduced in [22]. Every combinatorial optimization problem can be formalized as a tuple (I,S,F,c), where I denotes a set of problem instances and for every $x \in I$ the set $S(x)$ 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.

Denote $x_i = hd_i/r$, eliminate y_1, renumber the remaining variables from 1 to $n-1$ instead of 2 to n and replace n by $n+1$ then we obtain the tuple (I,S,F,c) given by

$$\begin{aligned} I &= \{x \in \mathbb{R}^n \mid x_i \geq 0\}, \\ S &= \{0,1\}^n, \\ F &= \{0,1\}^n, \\ c(y;x) &= \sum_{i=1}^{n}\sum_{j=1}^{i} y_j y_{j+1}\cdots y_i x_i - \sum_{i=1}^{n} y_i, \end{aligned} \tag{5}$$

where for a given $x \in I$ the problem is to find a $y \in F$ that minimizes $c(y;x)$. In the next section we investigate whether the problem given by (I, S, F, c) can be solved by an multi-layered perceptron.

3.2 A Multi-Layered Perceptron For The Finite Horizon Case

In the previous section we showed that the finite case of the DLP can be put in a 0–1-formulation. By noting that the cost function c given by (5) is linear in x it follows that the necessary conditions given in [22] are fulfilled, which implies that there exists a three-layered perceptron that exactly solves the problem. Furthermore, by the general construction method of [22] an exact three-layered perceptron can be found straightforwardly. Unfortunately, this three-layered perceptron has an exponential number of hidden units. In [22] it is also shown that an efficient multi-layered perceptron only exists if the problem is in POLYLOGSPACE. From the dynamic programming solution of Wagner and Whitin we know that the DLP can be reduced to the shortest path problem, which is in POLYLOGSPACE (see for instance [15]). This implies that the DLP is in POLYLOGSPACE and, hence, there is some hope of finding an efficient multi-layered perceptron that exactly solves the DLP.

Our search for an efficient multi-layered perceptron concentrated on finding an efficient exact two-layered perceptron. The reason for our faith in the existence of an exact two-layered is based on the results for small values of n. In these cases an exact two-layered perceptron that solves the DLP can be easily found by hand; see [21]. Furthermore, it can be shown that classification problem that corresponds with the DLP as defined in [22] satisfies the necessary conditions for the existence of an exact two-layered perceptron given in [23], see [21]. This has lead to the following conjecture.

Conjecture 1 *There exists a two-layered perceptron that exactly solves the DLP.*

The preliminary results given in [21] indicate that an exact two-layered perceptron for the DLP given by 5 requires $\mathcal{O}(n)$ hidden units. We are currently investigating whether the algorithms for the calculation of an exact two-layered perceptron given in [23] can be used to find an exact two-layered perceptron for the DLP.

In Section 4 we present the results of using a slightly modified backpropagation algorithm to find a two-layered perceptron for the DLP. The learning vectors required for the backpropagation algorithm can be obtained for instance by assuming demand is uniformly distributed on $[0, 1]$ or by using some data from the past, see also Section 3.3 and Section 4.1.

3.3 A Multi-Layered Perceptron For The Rolling Horizon Case

For the rolling horizon case one can of course use a multi-layered perceptron that is designed to solve the finite case in the same repeated manner as one can use for instance the Wagner and Whitin algorithm, see Section 2.2. However, this approach only uses part of the capabilities of multi-layered perceptrons. To exploit the associative, adaptive and pattern recognition capabilities of multi-layered perceptrons we set the following objective.

The output of the multi-layered perceptron with (say) n inputs and n outputs should be the first lotsize of the optimal schedule for the finite DLP over $n + k$ periods, where k is some integer possibly depending on n.

Thus, instead of giving the optimal first lotsize for the n period DLP, the multi-layered perceptron must 'predict' the demand in period $n + 1, n + 2, \ldots, n + k$ and use these predictions to calculate the optimal first lotsize for the $n + k$ period DLP. A multi-layered perceptron that fulfills this task can be found using the backpropagation algorithm. In the learning process the network is offered as an input the first n values of an $n+k$ dimensional demand vector and the corresponding output consists of the first n values of the optimal $n + k$ vector corresponding to the demand vector. Obviously, the quality of the obtained results depends on the value of k. In some cases there exists a value of k for which the results are optimal; see Section 4.2.

