International Journal of Production Research,
Vol. 46, No. 21, 1 November 2008, 6009–6027

Taylor & Francis
Taylor & Francis Group

Using ensemble and metaheuristics learning principles with artificial neural networks to improve due date prediction performance

RAHUL J. PATIL*

S.P.Jain Institute of Management and Research, Mumbai, India

(*Revision received October 2006*)

One of the common and important problems in production scheduling is to quote an attractive but attainable due date for an arriving customer order. Among a wide variety of prediction methods proposed to improve due date quotation (DDQ) accuracy, artificial neural networks (ANN) are considered the most effective because of their flexible non-linear and interaction effects modelling capability. In spite of this growing use of ANNs in a DDQ context, ANNs have several intrinsic shortcomings such as instability, bias and variance problems that undermine their accuracy. In this paper, we develop an enhanced ANN-based DDQ model using machine learning, evolutionary and metaheuristics learning concepts. Computational experiments suggest that the proposed model outperforms the conventional ANN-based DDQ method under different shop environments and different training data sizes.

Keywords: Metaheuristics; Neural networks; Due-date assignment; Artificial intelligence; Ensemble learning

1. Introduction

One of the common and important problems in production scheduling is to quote an attractive but attainable due date for an arriving customer order. Accurate DDQ improves delivery speed and on-time delivery performance and helps to build a lean and high velocity supply chain. Furthermore, accurate DDQ minimizes out-of-stock disruptions, minimizes idle labour, reduces plant costs, improves customer delivery promises and hence, reduces tardiness costs and lost customer goodwill. The costs of missing due dates can be extreme. Slotnick and Sobel (2001) report that in the aerospace industry, a tardiness penalty as high as $1 million per day is imposed on subcontractors of aircraft components for tardy deliveries. In addition, accurate DDQ is useful for better management of shop control activities such as order release and order review, as well as lead-time comparisons (Sabuncuoglu and Comlekci 2002). The scope and importance of due date quotation policies is expected to broaden further in the digital and global marketplace because of the shift of manufacturing focus from make-to-stock to make-to-order and assemble-to-order production models where online customers expect reliable due date and price quotes

*Email: rahuljpatil@spjimr.ernet.in

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online © 2008 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540701197036

for their requests (Keskinocak and Tayur 2004). Cheng and Gupta (1989) beautifully articulate the growing importance of the due date quotation problem:

> 'A review of the literature reveals that this aspect of the scheduling decision (determination of optimal due date values) is of particular importance to both researchers and practicing managers. Research in this area obviously has not been carried out to its completeness since it is apparent that so many areas still remain untouched. Practicing managers are increasingly faced with difficult situations, in which jobs have to be delivered on time otherwise costs will be incurred.'

Due date quotation is considered to be a difficult problem because of the random nature of job arrivals, variable routing sequences, and processing characteristics (Hsu and Sha 2004). Traditional due date quotation methods such as total work content and jobs in queue use simple linear regression methods and aggregate job and shop information to estimate lead times and to quote due dates (Ragatz and Mabert 1984). Because of their simplicity, these methods do not consider several complexities involved in the due date quotation such as non-linear and interaction effects (Philipoom *et al*. 1994).

Among a wide variety of prediction methods proposed to improve due date quotation (DDQ) accuracy, artificial neural networks (ANN) are considered the most effective because of their flexible non-linear and interaction effects modelling capability, and consequently have been recommended as a decision support tool for the DDQ (Hsu and Sha 2004). In spite of this growing use of ANNs in a DDQ context, ANNs have several intrinsic shortcomings that undermine their accuracy. First, they suffer from an instability problem (Breiman 1996a). Second, they are prone to over fitting (Sexton *et al*. 2003). Third, they are susceptible to bias and a variance dilemma (German *et al*. 1992). Clearly, the effectiveness of the ANNs-based DDQ can be improved by resolving the above limitations.

In the AI literature, ensemble and evolutionary learning methods have been suggested to solve these problems (Breiman 1996a, Sexton *et al*. 2003). In this paper, we develop an enhanced ANN-based DDQ model using machine learning and metaheuristics learning concepts. Specifically, we use genetic algorithms (GA) to search for neural network architectures that develop a parsimonious model of flow time prediction. We then study the use of two ensemble-learning methods: bootstrap aggregating (bagging); and boosting to minimize bias and variance errors. Important contributions of this research include the following.

1. We propose a DDQ model that improves the accuracy and reliability of the due date quotation decision. Moreover, our model is general in nature, and can be used to improve an ANN's performance for other important managerial problems.
2. From a research methodological perspective, it demonstrates the importance of combining concepts from statistics, artificial intelligence, and metaheuristics literature to develop better prediction models.

The paper is organized as follows: section 2 provides a literature review on the applications of ANNs to the DDQ problem and other business prediction problems;

section 3 describes the proposed ANNs-based DDQ algorithm, and discusses ensemble learning methods; section 4 discusses experimental design, simulation set-up, and computational process; section 5 presents the results of computational experiments; and section 6 reports the conclusions.

2. Literature review

The importance of assigning and meeting due dates is well recognized by both production managers and academic researchers. Sen and Gupta (1984), Cheng and Gupta (1989), Keskinocak and Tayur (2004) provide comprehensive reviews on DDQ models. To put the research contribution of this paper into perspective, we discuss prior research that investigates the use of ANNs for the DDQ problem. In addition, we discuss in brief the applications of ANNs in other business settings to demonstrate a broader application domain for the proposed prediction model.

To incorporate non-linear and interaction effects inherent in lead-time prediction problems, researchers have used ANNs for estimating lead times and assigning due dates. Philipoom *et al.* (1994) are amongst the first researchers to consider the use of ANNs in the DDQ problem. They examined the effectiveness of ANNs compared to the conventional DDQ rules such as total work content, the jobs in queue under different shop structures (\,flow shops, job shops) and different training data sizes. They found that ANNs outperformed conventional rules for the objectives of minimizing standard deviation of lateness and mean absolute deviation in most of the experiments.

