Neural Network and Regression Spline Value Function Approximations for Stochastic Dynamic Programming

Cristiano Cervellera
Institute of Intelligent Systems for Automation - ISSIA-CNR
National Research Council of Italy
Via De Marini 6, 16149 Genova, Italy
Email: cervellera@ge.issia.cnr.it

Aihong Wen and Victoria C. P. Chen*
Department of Industrial & Manufacturing Systems Engineering
The University of Texas at Arlington
Campus Box 19017, Arlington, TX 76019-0017, USA
Email: axw0932@omega.uta.edu, vchen@uta.edu

COSMOS Technical Report 04-05

Abstract

Dynamic programming is a multi-stage optimization method that is applicable to many problems in engineering. A statistical perspective of value function approximation in high-dimensional, continuous-state stochastic dynamic programming (SDP) was first presented using orthogonal array (OA) experimental designs and multivariate adaptive regression splines (MARS). Given the popularity of artificial neural networks (ANNs) for high-dimensional modeling in engineering, this paper presents an implementation of ANNs as an alternative to MARS. Comparisons consider the differences in methodological objectives, computational complexity, model accuracy, and numerical SDP solutions. Two applications are presented: a nine-dimensional inventory forecasting problem and an eight-dimensional water reservoir problem. Both OAs and OA-based Latin hypercube experimental designs are explored, and OA space-filling quality is considered.

Key Words: design of experiments, statistical modeling, Markov decision process, orthogonal array, Latin hypercube, inventory forecasting, water reservoir management.

1 Introduction

The objective of dynamic programming (**DP**) is to minimize a "cost" subject to certain constraints over several stages, where the relevant "cost" is defined for a specific problem. For an inventory problem, the "cost" is the actual cost involved in holding inventory, having to backorder items, etc., and the constraints involve capacities for holding and ordering items. An equivalent objective is

*Supported by NSF Grants #INT 0098009 and #DMI 0100123 and a Technology for Sustainable Environment (TSE) grant under the U. S. Environmental Protection Agency's Science to Achieve Results (STAR) program (Contract #R-82820701-0).

to maximize a "benefit," such as the robustness of a wastewater treatment system against extreme conditions. The *state* variables track the state of the system as it moves through the stages, and a *decision* is made in each stage to achieve the objective.

Recursive properties of the DP formulation permit a solution via the fundamental recurrence equation (Bellman 1957). In stochastic dynamic programming (**SDP**), uncertainty is modeled in the form of random realizations of stochastic system variables, and the estimated expected "cost" ("benefit") is minimized (maximized). In particular, Markov decision processes may be modeled by a SDP formulation through which the state variables resulting from each decision comprise a Markov process. DP has been applied to many problems in engineering, such as water resources, revenue management, pest control, and wastewater treatment. For general background on DP, see Bellman (1957), Puterman (1994), and Bertsekas (2000). For some applications, see Gal (1979), Shoemaker (1982), White (1985, 1988), Foufoula-Georgiou and Kitanidis (1988), Culver and Shoemaker (1997), Chen, Günther, and Johnson (2002), and Tsai et al. (2004). In high dimensions, traditional DP solution methods can become computationally intractable and attempts have been made to reduce the computational burden (Johnson et al. 1993, Eschenbach et al. 1995, Bertsekas and Tsitsiklis 1996).

For continuous-state SDP, the statistical perspective of Chen et al. (1999) and Chen (1999) enabled the first truly practical numerical solution approach for high dimensions. Their method utilized orthogonal array (**OA**, Chen 2001) experimental designs and multivariate adaptive regression splines (**MARS**, Friedman 1991). Although MARS has been applied in several areas (e.g., Carey and Yee 1992, Griffin et al. 1997, Kuhnert et al. 2000, Shepherd and Walsh 2002), artificial neural network (**ANN**) models are much more prevalent in all areas of engineering (e.g., Smets and Bogaerts 1992, Labossiere and Turkkan 1993, Abdelaziz et al. 1997, Chen and Rollins 2000, Li et al. 2000, Liu 2001, Pigram and Macdonald 2001, Kottapalli 2002). Given their popularity, an ANN-based SDP solution method would be more accessible to engineers. In this paper, we present:

- The statistical modeling approach to solving continuous-state SDP.
- Implementation of ANN models in continuous-state SDP as an alternative to MARS.
- Implementation of OA-based Latin hypercube (**OA-LH**) designs (Tang 1993) as an alternative to pure OAs.
- Consideration of the "space-filling" quality of generated experimental designs. The designs are intended to distribute discretization points so that they "fill" the state space; however, generated designs of the same type do not necessarily have equivalent space-filling quality.
- Comparisons between ANN and MARS value function approximations, including methodological perspectives, computational considerations, and numerical SDP solution quality on two applications.

The novelty of our study is to simultaneously consider how the SDP solution is affected by different approximation methods and designs of different types and different space-filling qualities.

In the statistical perspective of continuous-state SDP, the two basic requirements are an experimental design to discretize the continous state space and a modeling method to approximate the value function An appropriate design should have good space-filling quality, and an appropriate modeling method should be flexible in high dimensions, provide a smooth approximation, and be reasonably efficient. A review of several experimental designs and modeling methods is provided by Chen, Tsui et al. (2003) in the context of computer experiments. Of the possible modeling choices given in this review paper, only MARS, neural networks, and kriging satisfy the first two criteria.

Some preliminary exploration of the kriging model yielded positive results on a ten-dimensional wastewater treatment application (Chen and Welch 2001), but solving for the maximum likelihood estimates of the kriging parameters continues to be a computational challenge. For this application, kriging required 40 times more computational effort than MARS. In addition, kriging software is even harder to obtain than MARS software, which is counter to our motivation to employ ANNs.

Comparisons between MARS and ANNs for empirical modeling have been conducted by De Veaux et al. (1993) and Pham and Peat (1999), and in both studies MARS performed better overall. Although it is obvious that ANN models can be employed to approximate functions, the existing literature does not provide sufficient studies to predict how they will compare to MARS in the SDP setting. In continuous-state SDP, value functions are typically well-behaved (e.g., convex) and the SDP data involve little noise. Previous comparisons involved statistical data with considerable noise. It is expected that ANN models will perform better in low noise situations. However, ANN models can add extraneous curvature, a disadvantage when a smooth convex approximation is desired. MARS, on the other hand, constructs a parsimonious approximation, but its limited structure, although quite flexible, could affect its approximation ability, which would be most noticeable in low noise situations. Finally, computational speed is an important aspect of SDP, and the ANN model fitting process has a reputation of being slow.

Studies on different experimental designs for metamodeling have been conducted (Palmer 1998, Simpson, Lin et al. 2001, Allen et al. 2003). However, none have considered OA-LH designs. Given the success of OA designs in continuous-state SDP, we chose the OA-LH design as a variation that might work well with ANN models. Points of an OA design are limited to a fairly sparse grid, while the OA-LH design allows points to move off the grid. The grid restriction was convenient for the MARS structure, but provides no advantage for ANN models.

Section 2 describes SDP in the context of the two applications that are later used to test our methods. The SDP statistical modeling approach is given in Section 3. Section 4 provides background on OA and OA-LH designs. Section 5 describes ANNs, and Section 6 presents the comparisons between the ANNs and MARS, including the implementation of OA-LH designs. Conclusions are presented in Section 7.

2 SDP Applications

In continuous-state DP, the system *state variables* are continuous. Chen et al. (1999) and Chen (1999) solved inventory forecasting test problems with four, six, and nine state variables modeled over three stages. Application to ten- and seven-dimensional water reservoir SDP problems appears in Baglietto et al. (2001) and Baglietto et al. (2005), and application to a twenty-dimensional wastewater treatment system appears in Tsai et al. (2004). Here, we consider the nine-dimensional inventory forecasting problem, which is known to have significant interaction effects between inventory levels and forecasts due to the capacity constraints, and an eight-dimensional water reservoir problem, which has simpler structure and is more typical of continuous-state SDP problems.

2.1 Inventory Forecasting Problem

For an inventory problem, the inventory levels are inherently discrete since items are counted, but the range of inventory levels for an item is too large to be practical for a discrete DP solution. Thus, one can relax the discreteness and use a continuous range of inventory levels. The inventory forecasting problem uses forecasts of customer demand and a probability distribution on the forecasting updates to model the demand more accurately than traditional inventory models (see Hadley and Whitin 1963).

2.1.1 State and Decision Variables

The *state variables* for inventory forecasting consist of the inventory levels and demand forecasts for each item. For our nine-dimensional problem, we have three items and two forecasts for each. Suppose we are currently at the beginning of stage t. Let $I_t^{(i)}$ be the inventory level for item i at the *beginning of the current stage*. Let $D_{(t,t)}^{(i)}$ be the forecast made at the *beginning of the current stage* for the demand of item i occurring over the stage. Similarly, let $D_{(t,t+1)}^{(i)}$ be the forecast made at the *beginning of the current stage* for the demand of item i occurring over the *next* stage $(t+1)$. Then the *state vector* at the *beginning of stage* t is:

$$\boldsymbol{x}_t = \left(I_t^{(1)}, I_t^{(2)}, I_t^{(3)}, D_{(t,t)}^{(1)}, D_{(t,t)}^{(2)}, D_{(t,t)}^{(3)}, D_{(t,t+1)}^{(1)}, D_{(t,t+1)}^{(2)}, D_{(t,t+1)}^{(3)}\right)^T.$$

The *decision variables* control the order quantities. Let $u_t^{(i)}$ be the amount of item i ordered at the beginning of stage t. Then the *decision vector* is $\boldsymbol{u}_t = (u_t^{(1)}, u_t^{(2)}, u_t^{(3)})^T$, where $u_t^{(i)}$ is the amount of item i ordered in stage t. For simplicity, it is assumed that orders arrive instantly.

2.1.2 Transition Function

The *transition function* for stage t specifies the new state vector to be $\boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\epsilon}_t)$, where $\boldsymbol{x}_t$ is the state of the system (inventory levels and forecasts) at the beginning of stage t, $\boldsymbol{u}_t$ is the amount ordered in stage t, and $\boldsymbol{\epsilon}_t$ is the stochasticity associated with updates in the demand forecasts. The evolution of these forecasts over time is modeled using the multiplicative Martingale Model of Forecast Evolution, for which the stochastic updates are applied multiplicatively and assumed to have a mean of one, so that the sequence of future forecasts for stage $t+k$ forms a martingale (Heath and Jackson 1991). See Chen (1999) for details on the stochastic modeling.