Based on the results found for the finite case we have used a two-layered perceptron for the rolling case of the DLP as well. The main reason is that we expect that the finite case can be solved by a two-layered perceptron. Furthermore, since the Silver and Meal heuristic can be implemented in a one-layered perceptron, the results of a two-layered perceptron should be at least as good as this heuristic. Finally, a three-layered perceptron requires excessive training times. The number of hidden units was taken to be $\mathcal{O}(k)$, with k the integer discussed above.

It remains to determine the learning vectors. If there is some knowledge about the demand structure it can be used to simulate demand vectors. If there is some history about the demand it can be used to construct demand vectors. Sometimes a combination of these methods is possible. Of course the amount of information determines the value of k that can be taken. In the following section we discuss some results obtained by this approach and see how well the association, adaptation and pattern recognition capabilities perform.

4 Numerical Results

In this section we present a summary of the results that were obtained by training a two-layered perceptron to solve the DLP in both the finite horizon and the rolling horizon case, more numerical results can be found in [16]. To this end we used the standard backpropagation algorithm presented in [19] with a small extension. After the usual learning phase in which we used the backpropagation algorithm with the learning speed

set to 0.4 and the coefficient in the momentum term set to 0.7 we applied a second phase: the adjustment phase. In the adjustment phase we use again the backpropagation algorithm but only backpropagate an error down the network if the output of the network is not the desired output after rounding to 0 or 1. In the adjustment phase the learning speed is set to 0.01.

The motivation for this second phase is the observation that after the learning phase 80 to 90% of the learning vectors was treated correctly but there remained a group of specific inputs for which the network did not yield the correct output. The idea of the adjustment phase is that for the learning vectors in the bad group the weight changes are in the same direction and for the rest of the learning vectors these weight changes cancel each other out.

We compare the results for 500, 800 and 1200 learning vectors. The length of the learning phase was taken to be $30000/l$ cycles, where l is the number of learning vectors. In each cycle every learning vector is considered once. The adjustment phase consisted of $20000/l$ cycles.

The performance of a network is measured in three manners:

(i) The learning vectors: All the learning vectors used during learning of that network are given to the network once. The percentage of inputs for which the network provides the desired output is shown.

(ii) The test vectors: The same as with the learning vectors but now with a thousand arbitrary vectors. These are formed in the same way as the learning vectors but were not used during the training of the networks.

(iii) The Cost: In this test a schedule for the entire planning interval is determined by the network for all test vectors. The neural network is shifted repeatedly over the demands by the amount determined by the first lotsize given by the network, see also Section 2.2. In case of the finite horizon case the first n demands are padded with zeros until the end of the planning interval is reached. The cost of the final schedule found is compared with the optimal cost for the complete planning interval that can be calculated using the Wagner and Whitin algorithm.

4.1 The finite horizon case

We consider a 10 period finite horizon DLP with demand x_i random and uniformly in $[0.05, 0.90]$. We used a 10-15-10 two-layered network, but the size of the hidden layer is not very critical, 10-10-10 and 10-20-10 networks have comparable results.

In figure 2, 3 and 4 the results are given for the 10-15-10 network. The left diagram in these figures represents the results after the learning phase, the right diagram after the adjustment phase. The networks are developed with three different realizations of the

learning vectors. With each realization there are networks formed with 500, 800 and 1200 learning vectors, presented in the left, the middle and the right bar, respectively. In every bar the optimal result and average result of 10 networks formed with that parameter setting and realization of the learning vectors is shown. There are 10 networks formed because of the random initialization of the weights in the network.