In a follow up study, Philipoom *et al.* (1997) used this approach to minimize linear, quadratic and cubic objective functions of earliness and tardiness costs. They compared ANNs against conventional DDQ rules and linear programming-based DDQ solutions. Their result indicates that ANNs are accurate and robust because they provide superior results regardless of the shape of the cost function. In contrast, other methods showed poor performance with certain cost functions thereby throwing their reliability into question.

Hsu and Sha (2004) extended prior work by studying the use of ANNs in a complex job shop under different dispatching rules and order review/release policies. In their simulation experiments, ANNs-based DDQ rules outperformed regression based DDQ rules with objectives of on-time delivery and mean tardiness rates. In a similar study, Sha and Hsu (2004) found ANNs to be superior to regression-based rules in improving DDQ accuracy in a semiconductor manufacturing plant. This stream of research has concluded that ANNs are worthy of further experimentation as a methodology of choice in the DDQ context (Philipoom *et al.* 1994, Hsu and Sha 2004).

ANNs have also been used to solve other manufacturing problems. Huang *et al.* (1999) used ANNs to predict the production performance of a wafer fabrication plant. Udo (1992) provided an excellent review on the application of ANNs in various manufacturing situations. Similarly, ANNs have been used to solve managerial problems from finance and accounting such as predicting bank failures (Tam and Kiang 1992, Sexton *et al.* 2003). The important role of the ANNs methodology for due date prediction as well as for other business prediction and classification problems has motivated us to study the conventional ANNs-based due

date prediction approach, and subsequently to develop a better ANNs-based DDQ model.

3. An enhanced ANNs-based DDQ model

Our conceptual model for due date prediction is shown in figure 1 while section 3.5 describes its algorithmic structure. Our model uses bagging and boosting processes

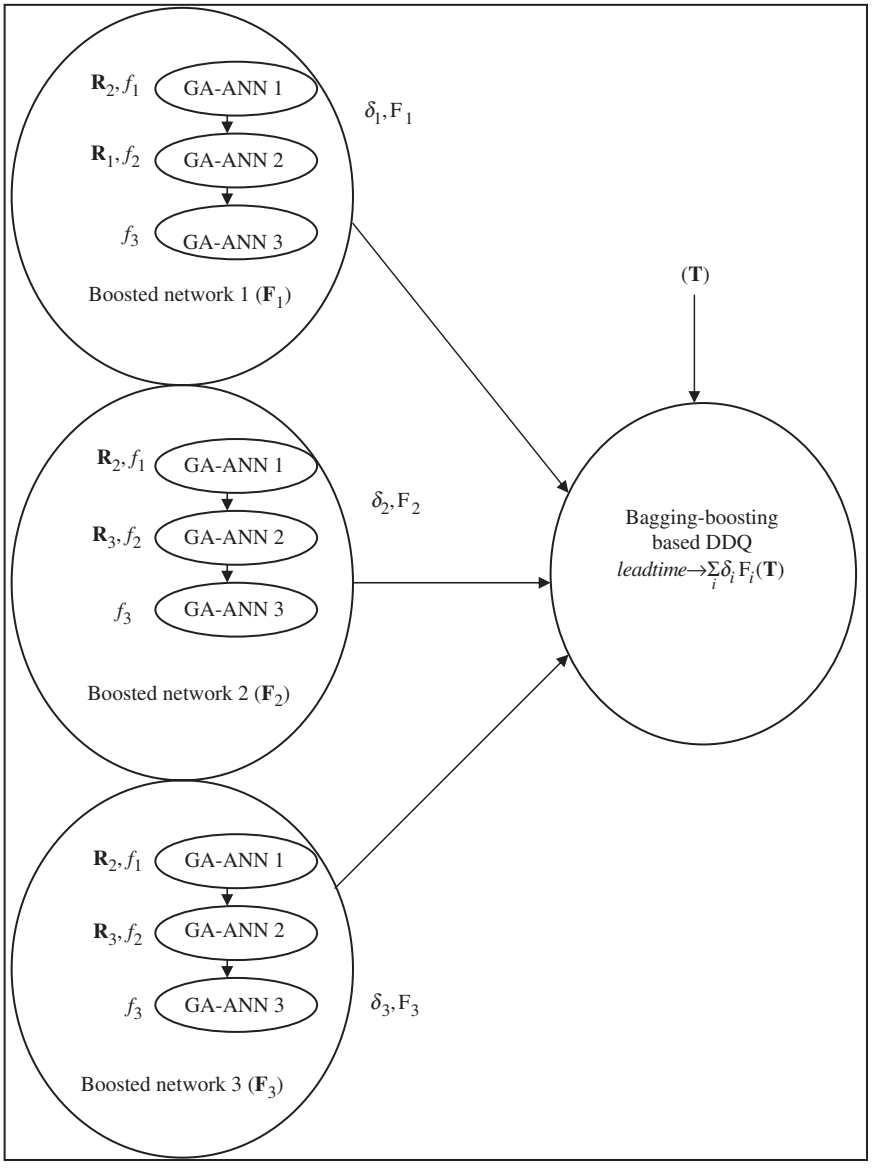

Figure 1. Ensemble and metaheuristics learning based DDQ model.

from ensemble learning literature and diversification and proximate optimality principles from metaheuristics literature to reduce bias and variance error components. The key elements of our prediction model are described in the sections below.

3.1 *Ensemble learning*

Ensemble learning methodology is based on the notion of perturbing and combining (Hansen and Salamon 1990, Breiman 1996b). An ensemble consists of a collection of neural networks and combines their predictions to obtain a final prediction. Our use of ensemble learning methods in the ANNs-based DDQ is motivated by two fundamental reasons. Learning algorithms can be viewed as search methods over hypotheses spaces that identify the best hypothesis in the space. A statistical problem arises when the size of training data is small compared to the hypotheses space. Without sufficient data, it is possible that a learning method produces a set of candidate hypotheses that give the same accuracy on the training data used. For example, different ANN topologies can result in the same accuracy on the training data. By constructing an ensemble from several of these hypotheses, the learning method can average the predictions, and hence reduce the risk of choosing a wrong hypothesis (Dietterich 2000).