2.1.3 Objectives and Constraints

The *constraints* for inventory forecasting are placed on the amount ordered, in the form of capacity constraints, and on the state variables, in the form of state space bounds. For this test problem, the capacity constraints are chosen to be restrictive, forcing interactions between the state variables, but are otherwise arbitrary. The bounds placed on the state variables in each stage are necessary to ensure accurate modeling over the state space.

The *objective function* is a cost function involving inventory holding costs and backorder costs. The typical V-shaped cost function common in inventory modeling is:

$$c_v(\boldsymbol{x}_t, \boldsymbol{u}_t) = \sum_{i=1}^{n_I} (h_i[I_{t+1}^{(i)}]_+ + \pi_i[-I_{t+1}^{(i)}]_+), \tag{1}$$

where $[q]_+ = \max\{0, q\}$, h_i is the holding cost parameter for item i, and π_i is the backorder cost parameter for item i. The *smoothed* version of equation (1), described in Chen (1999), is employed here.

2.2 Water Reservoir Problem

Optimal operation of water reservoir networks has been studied extensively in the literature (e.g., Archibald et al. 1997, Foufoula-Georgiou and Kitanidis 1988, Gal 1979, Lamond and Sobel 1995, Turgeon 1981; Yakowitz 1982 provides an excellent survey). The water reservoir network we consider consists of 4 reservoirs (see Figure 1 for the configuration). In typical models, the inputs to

the nodes are water released from upstream reservoirs and stochastic inflows from external sources (e.g., rivers and rain), while the outputs correspond to the amount of water to be released during a given time stage (e.g., a month).

2.2.1 State and Decision Variables

In a realistic representation of inflow dynamics, the amount of (random) water flowing into the reservoirs from external sources during a given stage t depends on the amounts of past stages. In this paper, we follow the realistic modeling of Gal (1979) and use an autoregressive linear model of order one. Thus, the ith component of the random vector $\boldsymbol{\epsilon}_t$, representing the stochastic external flow into reservoir i, has the form

$$\epsilon_t^{(i)} = a_t^{(i)} \epsilon_{t-1}^{(i)} + b_t^{(i)} + c_t^{(i)} \xi_t^{(i)} \ , \tag{2}$$

where $\boldsymbol{\xi}_t$ is a random correction that follows a standard normal distribution. The coefficients of the linear combinations $(\boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t)$ actually depend on t, so that it is possible to model proper external inflow behaviors for different months. The actual values of the coefficients used for the model were based on the one-reservoir real model described in Gal (1979) and extended to our 4-reservoir network.

Given equation (2), the *state vector* consists of the water level and stochastic external inflow for each reservoir:

$$\boldsymbol{x}_t = \left(w_t^{(1)}, w_t^{(2)}, w_t^{(3)}, w_t^{(4)}, \epsilon_t^{(1)}, \epsilon_t^{(2)}, \epsilon_t^{(3)}, \epsilon_t^{(4)} \right)^T \ ,$$

where $w_t^{(i)}$ is the amount of water in reservoir i $(i = 1, 2, 3, 4)$ at the beginning of stage t, and $\epsilon_t^{(i)}$ the stochastic external inflow into reservoir i during stage t. The *decision vector* is $\boldsymbol{u}_t = (u_t^{(1)}, u_t^{(2)}, u_t^{(3)}, u_t^{(4)})^T$, where $u_t^{(i)}$ be the amount of water released from reservoir i during stage t.

2.2.2 Transition Function

The *transition function* for stage t updates the water levels and external inflows. The amount of water at stage $t+1$ reflects the flow balance between the water that enters (upstream releases and stochastic external inflows) and the water that is released. In order to deal with unexpected peaks from stochastic external inflows, each reservoir has a floodway, so that the amount of water never exceeds the maximum value $W^{(i)}$. Thus, the new water level is

$$w_{t+1}^{(i)} = \min \left\{ w_t^{(i)} + \sum_{j \in U^{(i)}} u_t^{(j)} - u_t^{(i)} + \epsilon_t^{(i)} \ , \ W^{(i)} \right\} \ ,$$

where $U^{(i)}$ the set of indexes corresponding to the reservoirs which release water into reservoir i. For the stochastic external inflows, we can update them using equation (2). Like the inventory forecasting model, the two equations above can be grouped in the compact transition function $\boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\epsilon}_t)$.

2.2.3 Objectives and Constraints

The *constraints* for water reservoir management are placed on the water releases $u_t^{(i)}$ in the form:

$$0 \leq u_t^{(i)} \leq \min \left\{ w_t^{(i)} + \sum_{j \in U^i} u_t^{(j)} \ , \ R^{(i)} \right\}$$

where $R^{(i)}$ the maximum pumpage capability for reservoir i. This implies nonnegativity of w_{t+1}^i. The *objective function* is intended to maintain a reservoir's water level close to a target value $\tilde{w}_t^{(i)}$ and maximize benefit represented by a concave function g of the water releases and/or water levels (e.g., power generation, irrigation, etc.). For maintaining targets, the intuitive cost would be $|w_t^{(i)} - \tilde{w}_t^{(i)}|$, which is a V-shaped function. To facilitate numerical minimization, we utilized a smoothed version, denoted by $l(w_t^{(i)}, \tilde{w}_t^{(i)})$, that is analogous to the smoothed cost function for the inventory forecasting problem. For the modeling benefits, we consider a nonlinear concave function g of the following form (for $z \in [0, +\infty)$):

$$g(z, \mu, \beta) = \begin{cases} (\beta+1)z & 0 \le z \le \frac{1}{3}\mu \\ \beta z + \frac{1}{2}\mu - \frac{45}{\mu^2}(z - \frac{2}{3}\mu)^3 - \frac{153}{2\mu^2}(z - \frac{2}{3}\mu)^4 & \frac{1}{3}\mu \le z \le \frac{2}{3}\mu \\ \beta z + \frac{1}{2}\mu & z \ge \frac{2}{3}\mu \end{cases} .$$

Such a function models a benefit which becomes relevant for "large" values of the water releases, depending on suitable parameters μ and β. In our example, we used $R^{(i)}$ for μ and a small number 0.1 for β. Then, the total cost function at stage t can be written as

$$c_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\epsilon}_t) = \sum_{i=1}^{4} l(w_t^{(i)}, \tilde{w}_t^{(i)}) - \sum_{i=1}^{4} p^{(i)} g(u_t^{(i)}, \mu^{(i)}, \beta) \ ,$$

where the $p^{(i)}$,$i = 1, 2, 3, 4$ are weights chosen to balance the benefits against the costs.

3 SDP Statistical Modeling Solution Approach

3.1 Future Value Function

For a given stage, the *future value function* provides the optimal cost to operate the system from stage t through T as a function of the state vector $\boldsymbol{x}_t$. Following the notation in Section 2, this is written recursively as

$$\begin{array}{rll} F_t(\boldsymbol{x}_t) = & \min\limits_{\boldsymbol{u}_t} & E\{c(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\epsilon}_t) + F_{t+1}(\boldsymbol{x}_{t+1})\} \\ & \text{s.t.} & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\epsilon}_t), \\ & & (\boldsymbol{x}_t, \boldsymbol{u}_t) \in \Gamma_t. \end{array} \tag{3}$$

If we have the future value functions for all the stages, then given the initial state of the system, we can track the optimal solution through the value functions. Thus, given a finite time horizon T, the basic SDP problem is to find $F_1(\cdot), \ldots, F_T(\cdot)$. The recursive formulation permits the DP backward solution algorithm (Bellman 1957), first by assuming $F_{T+1}(\boldsymbol{x}_{T+1}) \equiv 0$, so that $F_T(\boldsymbol{x}_T) = c_T(\boldsymbol{x}_T)$ (where $c_T(\boldsymbol{x}_T)$ is the cost of the final stage depending only on $\boldsymbol{x}_T$), then solving in order for $F_T(\cdot)$ through $F_1(\cdot)$. See the description in Chen (1999) or Chen et al. (2000) for further details.

3.2 Numerical Solution Algorithm

Given the state of the system at the beginning of the current stage, the corresponding optimal cost yields one point on that stage's value function. The complete future value function is obtained by solving this optimization problem for all possible beginning states. For continuous states, this is an infinite process. Thus, except in simple cases where an analytical solution is possible, a numerical SDP solution requires a computationally-tractable approximation method for the future value function.

From a statistical perspective, let the state vector be the vector of predictor variables. We seek to estimate the unknown relationship between the state vector and the future value function. Response observations for the future value function may be collected via the minimization in equation (3). Thus, the basic procedure uses an experimental design to choose N points in the n-dimensional state space so that one may accurately approximate the future value function; then constructs an approximation.

Chen et al. (1999) introduced this statistical perspective using strength three OAs and MARS (smoothed with quintic functions) applied to the inventory forecasting test problems. MARS was chosen due to its ease of implementation in high dimensions, ability to identify important interactions, and easy derivative calculations. The choice of the strength three OAs was motivated by the desire to model interactions, but univariate modeling can suffer due to the small number of distinct levels in each dimension (e.g., $p = 13$ was the highest number of levels explored). Chen (1999) additionally explored strength two OAs with equivalent N to the strength three OAs, but with lower order balance permitting higher p; however, their performance was disappointing.

4 Discretizations based on Orthogonal Arrays

In continuous-state SDP, the classical solution method discretizes the state space with a full grid, then uses multi-linear interpolation to estimate the future value function. This full grid is equivalent to a full factorial experimental design, for which the number of points grows exponentially as the dimension increases. To avoid this, a fractional factorial design, i.e., taking a subset of the full grid, may be employed. OAs are special forms of fractional factorial designs. A full factorial design for n variables, each at p levels, contains all p^n level combinations. An OA of strength d ($d < n$) for n variables, each at p levels, contains all possible level combinations in any subset of d dimensions, with the same frequency, λ. Therefore, when projected down onto any d dimensions, a full factorial grid of p^d points replicated λ times is represented.

4.1 Measuring Space-filling Quality

There are a variety of ways to generate OAs (e.g., Rao 1946, Bose and Bush 1952, Raghavarao 1971, Hedayat and Wallis 1978, Dey 1985, Hedayat et al. 1996). However, there is no way to guarantee their space-filling quality. In practice, one can generate many OAs, calculate various space-filling measures, then choose the best of the bunch. Chen (2001) has conducted the only study of how this quality affects an approximation, and only OAs of strength three were studied.