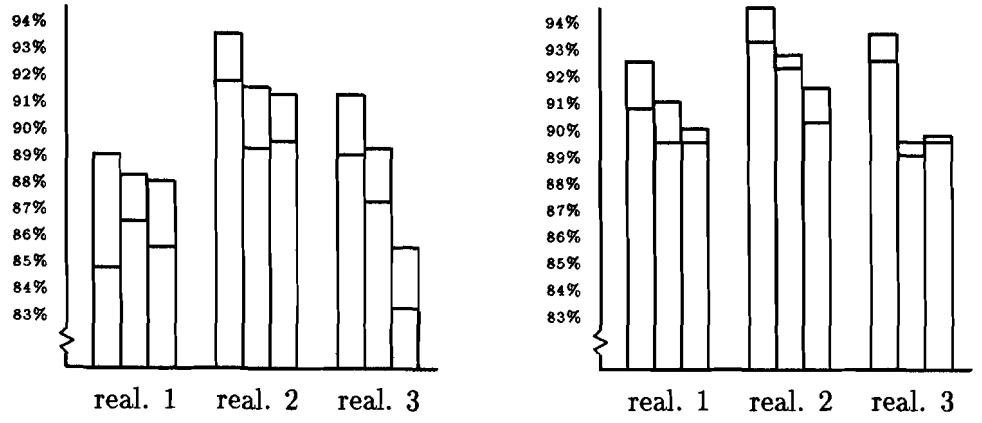

Figure 2: The learning vectors results.

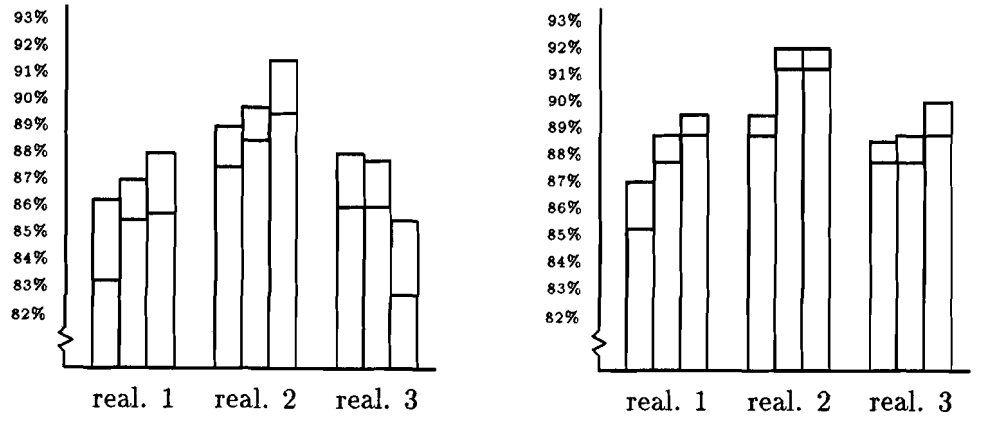

Figure 3: The test vectors results.

If we compare the results of the networks before and after adjustment we conclude that the results on the learning and test vectors improve and that the variance in the results becomes less. The cost results not necessarily improve because better results on the learning vectors not has to imply that costs go down. When the results after adjustment in figure 2 and 3 are compared, we see that certainly with 800 and 1200 learning vectors the networks generalize very good. Furthermore, the results with the learning vectors and the test vectors are almost equal.

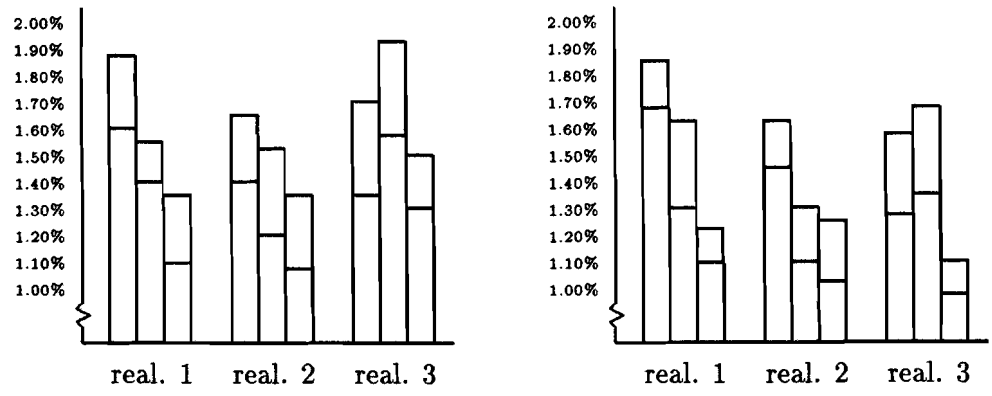

Figure 4: The cost results.