The second reason is computational. Many learning algorithms use a local search procedure that may become stuck in local optima. For example, neural network algorithms use a gradient descent procedure to minimize an error function over the training data. Hence, it may be difficult for a learning algorithm to find the best hypothesis. An ensemble constructed by running the search method from different starting points can provide a better approximation to the unknown true hypothesis than using a single hypothesis (Perrone and Cooper 1993).

The decomposition machine learning error into bias and variance components offers a better way of evaluating the quality of a machine learner (German *et al*. 1992). Bias error measures how close the learner produced by a learning algorithm will be to the true function while variance error measures how much the learning algorithm's final models vary with respect to each other for different training sets of the same size (German *et al*. 1992). Bagging and boosting are two ensemble learning methods, which are effective in reducing both bias and variance errors (Friedman 2003). We discuss the rationale for using these methods in the DDQ problem in the next section.

3.2 *Bagging and boosting processes*

Boosting is based on the premise that a single learner lacks the ability to capture complete knowledge regarding complex problem situations, and hence, one should use a series of learners, each capturing some different aspect of a problem, to improve accuracy (Freund and Schapire 1996). To accomplish this objective, boosting sequentially generates a series of neural networks where the training instances that are 'badly' predicted by the previous ANNs play a more important role in the training of latter neural networks (Duffy and Helmbold, 2000, Hastie *et al*. 2001). Thus, boosting tends to correct bias that may exist in predicting complex functions (Kotsiantis and Pintelas 2004).

This principle may be useful in the flow time prediction context for the following reason. The regression surface of a flow time is highly non-linear and interaction-dependent because of the stochastic nature of arrival and service processes, dispatching effects, and the presence of categorical variables such as machine status, and number of operations. While a complex response surface demands a complex function to incorporate all of its intricacies, we focus on the development of a parsimonious neural network model to reduce the variance component of the error. Use of a single ANN is likely to increase the bias error – that is the inability to incorporate all important learning patterns such as interactions and non-linearities. Therefore, extra neural networks are needed to capture this additional information about long/short flow times and to reduce the bias error. We hypothesize that the use of a boosted network of ANNs will provide better accuracy than a single ANN. Duffy and Helmbold (2000) have proposed a boosting algorithm for regression problems. We employ a variant of this algorithm in our prediction model.

Breiman (1996b) introduced the concept of bootstrap aggregating (bagging). Bagging generates replicates of training data sets by sampling with replacement from the original training set and calls the base learner to build hypothesis functions using these bootstrap data sets. It then combines the predictions of the entire hypotheses set to generate a final prediction. The effectiveness of the bagging process depends upon the instability of the prediction method. Breiman (1996b) notes:

> 'The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.'

The bagging process can be useful in our situation because of our use of ANNs as a base learner, which is considered instable (Breiman 1996a), and noisy data set due to the stochastic nature of arrival and service processes. Bagging and boosting have been viewed as separate methods of choice for regression and classification tasks. Kotsiantis and Pintelas (2004), Suen *et al*. (2005) criticize this approach, and provide several reasons for the combined use of bagging and boosting processes. First, bagging tends to reduce the variance component of error (Zhou *et al*. 2002) while boosting tends to reduce the bias component of the error (Breiman 1996b, Zhou *et al*. 2002). Second, boosting is considered stronger than bagging on noise-free data; in contrast bagging tends to perform well on noisy data. Hence, we hypothesize that the use of a bagging process with boosted networks will generate better predictions than the boosted network alone since the combination will correct both bias and variance errors.

3.3 *Metaheuristics principles*

Effectiveness of the bagging method depends upon the diversity of bootstrap samples used for building individual learners in an ensemble. Experiments suggest that the higher the diversity among bootstrap samples, the better the final prediction will be (Hastie *et al*. 2001). Bagging uses a random process to generate a collection of diversified bootstrap samples. However, the metaheuristics literature suggests that simple diversification principles can be more useful than randomization because of

their ability to systematically make use of historical information (Glover and Laguna 1997).

Memory can be used to adaptively construct a pool of diversified bootstraps. We therefore propose a memory-based diversification process to select training data points for individual neural networks. The idea is to keep track of the frequency of occurrence of every past training point and prevent the inclusion of previously selected points by using a penalty, which is a function of their frequency of occurrences. This process can build a collection of diversified bootstraps at each iteration and the penalty term deters the inclusion of the training patterns selected in previous bootstraps.

We hypothesize that it is important to identify good learners in the initial stages of the boosting process to build a good quality boosted network. This hypothesis is motivated by the proximate optimality principle suggested in tabu search, which stipulates that good moves at one level are likely to be found close to good moves at the next level in a construction process (Glover and Laguna 1997). In order to exploit this principle, a search process should stay at a particular level for a chosen number of iterations and should then restore the best structure found so far before moving to the next level. In the flow time prediction context, this best structure at every stage is nothing but the best neural network architecture found. This is the reason why we use a GA-based architecture search to find the best neural network architecture at each level before moving to the next level in the boosting process.

3.4 *Genetic algorithm to design neural network architecture*

In prior research, pre-specified neural network architecture was used to predict due dates (Philipoom *et al*. 1994, 1997). This approach may be inappropriate for several reasons. First, ANN training methods use all input variables to build a prediction model. However, it is unlikely that all the connections emanating from these variables contribute significantly to a final model. During ANN training, weights associated with individual variables are not allowed to have a zero value. Hence, algorithms have to match noisy non-zero weights against each other so as to cancel their effects on final prediction. While this process minimizes the detrimental influence of such weights on the accuracy of training data, it may not minimize the influence of such weights on unknown test data, as it was not included during the design of an ANN architecture (Sexton *et al*. 2003).

Second, over-fitting has been identified as another problem encountered during the training of ANNs (German *et al*. 1992). This is an outcome of additional and unnecessary weights used during the training process to map specific points in a learning data set. When over-fitting occurs, a neural network (NN) simply memorizes learning data instead of discovering informative hidden patterns. Thus, an NN algorithm should strike a balance between finding a good sample fit by using an appropriate number of parameters and building a parsimonious model by removing unnecessary weights. Clearly, a single pre-specified network cannot identify and remove noisy variables, and eliminate insignificant connections (weights), and therefore it cannot build a parsimonious model.