Looking at the two configurations of points in Figure 2, the one on the left is seen to space the points better. There are two basic ways to measure space-filling quality: minimax and maximin (refer to Chen, Tsui et al. 2003). Let $\mathcal{D}$ be the n-dimensional closed and bounded set of the continuous state space, and $\mathcal{P}$ be the set of n design points in $\mathcal{D}$. Minimax criteria seek to minimize the maximum distance between (nondesign) points in $\mathcal{D}$ and the nearest neighbor design point in $\mathcal{P}$. Using the same distance measure as above, define:

$$\text{MAXDIST}(\mathcal{D}, \mathcal{P}) = \sup_{x \in \mathcal{D}} \{ \min_{y \in \mathcal{P}} d(x, y) \}.$$

Since the supremum over the infinite set of points in $\mathcal{D}$ is difficult to calculate, this measure is usually calculated over a finite sample of points in $\mathcal{D}$, using a full factorial grid or some type of sampling scheme. However, this measure is generally impractical to calculate in high dimensions. A design that represents $\mathcal{D}$ well will have lower $\text{MAXDIST}(\mathcal{D}, \mathcal{P})$ than a design that does not.

Maximin criteria seek to maximize the minimum distance between any pair of points in $\mathcal{P}$. Let $d(x, y)$ be a distance measure (commonly Euclidean distance) over $\mathcal{R}$. Then define:

$$\text{MINDIST}(\mathcal{P}) = \min_{x \in \mathcal{P}} \{ \min_{\substack{y \in \mathcal{P} \\ y \neq x}} d(x, y) \}.$$

Higher MINDIST($\mathcal{P}$) should correspond to a more uniform scattering of design points. For the OA designs in this paper, their space-filling quality was judged using measures of both types and a measure introduced by Chen (2001).

4.2 Orthogonal Array-based Latin Hypercube Designs

Latin hypercube (LH) designs are equivalent to OA designs of strength one. In other words, if we project the N points of a LH design onto any single dimension, the points will correspond to N distinct levels in that dimension. However, only OAs of strength two and higher are technically orthogonal, so the LH design does not maintain this property. A hybrid OA-LH design, for which an LH design is (randomly) created from an OA structure, was introduced by Tang (1993) to address the lack of orthogonality in LH designs. Although OA-LH designs only satisfy LH properties, the underlying OA provides some balance. With regard to approximating a function, the LH design provides more information along a single dimension (for univariate terms), while the OA design provides better coverage in higher dimensions (for interactions).

In this paper, we utilize the strength three OAs described in Chen (2001), and the OA-LH designs are based on these strength three OAs. Specifically, we employed strength three OAs with $p = 11$ levels ($N = 1331$) and $p = 13$ levels ($N = 2197$) in each dimension, and two realizations of OA-LH designs were generated for each OA. For each size ($N = 1331$ and $N = 2197$), one "good," one "neutral," and one "poor" OA was selected based on space-filling measures described above. Thus, a total of 6 OAs and 12 OA-LH designs were tested. For the inventory forecasting problem, nine-dimensional designs were generated, and for the water reservoir problem, eight-dimensional designs were generated.

5 Artificial Neural Networks

5.1 Model Description

ANNs are mathematical modeling tools widely used in many different fields (for general background and applications see Lippmann 1987, Haykin 1999, Bertsekas and Tsitsiklis 1996, Cherkassky and Mulier 1998; for statistical perspectives see White 1989, Barron et al. 1992, Ripley 1993, or Cheng and Titterington 1994; for approximation capabilities see Barron 1993). From a purely "analytical" perspective, an ANN model is a nonlinear parameterized function that computes a mapping of n input variables into (typically) one output variable, i.e. n predictors and one response. The basic architecture of ANNs is an interconnection of *neural nodes* that are organized in layers. The input layer consists of n nodes, one for each input variable, and the output layer consists of one node for the output variable. In between there are *hidden layers* which induce flexibility into the modeling. *Activation functions* define transformations between layers, the simplest one being a linear function. Connections between nodes can "feed back" to previous layers, but for function approximation, the typical ANN is *feedforward* only with one hidden layer and uses "sigmoidal" activation functions (monotonic and bounded). A comprehensive description of various ANN forms may be found in Haykin (1999).

In our inventory forecasting SDP application, we utilized a feedforward one-hidden-layer ANN (**FF1ANN**) with a "hyperbolic tangent" activation function: $\text{Tanh}(x) = (e^x - e^{-2x})/(e^x + e^{-x})$. Define n as the number of input variables, H as the number of hidden nodes, v_{jh} as weights linking input nodes j to hidden nodes h, w_h as weights linking hidden nodes h to the output node, and θ_h and γ as constant terms called "bias" nodes (like intercept terms). Then our ANN model form is:

$$\hat{g}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{v}, \boldsymbol{\theta}, \gamma) = \text{Tanh}\left(\sum_{h=1}^{H} w_h Z_h + \gamma\right),$$

where for each hidden node h

$$Z_h = \text{Tanh}\left(\sum_{i=1}^{n} v_{ih} x_i + \theta_h\right).$$

The parameters v_{ih}, w_h, θ_h, and γ are estimated using a nonlinear least squares approach called *training*. For simplicity, we will refer to the entire vector of parameters as $\boldsymbol{w}$ in the following sections.

5.2 Model Fitting

Our training employed a batch processing method (as opposed to the more common on-line processing) that minimized the squared error loss lack-of-fit (LOF) criterion:

$$\text{LOF}(\hat{g}) = \sum_{j=1}^{N} [y_j - \hat{g}(\boldsymbol{x}_j, \boldsymbol{w})]^2,$$

where N is the number of data points on which the network is being trained, a.k.a., the training data set. Nonlinear optimization techniques used to conduct the minimization are typically gradient descent-based methods. We used the *backpropagation method* (see Rumelhart et al. 1986, Werbos 1994), which computes the gradient of the output with respect to the outer parameters, then "backpropagates" the partial derivatives through the hidden layers, in order to compute the gradient with respect to the entire set of parameters. This is completed in two phases: (i) a forward phase in which an input $\boldsymbol{x}_j$ is presented to the network in order to generate the various outputs (i.e., the output of the network and those of the inner activation functions) corresponding to the current value of the parameters, and (ii) a backward phase in which the gradient is actually computed on the basis of the values obtained in the forward phase. After all N samples are presented to the network and the corresponding gradients are computed, the descent algorithm step is completed by upgrading the vector of parameters $\boldsymbol{w}$ using the steepest-descent method:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla \text{LOF}(\boldsymbol{w}_k),$$

where $\nabla \text{LOF}(\boldsymbol{w}_k)$ is the gradient of the LOF function with respect to the vector $\boldsymbol{w}_k$. Implementation of this approach is very simple, since the computation of the gradient requires only products, due to the mathematical structure of the network. Unfortunately, backpropagation presents many well-known disadvantages, such as entrapment in local minima. As a consequence, the choice of the stepsize α_k is crucial for acceptable convergence, and very different stepsizes may be appropriate for different applications.

The best training algorithm for FF1ANNs, in terms of rate of convergence, is the Levenberg-Marquardt method (first introduced in Hagan and Menhaj 1994), which is an approximation of

the classic second-order (i.e. it makes use of the Hessian matrix) Newton method. Since our LOF criterion is quadratic, we can approximate its Hessian matrix via backpropagation as

$$J[\boldsymbol{e}(\boldsymbol{w})]^T J[\boldsymbol{e}(\boldsymbol{w})]$$

where

$$\boldsymbol{e}(\boldsymbol{w}) = [y_1 - \hat{g}(\boldsymbol{x}_1, \boldsymbol{w}), \ldots, y_N - \hat{g}(\boldsymbol{x}_N, \boldsymbol{w})]^T.$$

and $J[\boldsymbol{e}(\boldsymbol{w})]$ is the Jacobian matrix of $\boldsymbol{e}(\boldsymbol{w})$. Each row of the matrix contains the gradient of the j-th single error term $y_j - \hat{g}(\boldsymbol{x}_j, \boldsymbol{w})$ with respect to the set of parameters $\boldsymbol{w}$.

The update rule for the parameter vector at each step is:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - [J[\boldsymbol{e}(\boldsymbol{w}_k)]^T J[\boldsymbol{e}(\boldsymbol{w}_k)] + \lambda I]^{-1} J[\boldsymbol{e}(\boldsymbol{w}_k)]^T \boldsymbol{e}(\boldsymbol{w}_k)^T.$$

When λ is large, the method is close to gradient descent with a small stepsize. When $\lambda = 0$ we have a standard Newton method with an approximated Hessian matrix (Gauss-Newton method), which is more accurate near the minima. After each successful step (i.e., when the LOF decreases) λ is decreased, while it is increased when the step degrades the LOF.

5.3 Model Implementation

Other than the selection of a model fitting approach, the most important decision in implementing an ANN model is the choice of architecture. For a FF1ANN, the number of nodes in the hidden layer, H, determines the architecture. Unfortunately, the appropriate number of hidden nodes is never clear-cut. Existing theoretical bounds on the accuracy of the approximation are non-constructive or too loose to be useful in practice. Thus, it is not possible to define "a priori" the optimal number of hidden nodes, since it depends on the number of samples available and on the smoothness of the function we want to approximate. Too few nodes would result in a network with poor approximation properties, while too many would "overfit" the training data (i.e. the network merely "memorizes" the data at the expense of generalization capability). Therefore, a tradeoff between these two possibilities is needed. There are many rules of thumb and methods for model selection and validation in the literature, as well as methods to avoid or reduce overfitting (see Haykin 1999), but the choice of a good architecture requires several modeling attempts and is always very dependent on the particular application.

For our applications, we identified the appropriate number of hidden nodes by exploring many architectures and checking the average cost using a test data set from the simulation described in Section 6.2. The best results for the nine-dimensional inventory forecasting problem were obtained for values of $H = 40$ for $N = 1331$ and $H = 60$ for $N = 2197$. Similarly, the best results for the eight-dimensional water reservoir problem were obtained using $H = 10$ for both sample sizes. For larger values of H the solutions worsened.

6 ANNs and MARS Comparisons

ANNs have become very popular in many areas of science and engineering, due to their proven ability to fit any data set and to approximate any real function, together with the existence of accessible software, such as Matlab's (www.mathworks.com) Neural Network Toolbox, which was used in our study. The primary disadvanatages are (1) the difficulty in selecting the number of hidden nodes H and (2) potential overfitting of the training data. The second disadvantage occurs because it is tempting to try to fit the data as closely as possible even when it is not

appropriate. Both disadvantages can be handled by proper use of a test data set, but this process greatly increases the computational effort required. From a statistical perspective, ANNs have the additional disadvantage of an approximation that does not provide interpretable information on the relationships between the input (predictor) variables and output (response) variable.