4.2 The rolling horizon case

The second case that will be investigated is the rolling horizon case. In this case the learning takes place with data from the past. The networks are tested with a simulation from the future. In the test the networks have to produce a schedule for a planning interval of thousand periods with demand data that has not been used during learning. The horizon has length 5. For the schedule formed in this way the cost is calculated and compared to the optimal cost obtained by the Wagner and Whitin algorithm applied to the 1000 period planning interval. We use a 5-10-5 two-layered perceptron.

For the rolling horizon case we will also investigate situations with a pattern in demand, but first the demand in every period is chosen random between 0.05 and 0.90. The performance of Wagner and Whitin with horizon is shown in Table 1. With horizon 10, Wagner and Whitin performs optimal for the rolling horizon case of the 1000 periods problem.

Horizon length						
4	5	6	7	8	9	10
1.40	0.55	0.34	0.08	0.09	0.04	0.00

Table 1: Performance of Wagner and Whitin with horizon.

Therefore the desired output in the learning vectors is formed with the demand in 10 periods ($n = 5$ and $k = 5$). In this way also the desired output is optimal for the rolling horizon case of the 1000 periods problem. The results of the 5-10-5 networks are shown in figure 5

After adjustment the results are clearly getting better. The variance in the performance is much lower. After adjustment almost all networks perform at the same high level. Only the networks in realization 2 developed with 500 learning vectors perform not that good.

Next we investigate two situations with a cyclic demand pattern. When the cycle has

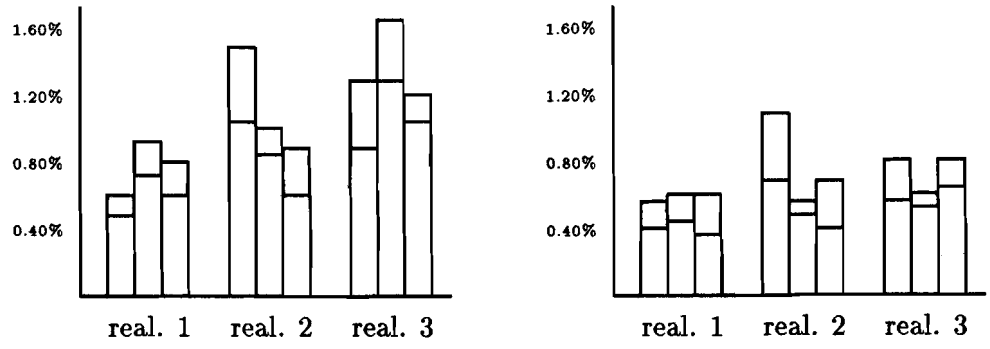

Figure 5: The cost results in the absence of demand pattern.

length t, then demand in the i'th period of that cycle is defined as X_i $(1 \leq i \leq t)$.

First a situation with cycle length 3 will be examined. The pattern is defined by: $X_1 \sim \mathcal{U}(0.05, 0.40)$, $X_2 \sim \mathcal{U}(0.25, 0.65)$ and $X_3 \sim \mathcal{U}(0.50, 0.90)$. Here $\mathcal{U}(A, B)$ denotes the uniform distribution on the interval (A, B). With horizon length 10 the cost of Wagner and Whitin is 0.03%. Again the desired output in the learning vectors is formed with 10 periods. The results of the 5-10-5 networks are shown in figure 6