The design of an optimal ANN can be viewed as a search problem in an architecture space where each architecture represents a solution vector. The optimal value of architecture on a performance measure (such as minimizing error or

minimizing model complexity) forms a response surface, and the problem is to find a solution vector (architecture) that maximizes the fitness. Genetic algorithms comprise a global search method that build a population of solutions and combine them using operators such as mutation and crossover to develop a better population of solutions (Sexton *et al*. 2003). The iterative nature of the GA survival of the fittest strategy builds high quality solutions in a reasonable amount of time.

In an ANN context, mutation can be either the addition of a new hidden node or the connection or deletion of an existing hidden node or connection (Yao and Liu 1997). Thus, a mutation operator not only identifies irrelevant noisy weights but also searchers for additional data patterns (Sexton *et al*. 2003). This process can help in building a global and yet parsimonious NN architecture. A crossover operator combines two NN architectures (Yao and Liu 1997). To build a good parsimonious prediction model, it is also equally important to define an objective function that includes both an 'error' component and a 'model complexity' component. Such an objective function can also be useful to explicitly evaluate the influence of mutation moves such as deleting connections on both quality and complexity dimensions. The use of GA allows us to incorporate these complex objective functions (Sexton *et al*. 2003). Clearly, a GA-based NN design search can build a global and parsimonious model of DDQ that can produce reliable and quality predictions.

3.5 *Proposed due date quotation algorithm*

In this section, we report the proposed due date quotation algorithm. It combines the concepts discussed in the previous sections to enhance DDQ accuracy. Figure 1 conceptually illustrates the bagging and boosting processes. Stage 1 applies the boosting process to different bootstrap samples to build a bagging-boosting ensemble. Step 1 from the algorithm constructs a diversified bootstrap sample, which is needed to improve the effectiveness of the bagging process. In step 2, an ensemble of ANNs is constructed using the boosting process. First, a residual vector is computed, which is then given to GA-based ANN to build a component function. The algorithm then adds a component function to a master function.

The master function assigns weights (importance) to each of its component functions depending upon its quality, which is determined via correlation coefficients between residuals and its predictions. This process is used to reduce noise that may be added to the boosted function due to noisy component functions. The algorithm also considers the ratio between standard deviation of residual vector and standard deviation of prediction vector while assigning weights. The higher the correlation coefficient and the ratio, the higher the weight assigned to the component function.

Let,

I = number of neural networks in the bagging.
J = number of neural networks in the boosted network.
N = size of original training set.
m_i = sample size of bootstrap sample i used for training neural network.
C_n = cumulative frequency of occurrence of observation n, initialize to 1.
a, b = lower and upper bounds on the bootstrap sample size.
$\mathbf{V}$ = vector of actual values for dependent variable (flow time) for a bootstrap sample i.

f_j = component function constructed at the iteration j during boosting process.
F_j = master function of a boosting process at the end of iteration j.
F_i = final master function created by boosting process for bootstrap sample i.
$\mathbf{R}_j$ = vector of residual values at iteration j for a bootstrap sample i.
input = matrix of values of predictor variables for a bootstrap sample i.
α_j = weight assigned to component function 'j' during boosting process.
δ_i = weight assigned to each master function during bagging process.

Algorithm

I. Build a 'bagging-boosting' ensemble of ANNs.
For $i = 1, \ldots, I$
1. Generate a diversified bootstrap training sample as follows:
   - Select bootstrap sample size, $m_i \sim \mathrm{U}(a, b)$.
   - For $n = 1, \ldots, N$
     a. Generate $x \sim \mathrm{U}(0, 1)$
     b. Set $w_n \rightarrow xC_n$
     End For
   - Rank training data points in ascending order of w_n
   - Generate bootstrap training sample 'i' from first m_i data points.
   - Update $C_n \rightarrow C_n + 1, \forall n \in$ bootstrap sample i
2. Create the boosted network using the bootstrap sample as follows:
   - Set $\mathbf{D}(x) \rightarrow 1/m_i, \forall x \in$ bootstrap sample i
   - Initialize master function F_0 to a zero function.
   - Set the parameters of GA for an architecture search.
   - For $j = 1, \ldots, J$
     - Set $\mathbf{R}_j \rightarrow \mathrm{V} - \sum_{j=0}^{j-1} \mathrm{F}_j$
     - Set $y_j^{\sim} \rightarrow \mathbf{R}_j - \mathrm{Avg}(\mathbf{R}_j)$
     - Run GA-based ANN architecture search on the modified data $(input, y_j^{\sim})$ having distribution $\mathbf{D}$ and record the best architecture (component function f_j) and its predictions on the bootstrap data.
     - Set

$$\varepsilon_j \rightarrow \frac{(\mathbf{R}_j - \mathrm{Avg}(\mathbf{R}_j))\,(f_j - \mathrm{Avg}(f_j))}{\left\|\mathbf{R}_j - \mathrm{Avg}(\mathbf{R}_j)\right\|_2 \left\|f_j - \mathrm{Avg}(f_j)\right\|_2}$$

     - Set

$$\alpha_j \rightarrow \frac{\varepsilon_t \left\|\mathbf{R}_j - \mathrm{Avg}(\mathbf{R}_j)\right\|_2}{\left\|f_j - \mathrm{Avg}(f_j)\right\|_2}$$

     - Set $\mathrm{F}_j \rightarrow \mathrm{F}_j - 1 + \alpha_j f_j$
     End For
   - Set $\mathrm{F}_i \rightarrow \mathrm{F}_j$
End For

II. Forecast lead-time using boosting and bagging ensemble.
- Set $\delta_i \rightarrow (1/I), \forall i$
- Forecast lead-time of an order with input vector '**T**' as $leadtime \rightarrow \sum_i \delta_i \mathrm{F}_i(\mathbf{T})$

4. Computational experiment

A computational experiment was undertaken to investigate the efficacy of boosting bagging, and a GA-based architecture search in improving the ANN-based DDQ performance.