MARS (Friedman 1991) is based on traditional statistical thinking with a forward stepwise algorithm to select terms in a linear model followed by a backward procedure to prune the model. The approximation bends to model curvature at "knot" locations, and one of the objectives of the forward stepwise algorithm is to select appropriate knots. The search for new basis functions can be restricted to interactions of a maximum order. For the inventory forecasting problems, MARS was restricted to explore up to three-factor interations. Chen (1999) also considered MARS limited to two-factor interactions, but the approximation degraded. However, for the water reservoir problem, two-factor interactions were sufficient. In addition, the backward procedure was not utilized since preliminary testing demonstrated a very large computational burden with little change on the approximation. After selection of the basis functions is completed, smoothness to achieve a certain degree of continuity may be applied. In our case, quintic functions were used to achieve second derivative continuity (see Chen 1999). The primary disadvantage of MARS is that the user must specify the maximum number of basis functions to be added, a parameter called $M_{\max}$. This is very similar to the difficulty in selecting H for an ANN, and, like ANNs, a test data set can be used to select $M_{\max}$. MARS is both flexible and easily implemented while providing useful information on significant relationships between the predictor variables and the response; however, commerical MARS software has only recently become available from Salford Systems (www.salford-systems.com). For our study, our own C code was employed.

From the perspective of SDP value function approximation, MARS and ANNs are both excellent for modeling in high dimensions when noise is small relative to the signal, as is the case with SDP. Their main difference is the interpretability of the approximation, but this is not a critical issue for accurate SDP solutions. An issue that may arise in other SDP applications is the selection of the key parameter, $M_{\max}$ for MARS and H for ANNs. For the inventory forecasting and water reservoir problems, the value functions have similar structure from stage to stage; thus, the same $M_{\max}$ or H for all stages is appropriate. When different $M_{\max}$ or H is preferred for different stages, then the SDP algorithm would greatly benefit from automatic selection of $M_{\max}$ or H. This is an issue for future research.

6.1 Memory and Computational Requirements

The memory and computational requirements for a FF1ANN with one output variable depend on the number of input (predictor) variables n, the number of hidden nodes H, and the size of the training data set N. For MARS, the key quantities are n, N, the number of eligible knots K, and the maximum number of basis functions $M_{\max}$. In this section, we will explore how the requirements might grow with increasing n (the dimension of an SDP problem). Chen et al. (1999) set $N = O(n^3)$, which assumes N is based on a strength three OA, $K = O(n)$, which assumes knots can only placed at the p distinct levels of the strength three OA, and $O(n) \leq M_{\max} \leq O(n^3)$. In this paper, the addition of the OA-LH designs permits the placement of a higher number of knots. However, given an adequate K, it should not be expected to increase with n since the larger K primarily impacts the quality of univariate modeling. Based on computational experience with a variety of value function forms, $K \leq 50$ is more than adequate, and a more reasonable upper limit on $M_{\max}$ is $O(N^{2/3}) = O(n^2)$. For ANNs, H depends on n, N, and the complexity of the underlying function. A reasonable range on H (in terms of growth) matches that of $O(n) \leq H \leq O(N^{2/3}) = O(n^2)$. One should keep in mind that the above estimates are intended to be adequate for approximating

SDP value functions, which are convex and, consequently, generally well behaved. The inventory forecasting value function, however, does involve a steep curve due to high backorder costs.

For memory requirements, a FF1ANN requires storage of $V = [(n+1)H + H + 1]$ parameters. Assuming each of these is a double-precision real number, the total number of bytes required to store a FF1ANN grows as

$$O(n^2) \leq S_{FF1ANN} = 8V \leq O(n^3).$$

The maximum total number of bytes required to store one smoothed MARS approximation with up to three-factor interactions grows as (Chen et al. 1999)

$$O(n) \leq S_{\text{MARS}} = 8(6M^*_{\max} + 2n + 3Kn + 1) \leq O(n^2).$$

Thus, it appears that the memory requirement for FF1ANNs grows faster than that for MARS. However, the largest FF1ANN tested in the paper required 5288 bytes and the largest MARS required 36,800 bytes, so for $n = 9$, FF1ANNs are still considerably smaller.

Another aspect of memory requirements is memory required while constructing the approximation. We utilized Levenberg-Marquardt (LM) backpropagation to train the FF1ANN, and a disadvantage of the LM method lies in its memory requirements. Each step requires the storage of a $(N \times V)$ matrix and the $(V \times V)$ Hessian approximation, so the memory needed grows as

$$O(n^5) \leq S2_{FF1ANN} = O(NnH) + O(n^2H^2) \leq O(n^6).$$

However, if the memory at disposal is sufficient, LM backpropagation is generally preferred over other training methods because of its faster convergence. MARS construction stores up to a $(N \times M_{\max})$ matrix, resulting in a growth of

$$O(n^4) \leq S2_{MARS} = O(NM_{\max}) \leq O(n^5).$$

The computational requirements for one iteration of our SDP statistical modeling solution approach consist of (i) solving the optimization in equation (3) for one stage t at every experimental design point and (ii) constructing the spline approximation of the future value function. The optimization evaluates a value function approximation and its first derivative several times in each stage, except the last. For FF1ANNs, (i) is performed via optimization that requires the evaluation of the output of an ANN and of its first derivative. The time required for these evaluations is $O(nH)$, so we can write the total time required to solve the SDP equation for all N points as $O(WnNH)$, where W is the work required to solve the minimization.

For what concerns (ii), we consider Levenberg-Marquardt backpropagation, which is the one we actually used for our FF1ANNs. The time for the computation of LM backpropagation is dominated by the Hessian approximation and inversion. The $(V \times V)$ Hessian approximation is the product of two $(N \times V)$ matrices that are obtained by backpropagation. Here, for each row of the matrix (which corresponds to a single experimental design point $\boldsymbol{x}_j$), the computation is basically reduced to $[H + 1 + (n+1)H]$ products, so the time required to obtain the gradient for all N experimental design points is $O(NnH)$. The matrix product requires NV^2 real valued products, so this operation (which is $O(Nn^2H^2)$) occupies the relevant part of the time required to compute the approximation. For the inversion of a $(V \times V)$ matrix, an efficient algorithm can provide a computational time that is $O(V^2) = O(n^2H^2)$. Therefore, the time needed to compute one step of LM backpropagation is dominated by the matrix product: $O(Nn^2H^2)$. Define W_{LM} to be the number of steps required to attain the backpropagation goal. (Typically, W_{LM} never needs to exceed 300.) Then the time required to train the network by LM backpropagation grows

as $O(W_{LM}Nn^2H^2)$. Thus, the total run time for one iteration of the FF1ANN-based SDP solution grows as

$$O(n^7) \leq C_{\text{FF1ANN}} = O(WNnH) + [O(W_{\text{LM}}n^2NH^2)] \leq O(n^9).$$

The total run time for one iteration of the MARS-based SDP solution grows as (Chen et al. 1999)

$$O(n^7) \leq C_{\text{MARS}} = O(WNM_{\max}) + O(nNM_{\max}^3) \leq O(n^{10}),$$

where the time to construct the MARS approximation (second term) dominates.

Actual average run times for one SDP iteration are shown in Table 1. For the inventory forecasting problem, the MARS and ANN runs were conducted on different machines in different countries. Despite this, it can been seen that the run times for the same N are comparable. In order to conduct a fairer computational comparison on the same programming platform, considerable effort was expended to develop a Matlab code that could access our MARS C code. Unfortunately, the ability to call C code from Matlab is still flawed, and the MARS runs suffered significant slow down. For the water reservoir problem, all runs were conducted in the same place; however, the slow down experienced by calling the MARS C code from Matlab is obvious. Unfortunately, at this time, a completely fair comparison that uses the same programming platform is not readily implementable. Overall, our experience demonstrates that the ANN model is clearly easier to implement.

6.2 SDP Solutions

Both the nine-dimensional inventory forecasting problem and the eight-dimensional water reservoir problem were solved over three stages ($T = 3$). Although several FF1ANN architectures were tested, only one FF1ANN architecture for each design was employed to generate the plots in Figures 3 through 8. For the inventory forecasting problem, $H = 40$ for $N = 1331$ and $H = 60$ for $N = 2197$, and for the water reservoir problem, $H = 10$ for both N. Several MARS approximations were constructed for each design by varying $M_{\max}$ and K. For the inventory forecasting problem, results with different $M_{\max}$ are shown, and for the water reservoir problem, $M_{\max} = 100$. For the OAs we set $K = 9$ for $p = 11$ and $K = 11$ for $p = 13$ (the maximum allowable in both cases). For the OA-LH designs K was chosen as high as 50. Simulations using the "reoptimizing policy" described in Chen (1999) were conducted to determine the optimal decisions and to test the SDP solutions.

6.2.1 Inventory Forecasting Problem

For each of 100 randomly chosen initial state vectors, a simulation of 1000 demand forecast $\boldsymbol{\epsilon}$ sequences was conducted for the SDP solutions. The simulation output provided the 100 mean costs (corresponding to the 100 initial state vectors), each over 1000 sequences. For each initial state vector, the smallest mean cost achieved by a set of 90 SDP solutions was used as the "true" mean cost for computing the "errors" in the mean costs. Boxplots in Figures 3 through 6 display the distributions (over the 100 initial state vectors) of these errors. The average over the 100 "true" mean costs was approximately 62.

Figures 3 and 4 display all 18 of the OA-based SDP solutions. They are organized left to right by the "goodness" of the OA design. The OA/MARS solutions with $p = 11$ ($N = 1331$) are all fairly close, except for the "worst" one, which is "MARS OA3P p11 300," i.e., the one that used the poor OA and $M_{\max} = 300$. The other MARS OA3P solution (with $M_{\max} = 200$) is used to represent the "best" OA/MARS solution, although it is essentially equivalent to the other good OA/MARS solutions. For the OA/FF1ANN solutions with $p = 11$ ($N = 1331$), the "worst" one is "ANN

OA3N p11 H40," i.e., the one that used the neutral OA3 (there is actually another outlier above 15). The "best" one is "ANN OA3P p11 H40," using the poor OA. The OA/FF1ANN solution with the good OA is not terrible, but is worst than most of the OA/MARS solutions because of its higher median. Ideally, we would want good OAs to result in better solutions than poor OAs; however, sometimes it does not happen this way because the poor qualities are located in regions of the state space that are not critical to finding a good solution.

For the MARS solutions with $p = 13$ ($N = 2197$), the "best" solution is "MARS OA3G p13 400," which used the good OA and $M_{\max} = 400$; while the "worst" is "MARS OA3P p13 400," which used the poor OA and $M_{\max} = 400$. For the FF1ANN solutions with $p = 13$ ($N = 2197$), the "best" solution is with the good OA, while the "worst" is with the neutral OA. It should be noted in Figures 3 and 4, as was illustrated in Chen (1999, 2001), that good OAs result in better SDP solutions on average.