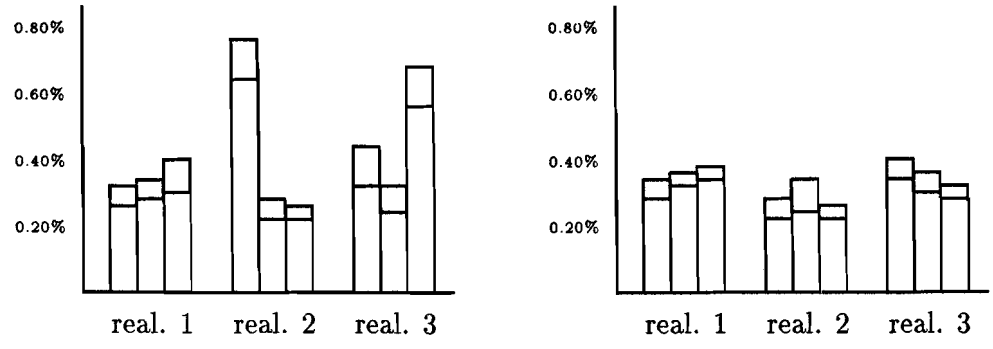

Figure 6: The cost results in the presence of a demand pattern with cycle length 3.

Again the results after adjustment are much better than before. The number of learning vectors seems to have no influence on the results, so perhaps it is possible to use less learning vectors. Especially in situations in which there is not much data available from the past this can be important. With horizon length 5 the cost of Wagner and Whitin is 0.93% and the cost of Silver and Meal is 5.50% higher than the optimal cost.

In the last situation that will be investigated the cycle length is 6 and the pattern is defined by: $X_1, X_2 \sim \mathcal{U}(0.02, 0.15)$, $X_3, X_4 \sim \mathcal{U}(0.09, 0.24)$ and $X_5, X_6 \sim \mathcal{U}(0.18, 0.33)$.

We have chosen the above values in such a way that the average production length is longer than in the previous case. Because the horizon length stays the same the Wagner and Whitin algorithm will perform less. This is because the demand in the periods beyond the horizon are now more important for producing a good scheme. This is also the case for the network but because of their prediction capabilities it should have less effect on

their performance.

The results of the networks developed with learning vectors where the desired output is formed with 12 periods are shown in figure 7

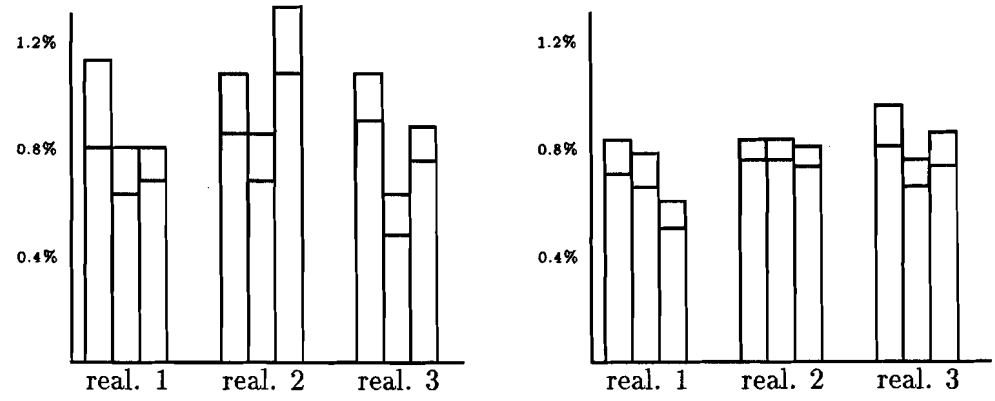

Figure 7: The cost results in the presence of a demand pattern with cycle length 6.

Again in general the results get better after adjustment. In some individual cases this is not true. The cause is not that during adjustment the amount of learning vectors unlearned is bigger than the amount of learning vectors learned. In fact, also in these cases the networks treat more learning vectors good after adjustment than before. The goal during learning is to form networks that treat as much as possible learning vectors good. However, when a certain network treats more learning vectors good than another network, this does not automatically mean that the cost of the scheme generated with that network are lower than the cost of the other network scheme. Not all the mistakes are as important. Some will cause low extra cost and others will cause high extra cost.

The cost of Silver and Meal are 2.47% and the cost of Wagner and Whitin are 4.71% higher than optimal. In [16] more demand patterns are investigated. The performance of Silver and Meal is not always as bad as in the situation examined here. But there are more situations in which Silver and Meal and/or Wagner and Whitin perform badly. On the other hand, there is not a situation in which the networks perform badly.