4.1 *Experimental design*

Prior research has shown that the performance of the ANN-based DDQ depends upon training sample size and shop structure (Sha and Hsu 2004). Hence, in order to examine the effectiveness of the proposed ideas under different shop scenarios for a standard deviation of lateness criterion, thus to test both the accuracy and reliability of the proposed method, we use the three-factor design as shown in table 1. Standard deviation of lateness is a frequently used criterion to measure the performance of the due date quotation method where lateness can be defined as follows.

$$Lateness = Actual\ Flow\ time - Forecasted\ Flow\ time$$

We used one hidden layer, 9 hidden units of neural network architecture as proposed by Philipoom *et al*. (1994) as the conventional neural network method. We used 5 general attributes in the first 4 hidden units and the remaining 5 hidden units were constructed using the 15 detailed level attributes. Refer to Philipoom *et al*. (1994) paper to understand the neural network design, training procedures and algorithmic details. A GA-based method was used to identify the best neural network architecture. A GA-based boosting method with only the boosting algorithm from section 3.5 was used to build a network of ANNs. A GA-based bagging and boosting method using the algorithm proposed in section 3.5 was used to create an ensemble of ANNs. We used the four different levels of the prediction method to systematically demonstrate the importance of combining different machine learning principles in improving accuracy. This idea is effectively used in the metaheuristics literature where methods such as tabu search, path relinking, and scatter search are combined to produce high quality solutions (Glover and Laguna 1997).

Table 1. Experimental factors used in the computational study.

Factor	Level	Description
ANN method	4	1. Conventional method 2. GA-based ANN method 3. Boosting with GA-based ANN method 4. Bagging boosting with GA-based ANN method
Shop environment	3	1. Standard 2. Structured 3. Unstructured shop
Learning data size	5	500, 2000, 3500, 5000, 6500

Prior research has shown that the performance of an ensemble ANN saturates as the number of ANNs in the ensemble approaches 8 to 10 (Perrone and Cooper 1993). This gives a measure of the number of distinct neural networks in the population. In the proposed algorithm, the size of a neural network ensemble is equal to $I \times J$. Hence, we use $I = 3$ and $J = 3$ in our computational experiments, resulting in an ensemble of 9 neural networks for the DDQ model.

4.2 *Shop environments*

We simulated three shop environments: standard, structured and unstructured. In the standard shop, the arriving job was routed thorough 5 stages in a sequential order. Each stage had two machines to process jobs, and each job was processed at one of these two machines. This decision was made when the job entered the shop. Machine processing times were drawn from an exponential distribution with a mean of 1.8, and inter-arrival times were drawn from an exponential distribution with a mean of 1.0. This resulted in a shop utilization of 90 per cent. Jobs in queue were dispatched using a shortest processing time rule. In the structured shop, jobs were routed deterministically through five machines in a sequential manner. The unstructured shop consisted of 15 machines with completely random job routings. The number of operations per job was varied between 1 and 15 as per geometric distribution. Commercial simulation software with customized software routines was used to simulate the shop environments and to collect data.

Prior DDQ research has demonstrated the importance of including important predictors such as total work content and number of operations in the DDQ model (Ragatz and Mabert 1984). We also use these predictor variables in our experiments, along with several others. For the unstructured shop, there would be 15 operations instead of the 5 operations as reported below. Philipoom *et al.* (1994) provide the descriptions of and the rationale for including the following job and shop specific characteristics for flow time prediction:

Input	Description
General job and shop characteristics	
1	Number of operations required by job *i*
2	Sum of processing times for job *i*
3	Sum of jobs presently in queues on job *i* routing
4	Sum of expected waiting times at machines on job *i* routing
5	Total processing time of jobs in queues on job *i* routing
Detailed job and shop specific characteristics	
6–10	Number of jobs presently in queue on the machines which do operations 1–5
11–15	Number of operations which must be done on machine which does operations 1–5 on job *i* in order to complete all jobs presently in the shop
16–20	Processing time for operations 1–5

Replication period in the simulation experiments was varied according to the sample size. For example, for experiments involving training data sizes of 3500 jobs, shops were simulated for 8700 jobs in two replications of 3600 and 5100 jobs respectively. For experiments involving training data size of 5000 jobs, shops were simulated for 10 200 jobs in two replications of size 5100 jobs each. A warm up period of 100 jobs was used in both replications to reach normal conditions. As a job entered the shop, all relevant predictor values were recorded, and as the job left the shop, its actual flow time was recorded.

Data from the first replication was used to construct a training data set, and data from the second replication was used to construct a test data set. Billings *et al*. (1992) have pointed out the importance of using separate data to test a neural network's predictive ability. Consistent with Friedman's (1999) recommendations for the size of a bootstrap sample, we used 70, 75, and 66.6% to select the bootstrap sample sizes in our experiments.

4.3 *ANN training and testing*

We used a back propagation learning algorithm to determine neural network weights for a given architecture. This is a widely recommended training algorithm for regression problems (Philipoom *et al*. 1994, Hsu and Sha 2004). During the training process, we used a bootstrap sample to train a neural network and the remaining (out of bag) data points to stop the training process.

One important goal of this study was to examine the use of GA-based architecture search in the DDQ problem. We used the following parameters values for a GA-based ANN architecture search after conducting design of experiments analysis over the parameters: population size = 25; number of generations = 200; crossover rate = 0.8; mutation rate = 0.2; and penalty term = 5. The penalty term was used to restrict the number of connections used in the final ANN model. We used a GA algorithm to determine mutation and crossover strategies to create a range of ANN designs, and subsequently trained and evaluated them using the procedure above. The algorithm selects the best ANN architecture found in the search process, which best refers to an architecture that achieves the best performance on the objective function that includes both quality and complexity dimensions. Finally, a network of the ANNs constructed during the boosting and bagging process was used to predict flow times for orders in the test data set. Predictions are then used to compute the standard deviation of lateness on the test data set. Commercial neural network and GA software was used to perform the GA-based search and to train neural networks.