Figures 5 and 6 display the best and worst solutions for each of the four types of SDP solutions, with the best solutions on the left and worst on the right. The OA/FF1ANN and OA/MARS solutions are repeated from Figures 3 and 4. Note the solution for "ANN LH1N N1331 H40," which used the first OA-LH realization from the neutral OA. This is the "worst" solution among the FF1ANNs using OA-LH designs, but it is actually a good solution. The interesting observation here is that the FF1ANNs performed better with the OA-LH designs. The best solution overall is "MARS LH2G N2197 400 K11," which used the second OA-LH realization from the good OA, $M_{\max} = 400$, and $K = 11$. However, there is one disconcerting solution, and that is "MARS LH2G N1331 300 K35." It is troubling because it uses a LH design generated from the good OA, but it is visibly one of the "worst" solutions. To some extent the goodness measures are no longer valid for the OA-LH designs, because the randomization used in generating them may have resulted in a poorly spaced realization.

6.2.2 Water Reservoir Problem

For each of 50 randomly chosen initial state vectors, a simulation of 100 demand forecast $\boldsymbol{\epsilon}$ sequences was conducted for the SDP solutions. The simulation output provided the 50 mean costs (corresponding to the 50 initial state vectors), each over 100 sequences. The "errors" in mean costs were computed in the same manner as the inventory forecasting problem. Boxplots in Figures 7 and 8 display the distributions (over the 50 initial state vectors) of these errors. The average over the 50 "true" mean costs was approximately -47.

Figure 7 displays all the OA-based SDP solutions. They are organized left to right by OA design size and by the space-filling quality of the OA design. Among the SDP solutions with $p = 11$ ($N = 1331$) the "worst" one is clearly "ANN OA3P p11 H=10." The "best" one is not as easy to identify, but "ANN OA3N p11 H=10" is seen to have the lowest median. Among the SDP solutions with $p = 13$ ($N = 2197$) the "worst" one, having a significantly longer box, is "MARS OA3P p13 100." All the other solutions for this size design are somewhat comparable, but "MARS OA3N p13 100" is seen to have the lowest median. For both sizes, the worst solutions employed the poor OA, indicating that poor solutions can be avoided by not using a poor OA. This agrees with the findings in the previous section.

Figure 8 displays the best and worst solutions from Figure 7 along with the best and worst OA-LH/FF1ANN and OA-LH/MARS solutions. The figure is organized with the with the best solutions on the left and worst on the right. Some improvement is seen using the OA-LH designs. Unlike the inventory forecasting problem, the water reservoir problem should not require many interaction terms in the value function approximation; thus, the increased univariate information from LH designs should be an advantage. Like the findings in the previous section, the worst OA-

LH/FF1ANN solutions are not bad, supporting the expectation that ANN models perform well with OA-LH designs.

7 Concluding Remarks

The statistical modeling approach for numerically solving continuous-state SDP was used to simultaneously study OA-LH designs and an ANN-based implementation as alternatives to the OA/MARS method of Chen et al. (1999) and Chen (1999). In addition, the space-filling quality of the OA designs was considered. For the SDP solutions using OA designs, good OAs reduced the possibility of a poor SDP solution. For the inventory forecasting problem, involving significant interaction terms, MARS performed similarly using both OA and OA-LH designs, while FF1ANN solutions noticeably improved with OA-LH designs. For the water reservoir problem, Overall, good solutions could be achieved using MARS models with either good OA or OA-LH designs and FF1ANN models with good OA-LH designs. With regard to both computational SDP requirements and quality of SDP solutions, we judged FF1ANNs and MARS to be very competitive with each other overall. A small advantage to MARS is the information on the relationships between the future value function and the state variables that may be gleaned from the fitted MARS model. Current research is developing a method to automatically select $M_{\max}$ for MARS, so as to permit flexiblity in the approximation across different SDP stages. Unfortunately, a method for automatically selecting H for ANNs does not seem possible.

References

Abdelaziz, A. Y., M. R. Irving, M. M. Mansour, A. M. El-Arabaty, A. I. Nosseir (1997). "Neural network-based adaptive out-of-step protection strategy for electrical power systems." *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, **5(1)**, pp. 35–42.

Allen, T., M. A. Bernshteyn, and K. Kabiri (2003). "Constructing Meta-Models for Computer Experiments." *Journal of Quality Technology*, **35**, pp. 264–274.

Archibald, T. W., K. I. M. McKinnon and L. C., Thomas (1997). "An aggregate stochastic dynamic programming model of multireservoir Systems. " *Water Resources Research*, **33**, pp. 333–340.

Baglietto, M., C. Cervellera, T. Parisini, M. Sanguineti, and R. Zoppoli (2001). "Approximating Networks for the Solution of T-stage Stochastic Optimal Control Problems." *Proceedings of the IFAC Workshop on Adaptation and Learning in Control and Signal Processing.*

Baglietto, M., C. Cervellera, M. Sanguineti, R. Zoppoli (2005). "Water reservoirs management under uncertainty by approximating networks and learning from data." In *Topics on Participatory Planning and Managing Water Systems*, Elsevier Science Ltd., to appear.

Barron, A. R. (1993). "Universal Approximation Bounds for Superpositions of a Sigmoidal Function." *IEEE Trans. on Inf. Theory*, **39**, pp. 930–945

Barron, A. R., R. L. Barron, and E. J. Wegman (1992). "Statistical Learning Networks: a Unifying View." *Computer Science and Statistics: Proceedings of the 20th Symposium on the Interface*, pp. 192–203.

Bellman, R. E. (1957). *Dynamic Programming.* Princeton: Princeton University Press.

Bertsekas, D. P. and J. N. Tsitsiklis (1996). *Neuro-Dynamic Programming.* Belmont, Massachusetts: Athena Scientific.

Bertsekas, D. P. and J. N. Tsitsiklis (2000). *Dynamic Programming and Optimal Control.* Belmont, Massachusetts: Athena Scientific.

Bose, R. C. and K. A. Bush (1952). "Orthogonal Arrays of Strength Two and Three." *Annals of Mathematical Statistics*, **23**, pp. 508–524.

Carey, W. P. and S. S. Yee (1992). "Calibration of nonlinear solid-state sensor arrays using multivariate regression techniques." *Sensors and Actuators, B: Chemical*, **B9(2)**, pp. 113–122.

Chen, V. C. P. (1999). "Application of MARS and Orthogonal Arrays to Inventory Forecasting Stochastic Dynamic Programs." *Computational Statistics and Data Analysis*, **30**, pp. 317–341.

Chen, V. C. P. (2001). "Measuring the Goodness of Orthogonal Array Discretizations for Stochastic Programming and Stochastic Dynamic Programming." *SIAM Journal of Optimization*, **12**, pp. 322–344.

Chen, V. C. P., J. Chen, and M. B. Beck (2000). "Statistical Learning within a Decision-Making Framework for More Sustainable Urban Environments." In *Proceedings of the Joint Research Conference on Statistics in Quality, Industry, and Technology*, Seattle, Washington, June 2000.

Chen, V. C. P., D. Günther and E. L. Johnson (2002). "Solving for an Optimal Airline Yield Management Policy via Statistical Learning." *Journal the Royal Statistical Society, Series C*, to appear.

Chen, V. C. P. and D. K. Rollins (2000). "Issues Regarding Artificial Neural Network Modeling for Reactors and Fermenters." *Bioprocess Engineering*, **22**, pp. 85–93.

Chen, V. C. P., D. Ruppert, and C. A. Shoemaker (1999). "Applying Experimental Design and Regression Splines to High-Dimensional Continuous-State Stochastic Dynamic Programming." *Operations Research*, **47**, pp. 38–53.

Chen, V. C. P., K.-L. Tsui, R. R. Barton, and J. K. Allen (2003). "A Review of Design and Modeling in Computer Experiments." In *Handbook of Statistics: Statistics in Industry* (R. Khattree and C. R. Rao, eds.), **22**, Amsterdam: Elsevier Science, pp. 231–261.

Chen, V. C. P. and W. J. Welch (2001). "Statistical Methods for Deterministic Biomathematical Models." In *Proceedings of the 53rd Session of the International Statistical Institute*, Seoul, Korea, August 2001.

Cheng, B. and D. M. Titterington (1994). "Neural Networks: a Review from a Statistical Perspective (with discussion)." *Statistical Science*, **9**, pp. 2–54.

Cherkassky, V. and F. Mulier (1998). *Learning from Data: Concepts, Theory and Methods*. New York: Wiley.

Culver, T. B. and C. A. Shoemaker (1997). "Dynamic Optimal Ground-Water Reclamation with Treatment Capital Costs." *Journal of Water Resources Planning and Management*, **123**, pp. 23–29.

De Veaux, R. D. D. C. Psichogios, and L. H. Ungar (1993). "Comparison of two nonparametric estimation schemes: MARS and neural networks." *Computers & Chemical Engineering*, **17(8)**, pp. 819–837.

Dey, A. (1985). *Orthogonal Fractional Factorial Designs*. New York: John Wiley & Sons.

Eschenbach, E. A., C. A. Shoemaker, and H. Caffey (1995). "Parallel Processing of Stochastic Dynamic Programming for Continuous State Systems with Linear Interpolation." *ORSA Journal on Computing*, **7**, pp. 386–401.

Foufoula-Georgiou, E. and P. K. Kitanidis (1988). "Gradient Dynamic Programming for Stochastic Optimal Control of Multidimensional Water Resources Systems." *Water Resources Research*, **24**, pp. 1345–1359.

Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines (with discussion)." *Annals of Statistics*, **19**, pp. 1–141.

Gal, S. (1979). "Optimal Management of a Multireservoir Water Supply System." *Water Resources Research*, **15**, pp. 737–748.

Griffin, W. L. N. I. Fisher, J. H. Friedman, C. G. Ryan (1997). "Statistical techniques for the classification of chromites in diamond exploration samples." *Journal of Geochemical Exploration*, **59(3)**, pp. 233–249.

Hadley, G. and T. M. Whitin (1963). *Analysis of Inventory Systems*. Englewood Cliffs, NJ: Prentice-Hall.

Hagan, M. T. and M. Menhaj (1994). "Training Feedforward Networks with the Marquardt Algorithm." *IEEE Trans. on Neur. Networks*, **5**, pp. 989–993.

Haykin, S. S. (1999). *Neural Networks: A Comprehensive Foundation*, second edition. Upper Saddle River, NJ: Prentice Hall.