5 Concluding Remarks

We addressed the possibility of solving problems in production planning using neural networks. We focussed on the dynamic lotsizing problem (DLP) and showed how this problem can be treated using multi-layered perceptrons. The existence of a three-layered perceptron that exactly solves the finite horizon case of the DLP was demonstrated and it was conjectured that an exact two-layered perceptron exists for this problem too. However, the classification, association, adaptation and pattern recognition capabilities of multi-layered perceptrons show to full advantage in the rolling horizon case of the DLP where

there is uncertainty in demand. In this case a properly trained two-layered perceptron outperforms traditional algorithms like the Wagner and Whitin algorithm and the Silver and Meal heuristic. Especially the robustness of the neural net approach with respect to the demand structure is noteworthy. A drawback of the use of neural networks is the large amount of time required for the training.

We end this paper by discussing some possible future extensions. With respect to the DLP one can think of experimenting with other demand structures. Also changing demand structures which were not discussed in this paper can be considered to test the adaptation capabilities of the neural net approach. Experiments with different learning algorithms in combination with a lower number of training vectors might speed up the training phase. Finally, one can try the neural net approach to other and more complex problems in the field of production planning.

References

[1] H.C. BAHL, L.P. RITZMAN AND J.N.D. GUPTA, Determining Lot Sizes and Resource Requirements: A Review, *Operations Research,* 35, 3, 329–345, 1987.

[2] G.R. BITRAN, T.L. MAGNANTI AND H.H. YANASSE, Approximation Methods for the Uncapacitated Dynamic Lot Size Problem, *Management Science,* 30, 9, 1121–1140, 1984.

[3] G.R. BITRAN AND H.H. YANASSE, Computational Complexity of the Capacitated Lot Size Problem, *Management Science,* 28, 10, 1174–1186, 1982.

[4] J.D. BLACKBURN AND R.A. MILLEN, Heuristic Lot-Sizing Performance In A Rolling-Schedule Environment, *Decision Sciences,* 11, 4, 691–701, 1980.

[5] N.P. DELLAERT, *Production To Order,* Springer Lecture Notes on Economics and Mathematical Systems 333, Springer Verlag, Berlin, 1989.

[6] J.R. EVANS, An Efficient Implementation of the Wagner-Whitin Algorithm for Dynamic Lot-Sizing, *Journal of Operations Management,* 5, 229–235, 1985.

[7] M. FLORIAN, J.K. LENSTRA AND A.H.G. RINNOOY KAN, Deterministic Production Planning: Algorithms and Complexity, *Management Science,* 26, 7, 669–679, 1980.

[8] J. MAES AND L.N. VAN WASSENHOVE, Multi Item Single Level Capacitated Dynamic Lotsizing Heuristics: A Computational Comparison (Part I: Static Case), *IIE Transactions,* 114–123, 1986.

[9] J. MAES AND L.N. VAN WASSENHOVE, Multi Item Single Level Capacitated Dynamic Lotsizing Heuristics: A Computational Comparison (Part II: Rolling Horizon), *IIE Transactions,* 124–129, 1986.

[10] E.A. SILVER AND J.C. MEAL, A Heuristic for selecting lot size quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment, *Production and Inventory Management,* 14, 2, 64–74, 1973.

[11] M. SALOMON, *Deterministic Lotsizing Models for Production Planning,* Phd. Thesis, Erasmus Univ. Rotterdam, 1990.

[12] A. WAGELMANS, S. VAN HOESEL AND A. KOLEN, Economic Lot-Sizing: An $\mathcal{O}(n \log n)$-Algorithm that runs in linear time in the Wagner-Whitin case, *CORE Discussion Paper N° 8922,* CORE, 1989.

[13] H.M. WAGNER AND T.M. WHITIN, Dynamic Version of the Economic Lot Size Model, *Management Science,* 5, 1, 89–96, 1958.