5. Results and discussion

Tables 2, 3 and 4 summarize the results of the computational experiments for standard, structured and unstructured shops respectively under different training data sizes. Conventional linear models were not examined in this study as prior research has consistently demonstrated the superior performance of ANNs compared to regression (Sha and Hsu 2004). The batch-means method was used to conduct paired *t*-tests where 10 batches of size 500 each were formed. Large batch sizes were chosen so that

Table 2. Comparison (statistical) of due date prediction models performance on standard deviation of lateness criterion for standard shop under different training data sizes.

Sample size		500	2000	3500	5000	6500
Conventional ANN vs. GA-based ANN	t	12.23	11.92	10.68	8.49	7.83
	p	0.0001	0.0001	0.0001	0.0001	0.0001
GA-based ANN vs. boosting	t	10.82	9.39	7.43	8.03	6.29
	p	0.0001	0.0001	0.0001	0.0001	0.0001
Boosting vs. bagging and boosting	t	8.57	6.15	6.93	5.61	3.12
	p	0.0001	0.0001	0.0001	0.00017	0.006

Table 3. Comparison (statistical) of due date prediction models performance on standard deviation of lateness criterion for structured shop under different training data sizes.

Sample size		500	2000	3500	5000	6500
Conventional ANN vs. GA-based ANN	t	8.54	7.62	6.37	5.89	5.07
	p	0.0001	0.0001	0.0001	0.0001	0.0001
GA-based ANN vs. boosting	t	7.43	8.18	7.96	6.77	4.91
	p	0.0001	0.0001	0.0001	0.0001	0.0001
Boosting vs. bagging and boosting	t	3.38	2.21	1.09	1.53	1.15
	p	0.004	0.027	0.15	0.08	0.14

Table 4. Comparison (statistical) of due date prediction models performance on standard deviation of lateness criterion for unstructured shop under different training data sizes.

Sample size		500	2000	3500	5000	6500
Conventional ANN vs. GA-based ANN	t	14.37	12.19	13.42	8.78	8.92
	p	0.0001	0.0001	0.0001	0.0001	0.0001
GA-based ANN vs. boosting	t	12.71	11.52	11.63	9.46	7.65
	p	0.0001	0.0001	0.0001	0.0001	0.0001
Boosting vs. bagging and boosting	t	9.95	8.36	7.99	6.52	5.04
	p	0.0001	0.0001	0.0001	0.0001	0.00034

batch-means became approximately uncorrelated and normally distributed and thus met the assumptions of the paired t-tests. Also, the use of a single population eliminated the need to test the homoskedasticity of variances. One tailed test of hypothesis for the standard deviation of lateness was formulated as follows:

$$H_0: \mu_D \leq 0$$
$$H_1: \mu_D \geq 0$$

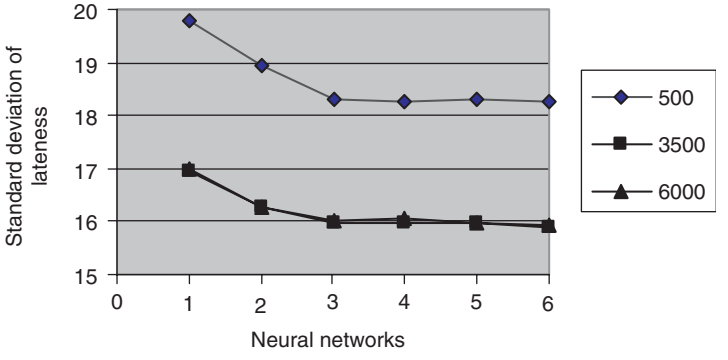

Figure 2. Impact of adding ANNs on due date prediction accuracy for standard shop.

where D is the difference between the standard deviation of lateness of the two methods under comparison. The level of significance was set to 0.01 and Bonferroni's inequality was used to control the experiment wise error rate due to multiple comparisons (therefore *p*-value must be less than 0.0033 for comparisons to be statistically significant). Results indicated that a strategy of using a GA for searching the neural network architecture resulted in a considerable improvement in flow time prediction performance. This was because the GA method was able to choose a better architecture in terms of the number of hidden nodes, active connections, and input variables compared to a strategy of using a fixed architecture.

The GA algorithm used a penalty term to reduce the complexity of the final ANN model to reduce variance error. In addition, we used validation error as a criterion to stop the training process to avoid over-fitting and thereby reducing variance error. As discussed earlier, the flow time prediction problem exhibits higher-order interaction effects. We conjectured that a single parsimonious ANN might not capture all flow time patterns, and therefore used a boosting process to capture any additional patterns.

Prediction performance was significantly improved with the use of the boosting method suggesting the ability of boosting to capture additional flow time patterns with the use of additional ANNs. Figure 2 shows the standard deviation of lateness reduced as number of ANNs increased in a standard shop. The figure also shows the 'diminishing returns' nature of the boosting process. This is because the number of missed hidden patterns required to improve accuracy decreases with the increase of the number of ANNs, thereby reducing the importance of additional ANNs.

Figure 2 also reveals that a v-shaped trend was not observed during the boosting process. This was because our boosting method added the contribution of each neural network based on its quality determined via the correlation coefficient and thus reduced the damage caused due to the inclusion of noisy neural networks. The figure also demonstrates how the performance of boosting algorithm varied with the sample size. Specifically, as the sample size decreased, the magnitude of the benefit from the boosting enhanced.

Finally, we used the bagging process with the boosting and GA method. Figures 3, 4 and 5 showed the results of the prediction methods under standard,

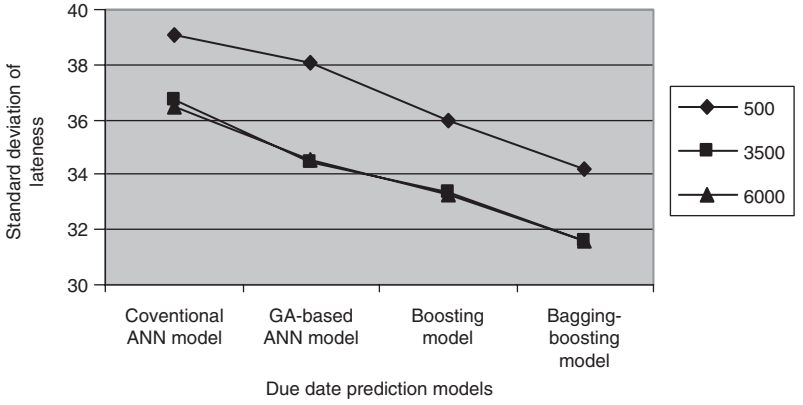

Figure 3. Performance of due date prediction models in standard shop.