Heath, D. C. and P. L. Jackson (1991). "Modelling the Evolution of Demand Forecasts with Application to Safety Stock Analysis in Production/Distribution Systems." Technical Report #989, School of Operations Research & Industrial Engineering, Cornell University, Ithaca, NY.

Hedayat, A. S. and W. D. Wallis (1978). "Hadamard Matrics and their Applications." *Annnals of Statistics*, **6**, pp. 1184–1238.

Hedayat, A. S., J. Stufken, and G. Su (1996). "On Difference Schemes and Orthogonal Arrays of Strength t." *Journal of Statistical Planning and Inference*, **56**, pp. 307–324.

Johnson, S. A., J. R. Stedinger, C. A. Shoemaker, Y. Li, and J. A. Tejada-Guibert (1993). "Numerical Solution of Continuous-State Dynamic Programs Using Linear and Spline Interpolation." *Operations Research*, **41**, pp. 484–500.

Kottapalli, S. (2002). "Neural network based representation of UH-60A pilot and hub accelerations." *Journal of the American Helicopter Society*, **47(1)**, pp. 33–41.

Kuhnert, P. M. K.-A. Do, and R. McClure (2000). "Combining non-parametric models with logistic regression: An application to motor vehicle injury data." *Computational Statistics and Data Analysis*, **34(3)**, pp. 371–386.

Labossiere, P. and N. Turkkan (1993). "Failure prediction of fibre-reinforced materials with neural networks." *Journal of Reinforced Plastics and Composites*, **12(12)**, pp. 1270–1280.

Lamond, B. F., and M. J. Sobel, (1995). "Exact and approximate solutions of affine reservoirs models." *Operations Research*, **43**, pp. 771–780.

Li, Q. S., D. K. Liu, J. Q. Fang, A. P. Jeary, and C. K. Wong (2000). "Damping in buildings: Its neural network model and AR model." *Engineering Structures*, **22(9)**, pp. 1216–1223.

Lippmann, R. P. (1987). "An Introduction to Computing with Neural Nets." *IEEE ASSP Magazine*, pp. 4–22.

Liu, J. (2001). "Prediction of the molecular weight for a vinylidene chloride/vinyl chloride resin during shear-related thermal degradation in air by using a back-propagation artificial neural network." *Industrial and Engineering Chemistry Research*, **40(24)**, pp. 5719–5723.

Palmer, K. D. (1998). *Data Collection Plans and Meta Models for Chemical Process Flowsheet Simulators*. Ph.D. dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, U.S.A.

Pham, D. T. and B. J. Peat (1999). "Automatic learning using neural networks and adaptive regression." *Measurement and Control*, **32(9)**, pp. 270–274.

Pigram, G. M. and T. R. Macdonald (2001). "Use of neural network models to predict industrial bioreactor effluent quality." *Environmental Science and Technology*, **35(1)**, pp. 157–162.

Puterman, M. L. (1994). *Markov Decision Processes*. New York, NY: John Wiley & Sons, Inc.

Raghavarao, D. (1971). *Constructions and Combinatorial Problems in Design of Experiments*. New York: John Wiley & Sons.

Rao, C. R. (1946). "Hypercubes of Strength '*d*' Leading to Confounded Designs in Factorial Experiments." *Bulletin of the Calcutta Mathematical Society*, **38**, pp. 67–78.

Ripley, B. D. (1993). "Statistical Aspects of Neural Networks." In *Networks and Chaos - Statistical and Probabilistic Aspects* (O. E. Barndorff-Nielsen, J. L. Jensen, and W. S. Kendall, eds.). New York: Chapman & Hall, pp. 40–123.

Rumelhart, D. E. , G. E. Hinton and R. J. Williams (1986). "Learning Internal Representations by Error Propagation." In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart and J. L. McClelland, eds.). Cambridge, MA: MIT Press, pp. 318–362.

Shepherd, K. D. and M. G. Walsh (2002). "Development of reflectance spectral libraries for characterization of soil properties." *Soil Science Society of America Journal*, **66(3)**, pp. 988–998.

Shoemaker, C. A. (1982). "Optimal Integrated Control of Univoltine Pest Populations with Age Structure," *Operations Research*, **30**, pp. 40–61.

Simpson, T. W., D. K. J. Lin, and W. Chen (2001). "Sampling Strategies for Computer Experiments: Design and Analysis." *International Journal of Reliability and Applications*, **2(3)**, pp. 209–240.

Smets, H. M. G. and W. F. L. Bogaerts (1992). "Deriving corrosion knowledge from case histories: The neural network approach." *Materials & Design*, **13(3)**, pp. 149–153.

Tang, B. (1993). "Orthogonal Array-Based Latin Hypercubes." *Journal of the American Statistical Association*, **88**, pp. 1392–1397.

Tsai, J. C. C., V. C. P. Chen, J. Chen, and M. B. Beck (2004). "Stochastic Dynamic Programming Formulation for a Wastewater Treatment Decision-Making Framework." *Annals of Operations Research*, Special Issue on Applied Optimization Under Uncertainty, **132**, pp. 207–221.

Turgeon, A. (1981). "A decomposition method for the long-term scheduling of reservoirs in series." *Water Resources Research*, **17**, pp. 1565–1570.

Werbos, P. K. (1994). *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. New York: Wiley.

White, D. J. (1985). "Real Applications of Markov Decision Processes." *Interfaces*, **15(6)**, pp. 73–83.

White, D. J. (1988). "Further Real Applications of Markov Decision Processes." *Interfaces*, **18(5)**, pp. 55–61.

White, H. (1989). "Learning in Neural Networks: a Statistical Perspective." *Neural Computation*, **1**, pp. 425–464.

Yakowitz, S. (1982). "Dynamic programming applications in water resources." *Water Resources Research*, **18**, pp. 673–696.

Table 1: Actual average run times (in minutes) for one SDP iteration using FF1ANN and MARS models. The FF1ANN runs used $W_{\text{LM}} = 125$. For the inventory forecasting problem, Matlab was used to conduct the FF1ANN runs on two Pentium II (P2) machines, and a C code executed the MARS runs on a different P2 machine. For the water reservoir problem, all runs were conducted in Matlab on an Intel Xeon (IX) workstation; however, calling the MARS C code in Matlab resulted in significant slow down.

		Inventory Forecasting			**Water Reservoir**
Model	N	**450MHz P2**	**550MHz P2**	**933MHz P2**	**3.06GHz IX**
ANN	1331	65m ($H = 40$)		90m ($H = 40$)	13m ($H = 10$)
MARS	1331		61m ($M_{\max} = 300$)		43m ($M_{\max} = 100$)
ANN	2197	185m ($H = 60$)		87m ($H = 60$)	18m ($H = 10$)
MARS	2197		112m ($M_{\max} = 400$)		67m ($M_{\max} = 100$)

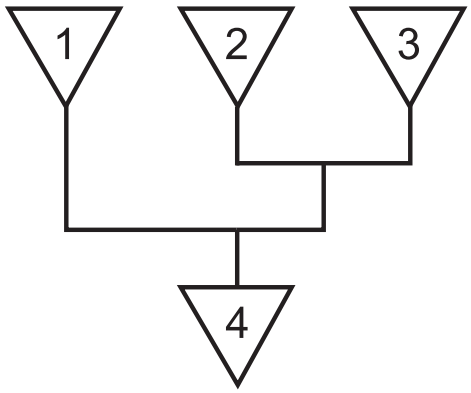

Figure 1: Reservior network

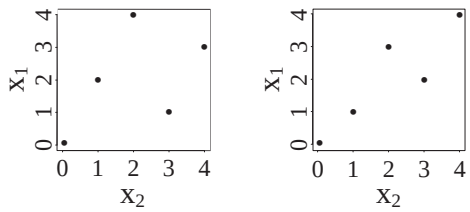

Figure 2: Two configurations of design points.

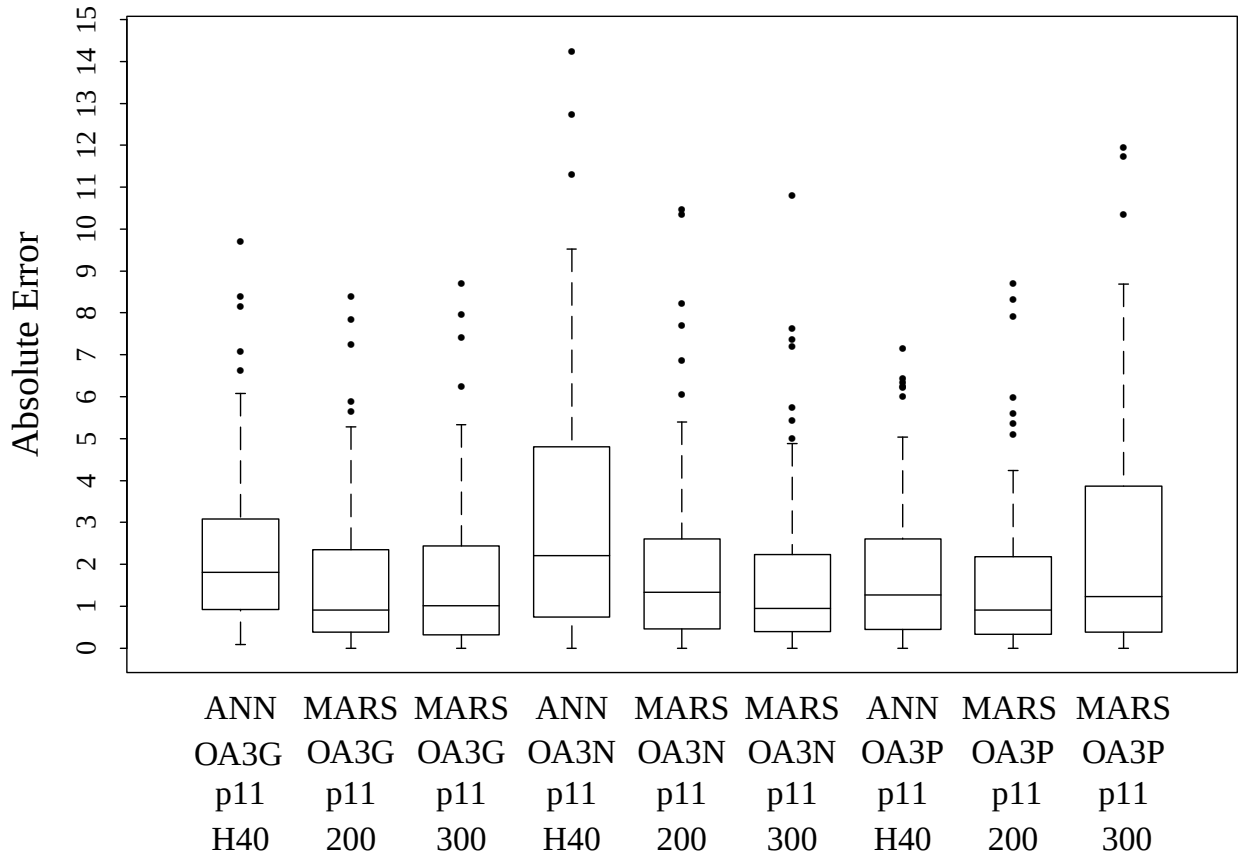

Figure 3: Inventory Forecasting: Boxplots illustrate the distribution of absolute error for all 9 SDP solutions using strength 3 OAs with $p = 11$ ($N = 1331$). Below each boxplot, the first line indicates the use of ANNs or MARS. The second line indicates the type of OA, specifically: OA3G = "good" strength 3 OA, OA3N = "neutral" strength 3 OA, and OA3P = "poor" strength 3 OA. The third line specifies $p = 11$, and the fourth line provides $M_{\max}$ for MARS and the number of hidden nodes (H) for ANNs.