[14] C. CHIU, C.Y. MAA AND M.A. SHANBLATT, An Artificial Neural Network Algorithm For Dynamic Programming, *International Journal of Neural Systems,* 1, 3, 211–220, 1990.

[15] G.A.P. KINDERVATER, *Exercises in Parallel Combinatorial Computing,* Phd. Thesis, Erasmus Univ. Rotterdam, 1989.

[16] M.J.A.L. VAN KRAAIJ, *The Use of Neural Networks for Lotsizing,* Master Thesis (in Dutch), Eindhoven Univ. of Techn., 1991.

[17] A. LAPEDES AND R. FARBER, How Neural Nets Work, in *(D.Z. Anderson, ed.) Neural Information Processing Systems,* 442–456, 1987.

[18] Y.H. PAO, *Adaptive Pattern Recognition and Neural Networks,* Addison-Wesley, 1989.

[19] D.E. RUMELHART, G.E. HINTON AND R.J. WILLIAMS, Learning Internal Representations by Error Propagation, in *(D.E. Rumelhart and J.L. McClelland, Eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations,* MIT Press, 318–362, 1986.

[20] A.S. WEIGEND, B.A. HUBERMAN AND D.E. RUMELHART, Predicting The Future: A Connectionist Approach, *International Journal of Neural Systems,* 1, 3, 193–200, 1990.

[21] P.J. ZWIETERING, Private Communication, 1991.

[22] P.J. ZWIETERING, E.H.L. AARTS AND J. WESSELS, The Design and Complexity of Exact Multi-Layered Perceptrons, *Memorandum COSOR 91-08, Eindhoven Univ. of Techn.,* 1991.

[23] P.J. ZWIETERING, E.H.L. AARTS AND J. WESSELS, The Classification Capabilities Of Exact Two-Layered Perceptrons, *Memorandum COSOR 91-09, Eindhoven Univ. of Techn.,* 1991.

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science
PROBABILITY THEORY, STATISTICS, OPERATIONS RESEARCH
AND SYSTEMS THEORY
P.O. Box 513
5600 MB Eindhoven, The Netherlands

Secretariate: Dommelbuilding 0.03
Telephone : 040-473130

-List of COSOR-memoranda - 1991

Number	Month	Author	Title
91-01	January	M.W.I. van Kraaij W.Z. Venema J. Wessels	The construction of a strategy for manpower planning problems.
91-02	January	M.W.I. van Kraaij W.Z. Venema J. Wessels	Support for problem formulation and evaluation in manpower planning problems.
91-03	January	M.W.P. Savelsbergh	The vehicle routing problem with time windows: minimizing route duration.
91-04	January	M.W.I. van Kraaij	Some considerations concerning the problem interpreter of the new manpower planning system formasy.
91-05	February	G.L. Nemhauser M.W.P. Savelsbergh	A cutting plane algorithm for the single machine scheduling problem with release times.
91-06	March	R.J.G. Wilms	Properties of Fourier-Stieltjes sequences of distribution with support in [0,1).
91-07	March	F. Coolen R. Dekker A. Smit	Analysis of a two-phase inspection model with competing risks.
91-08	April	P.J. Zwietering E.H.L. Aarts J. Wessels	The Design and Complexity of Exact Multi-Layered Perceptrons.
91-09	May	P.J. Zwietering E.H.L. Aarts J. Wessels	The Classification Capabilities of Exact Two-Layered Peceptrons.
91-10	May	P.J. Zwietering E.H.L. Aarts J. Wessels	Sorting With A Neural Net.
91-11	May	F. Coolen	On some misconceptions about subjective probability and Bayesian inference.

COSOR-MEMORANDA (2)

91-12	May	P. van der Laan	Two-stage selection procedures with attention to screening.
91-13	May	I.J.B.F. Adan G.J. van Houtum J. Wessels W.H.M. Zijm	A compensation procedure for multiprogramming queues.
91-14	June	J. Korst E. Aarts J.K. Lenstra J. Wessels	Periodic assignment and graph colouring.
91-15	July	P.J. Zwietering M.J.A.L. van Kraaij E.H.L. Aarts J. Wessels	Neural Networks and Production Planning