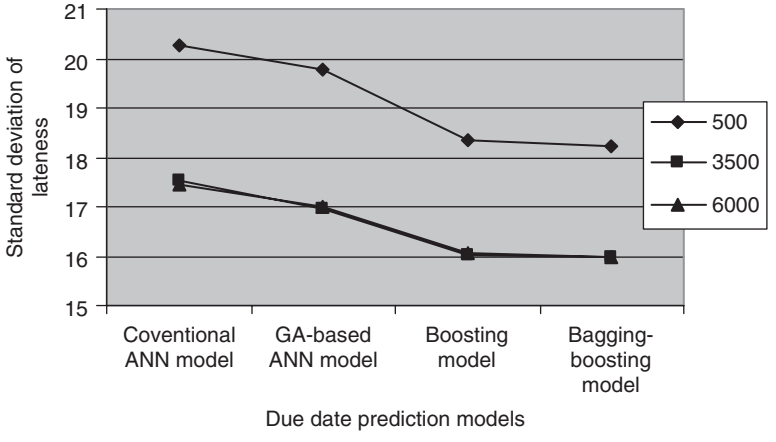

Figure 4. Performance of due date prediction models in structured shop.

structured and unstructured shop situations for different sample sizes. The results for the bagging process were variable although always positive – sometimes bagging resulted in considerable improvement and in other situations, did not improve the performance. Also, the magnitude of the benefit of using bagging increased as the shop complexity increased and the sample size decreased.

We used the boosting process, to reduce bias by using a network of independent ANNs. The performance of bagging depends upon the complexity of the function estimated by a boosted network. If the boosting process builds a smooth function that is a reasonably good estimate of a true function, then bagging might not improve accuracy because the function will not have a large variance error. However, if the boosting process has constructed a complex function, then bagging improves accuracy because of its ability to smooth the function.

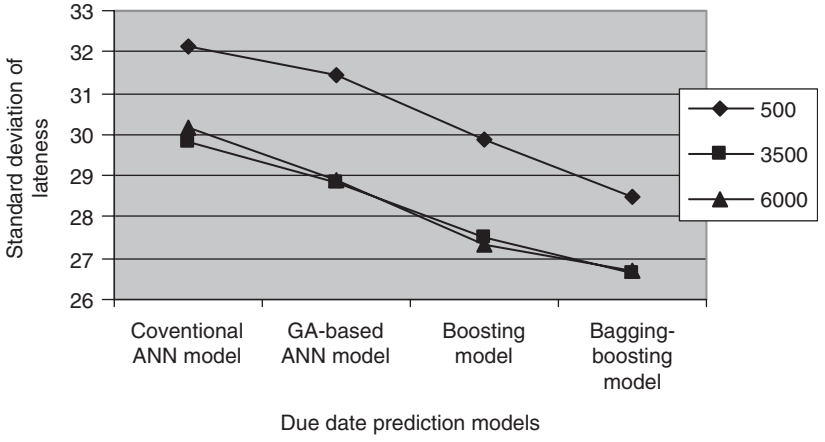

Figure 5. Performance of due date prediction models in unstructured shop.

Table 5. Computational time (in seconds) needed for prediction models.

	Sample size	Conventional ANN	GA-based ANN	Boosting-based ANN	Bagging-boosting ANN
Standard shop	500	6.5	165.5	440.5	1316
	3500	21.5	410.5	1193.5	3594.5
	6500	31.0	683.0	1986	5942.5
Structured shop	500	6	169	448.5	1347.5
	3500	20.5	424	1202	3612.5
	6500	30.5	681.5	1979.5	5928
Unstructured shop	500	12.5	228.5	646.5	1927.5
	3500	40.5	583.5	1703	5116.5
	6500	57.5	895.5	2684.5	8051

It is also important to understand the computational implications of the proposed methodology. Table 5 shows how the computational load (in seconds) varied with the due date prediction models for all the shops at 500, 3500 and 6500 sample sizes. We timed our experiments on an IBM machine with an Intel Pentium 4 processor of 2600 MHz. Though computational load increased with the size of bagging and boosting networks, and the GA-based search, we believe that the improvements in prediction accuracy are worth the additional computational load. This is because DDQ design can affect a firm's profitability and reputation over the long term. Clearly, the proposed DDQ model demonstrates the potential to improve a firm's performance because of its ability to quote accurate due dates. Second, the number of training data points needed for building a DDQ model are often considerably lower compared to the practical capacity of today's ANN algorithms. Coupled with advances in computing power, we believe that it is possible to accomplish the model building process in a reasonable amount of time. Third, the parameters used in

the computation are controllable. That is, users have the flexibility to select parameter settings that reduce computational load, although this strategy may result in deterioration of prediction performance.

6. Conclusions

The purpose of this study was to test the use of machine learning and metaheuristic methods to improve ANN-based due date quotation performance. Computational results indicate that these methods significantly improve the accuracy of flow time prediction. Genetic algorithm-based neural network architecture search reduced standard deviation of lateness because of its ability to construct a global and parsimonious model of flow time prediction with the use of crossover and mutation operators. Boosting reduced the standard deviation of lateness because of its ability to capture the hidden patterns in flow time prediction in an effective manner. The benefit of using the bagging process, however, varied across computational experiments. Sometimes, bagging improved accuracy by performing smoothing in a functional space and by constructing a better function with the use of a combination of ANN models. However, sometimes, it did not result in improvement. We conclude that machine-learning concepts can be useful to further improve ANN-based due date prediction performance. More importantly, the improvement can be accomplished for a given learner, ANN, and without the additional use of training data points.