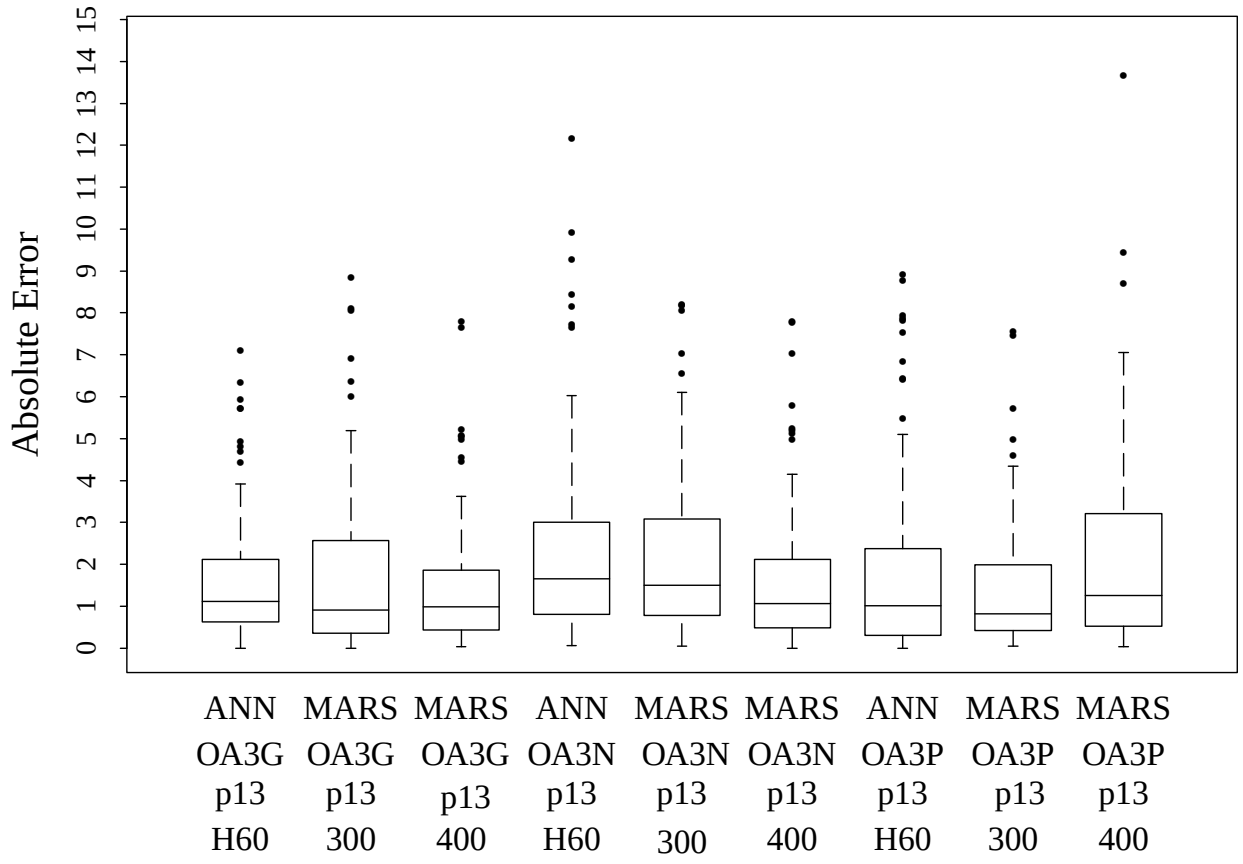

Figure 4: Inventory Forecasting: Boxplots illustrate the distribution of absolute error for all 9 SDP solutions using strength 3 OAs with $p = 13$ ($N = 2197$). Labeling below the boxplots uses the same notataion as that in Figure 3.

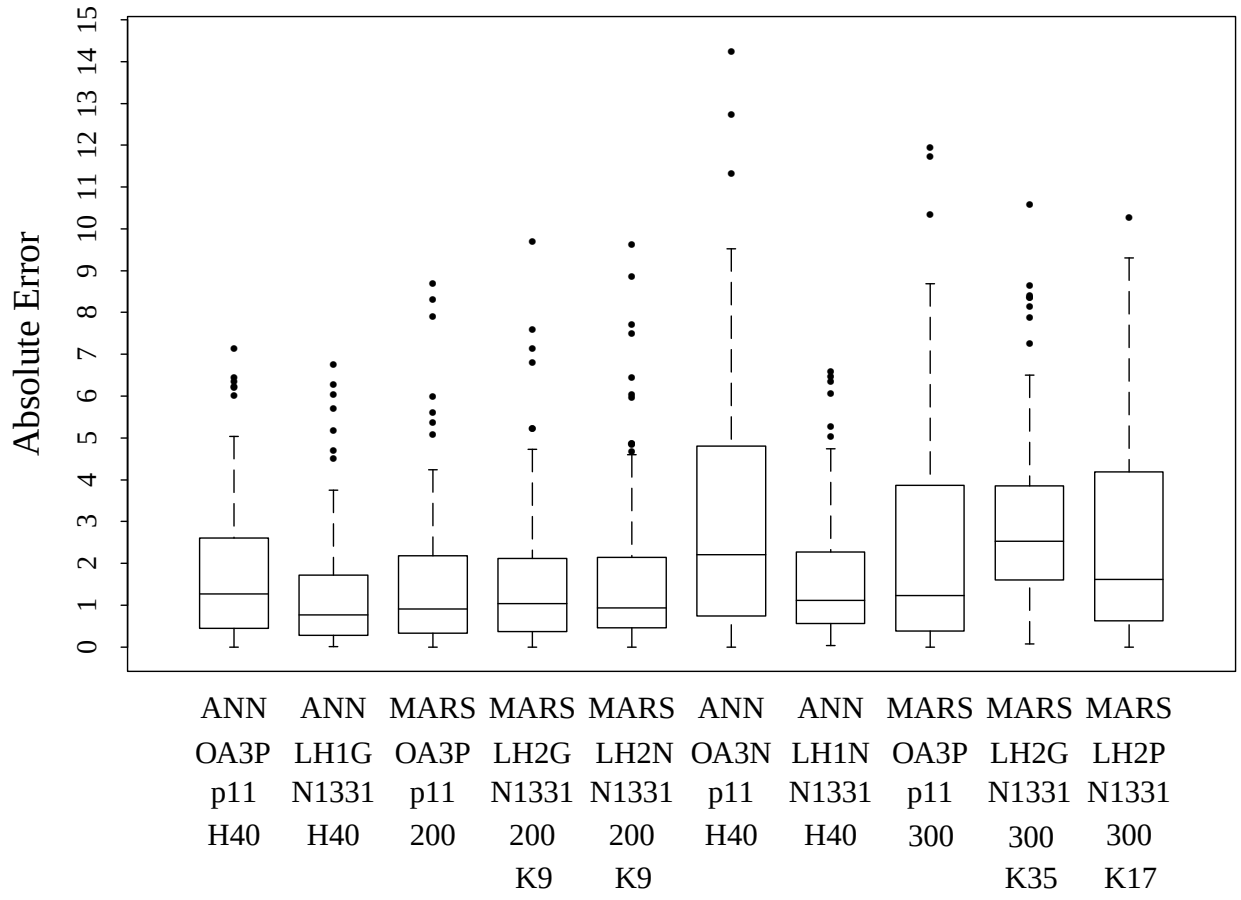

Figure 5: Inventory Forecasting: Boxplots illustrate the distribution of absolute error for 10 SDP solutions using $N = 1331$. Included from Figure 3 are the best (ANN/OA3P; MARS/OA3P/$M_{\max} = 200$) and worst (ANN/OA3N; MARS/OA3P/$M_{\max} = 300$) solutions. Below each boxplot, the first line indicates the use of ANNs or MARS. The second line indicates the type of experimental design, specifically: OA3 notation as in Figure 3, LH1G = first Latin hypercube randomization based on a "good" strength 3 OA, LH2G = second Latin hypercube randomization based on a "good" strength 3 OA, and similarly for LH1N, LH2N, LH1P, and LH2P. The third line indicates the size of the design, specifying $p = 11$ for the OAs and $N = 1331$ for the OA-LH designs. The fourth line provides $M_{\max}$ for MARS and the number of hidden nodes (H) for ANNs. Finally, a fifth line appears for the OA-LH/MARS solutions indicating the number of eligible knots (K) utilized in the MARS algorithm. The three best OA-LH-based solutions are shown beside the best OA-based solutions (in the left half), and similarly for the worst solutions (in the right half).