Our model is designed to improve the instable nature of ANNs, and hence, can also be used with other instable learners such as classification and regression trees (CART). In the DDQ context, Sha and Liu (2005) have investigated the use of CART in improving the due date performance on mean squared lateness and mean absolute lateness performance in a dynamic job shop. It will be interesting to investigate the use of our model with CART in this situation. ANNs have been used to solve other important managerial problems in operations, accounting, and finance as discussed in the literature review section. The prediction model is general in nature, and hence, we believe that it is worthwhile to examine the effectiveness of the model in these problem contexts.

Acknowledgements

The authors would like to thank Prof. Stephen R. Lawrence for his valuable suggestions and guidance offered during their research project.

References

Billings, S.A., Jamaluddin, H.B. and Chen, S., Properties of neural networks with application to modeling non-linear dynamic systems. *Int. J. Cont.*, 1992, **55**, 193–224.

Breiman, L., Bagging predictors. *Mach. Learn.*, 1996a, **24**, 123–140.

Breiman, L., Bias, variance, and arcing classifiers. Technical report: technical report 460, Statistics department, University of California, Berkley, 1996b.

Cheng, T.C.E. and Gupta, M.C., Survey of scheduling research involving due date determination decisions. *Eur. J. Oper. Res.*, 1989, **38**, 156–166.

Dietterich, T.G., Ensemble methods in machine learning. *Lect. Notes. Comput. Sci.*, 2000, **1857**, 1–15.

Duffy, N. and Helmbold, D., Leveraging for regression, in *Proceedings of 13th Annual Conference on Computational Learning Theory*, 2000, pp. 208–219.

Freund, Y. and Schapire, R.E., Experiments with a new boosting algorithm, in *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996, pp. 148–156. 1996.

Friedman, J.H., Stochastic gradient boosting. Technical report, Department of Statistics, Stanford, 1999.

Friedman, J.H., Recent advances in predictive (machine) learning. Technical report, Department of Statistics, Stanford, 2003.

German, S., Bienenstock, E. and Doursat, R., Neural networks and the bias and variance dilemma. *Neural Comput.*, 1992, **4**, 1–58.

Glover, F. and Laguna, M., *Tabu Search*, 1997 (Kluwer Academic Publishers: MA).

Hansen, L.K. and Salamon, P., Neural Network ensembles. *IIE Trans. Pattern Anal. Machine Learn.*, 1990, **12**, 993–1001.

Hastie, T., Tibshirani, R. and Friedman, J.H., *The Elements of Statistical Learning: Data Mining, Interference and Prediction*, 2001 (Springer: NY).

Hsu, S.Y. and Sha, D.Y., Due date assignment using artificial neural networks under different shop floor strategies. *Int. J. Prod. Res.*, 2004, **42**, 1727–1745.

Huang, C.L., Huang, Y.H., Chang, T.Y., Chang, S.H., Chung, C.H., Huang, D.T. and Li, R.K., The construction of production performance prediction system for semiconductor manufacturing with artificial neural networks. *Int. J. Prod. Res.*, 1999, **37**, 1387–1402.

Keskinocak, P. and Tayur, S., Due date management policies. In *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-business era*, edited by D. Simchi-Levi, S.D. Wu and Z.J. Shen, pp. 485–554, 2004 (Kluwer Academic Publishers: MA).

Kotsiantis, S.B. and Pintelas, P.E., Combining bagging and boosting. *Int. J. Comput. Intell.*, 2004, **1**, 324–333.

Perrone, M.P. and Cooper, L.N., hen networks disagree: Ensemble methods for hybrid neural networks. In *Artificial Neural Networks for Speech and Vision*, edited by R.J. Mammone, pp. 126–142, 1993 (Chapman & Hall: London).

Philipoom, P.R., Rees, L.P. and Wiegman, L., Using neural networks to determine internally set due-date assignments for shop scheduling. *Deci. Sci.*, 1994, **25**, 825–847.

Philipoom, P.R, Wiegman, L. and Rees, L.P., Cost based due date assignment with the use of classical and neural network approaches. *Nav. Res. Log.*, 1997, **44**, 21–46.

Ragatz, G.L. and Mabert, V.A., A simulation analysis of due date assignment rules. *J. Oper. Manag.*, 1984, **5**, 27–39.

Sabuncuoglu, I. and Comlekei, A., Operation-based flow time estimation in a dynamic job shop. *Int. J. Manage. Sci.*, 2002, **30**, 423–442.

Sen, T. and Gupta, S.K., A state-of-the-art survey of static scheduling research involving due dates. *Int. J. Man. Sci.*, 1984, **12**, 63–76.

Sexton, R.S., Sriram, R.S. and Etheridge, H., Improving decision effectiveness of artificial neural networks: a modified genetic algorithm approach. *Deci. Sci.*, 2003, **34**, 421–442.

Sha, D.Y. and Hsu, S.Y., Due date assignment in wafer fabrication using artificial neural networks. *Int. J. Adv. Manuf. Tech.*, 2004, **23**, 768–775.

Sha, D.Y. and Liu, C.H., Using data mining for due date assignment in a dynamic job shop environment. *Int. J. Adv. Manuf. Tech.*, 2005, **25**, 1164–1174.

Slotnick, S.A. and Sobel, M.J., Lead time rules in distributed manufacturing systems. Technical Memorandum Number 743. Department of Operations, Weatherhead School of Management, OH, 2001.

Suen, Y.L., Melville, P. and Mooney, R., Combining bias and variance reduction techniques for regression. Technical Report UT-AI-TR-05-321, Artificial Intelligence Lab, University of Texas at Austin, 2005.

Tam, K.Y. and Kiang, M.Y., Managerial applications of neural networks: the case of bank failure predictions. *Manage. Sci.*, 1992, **38**, 926–947.

Udo, G.L., Neural networks applications in manufacturing process. *Comp. Indust. Eng.*, 1992, **23**, 97–100.

Yao, X. and Liu, Y., New evolutionary system for evolving artificial neural networks. *IEEE Trans. Neural Networ.*, 1997, **8**, 694–713.

Zhou, Z.H., Wu, J. and Tang, W., Ensembling neural networks: many could be better than all. *Artif. Intell.*, 2002, **137**, 239–263.