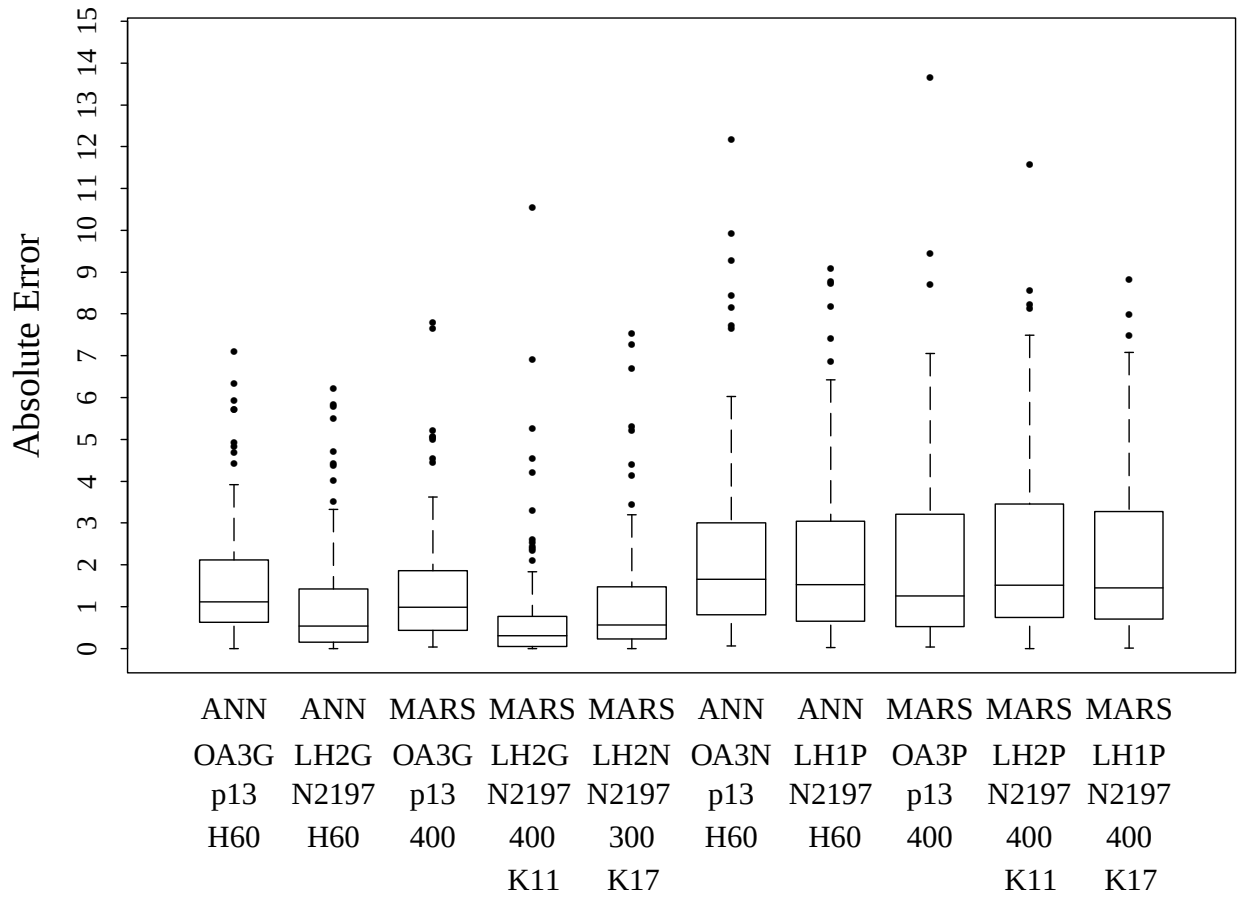

Figure 6: Inventory Forecasting: Boxplots illustrate the distribution of absolute error for 10 SDP solutions using $N = 2197$ experimental design points. Included from Figure 4 are the best (ANN/OA3G, MARS/OA3G/$M_{\max} = 400$) and worst (ANN/OA3N, MARS/OA3P/$M_{\max} = 400$ solutions. Labeling below the boxplots uses the same notation as that in Figure 5.

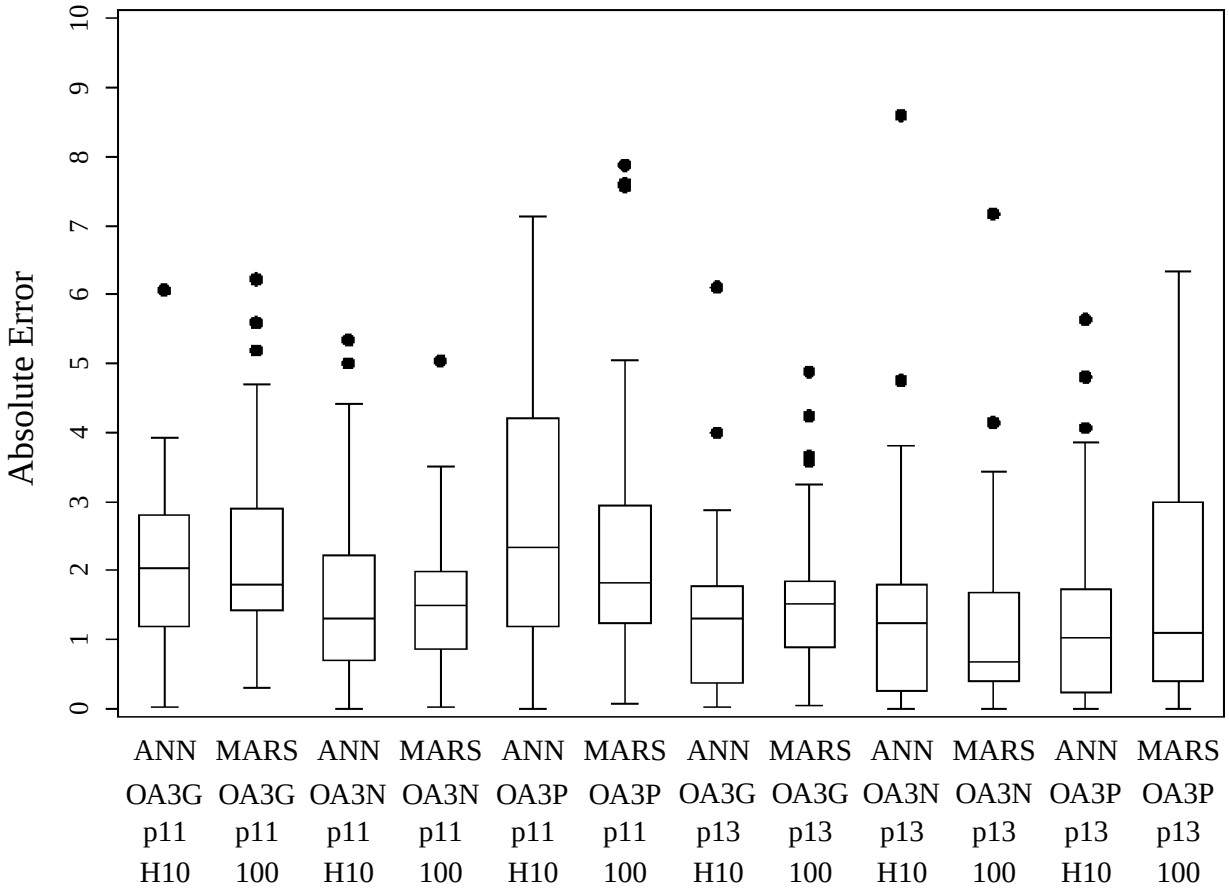

Figure 7: Water Reservoir: Boxplots illustrate the distribution of absolute error for all 12 SDP solutions using strength 3 OAs with $p = 11$ ($N = 1331$) and $p = 12$ ($N = 2197$). Labeling below the boxplots uses the same notation as that in Figure 3.

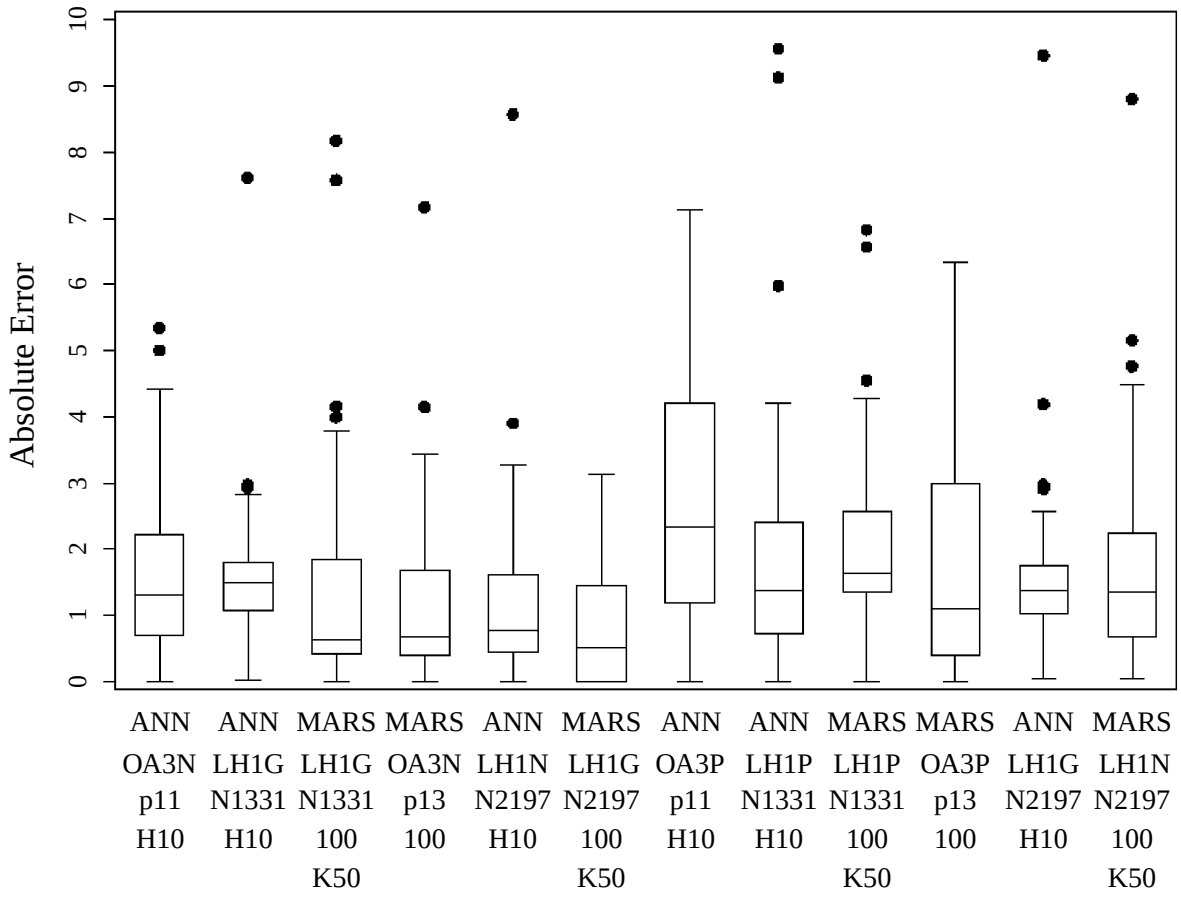

Figure 8: Water Reservoir: Boxplots illustrate the distribution of absolute error for 12 SDP solutions. Included from Figure 7 are the best (ANN/OA3N/p11, MARS/OA3N/p13) and worst (ANN/OA3P/p11, MARS/OA3P/p13) solutions. Labeling below the boxplots uses the same notation as that in Figure 5.