Intelligent Supply Chain Management Using Adaptive Critic Learning

Stephen Shervais, *Member, IEEE*, Thaddeus T. Shannon, *Student Member, IEEE*, and George G. Lendaris, *Fellow, IEEE*

***Abstract*—A set of neural networks is employed to develop control policies that are better than fixed, theoretically optimal policies, when applied to a combined physical inventory and distribution system in a nonstationary demand environment. Specifically, we show that model-based adaptive critic approximate dynamic programming techniques can be used with systems characterized by discrete valued states and controls. The control policies embodied by the trained neural networks outperformed the best, fixed policies (found by either linear programming or genetic algorithms) in a high-penalty cost environment with time-varying demand.**

***Index Terms*—Adaptive critics, approximate dynamic programming, artificial neural networks, dual heuristic programming, genetic algorithms, supply chain management.**

I. Introduction

PHYSICAL distribution systems enable an organization to meet final demand for goods it obtains from outside suppliers and to which it may or may not add value by processing. In our test system (Fig. 1), the supply chain is a multiechelon depot system. Each depot holds inventory (at some cost) and orders replenishment from the next echelon up; the highest node is an outside producer/supplier. Holding costs are lower at higher echelon storage sites. Deliveries are via the most economical transportation mode consistent with the demand and reorder cycle. Priority replenishment may be made via a high-speed, high-cost mode, and backorders are allowed. Transportation resources are multimodal and under policy control, but are limited and can only be changed periodically. The objective is to optimize the inventory, ordering, and transport policies with regard to some financial criterion; in our case the criterion is to minimize total cost. Final demand may fluctuate, and average demand can change over time.

Most prior studies in this application area have concentrated only on inventory allocations (economic order quantity) [7], [9], or only on transportation allocations (solid transportation problem) [1], [5]. For example, out of forty-six papers consulted during our research, only ten dealt with the combined inventory/transport problem, and none of the work we reviewed dealt with the combined inventory/transport problem in a stochastic, nonstationary environment. A 1999 review by Sarmiento and

Manuscript received November 16, 2001; revised May 13, 2001. This work is supported by the National Science Foundation under Grant ECS-9904378. This paper was recomended by Associate Editor S. E. Shimony.

S. Shervais is with the Eastern Washington University, Cheney, WA 99004 USA (e-mail: sshervais@ewu.edu).

T. T. Shannon and G. G. Lendaris are with the Portland State Univesity, Portland, OR 97201 USA.

Digital Object Identifier 10.1109/TSMCA.2003.809214

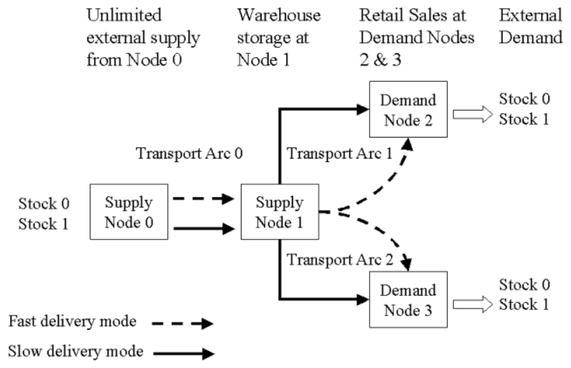


Fig. 1. Physical distribution problem. Inventory is generated at node 0 (a supply node), moved to warehouse supply node 1 using multi-mode, but limited, transport capacity, then shipped as needed to meet stocking requirements and fulfill external demand at retail demand nodes 2 and 3.

Nagi [21] lists no papers in this area, and identified the problem as one that needs additional research. The investigation of nonstationary demand appeared only in the thirteen papers that dealt solely with the inventory problem.

This work addresses the selection of an optimal set of **both** transport and inventory policies for a multiproduct, multi-echelon, multimodal physical distribution system in a nonstationary environment.

The problem is highly multidimensional, even with a small system. Both state and control variables are discrete valued and end-use demand is best characterized by a random variable. The cost surface in policy space for such systems tends to be quite discontinuous, with low cost and high cost regions separated by no more than a single transport unit.

Adaptive critic methods are a kind of reinforcement learning, wherein the parameters of a controller are periodically modified in a principled manner to effect reduction of estimated long-term cost. Most previous applications of adaptive critic methodologies have been in the context of continuous dynamic systems. Recent examples include applications such as an autopilot design for agile missile control [8] and an autonomous steering and speed control system for an automobile [11]. This paper demonstrates the use of adaptive critic based approximate dynamic programming techniques to the simultaneous optimization of inventory control and transportation policies for a discrete event physical distribution system operating within a changing business environment.

The Adaptive Critic method used here is known as dual heuristic programming (DHP). In the classic inventory/transport problem, the physical distribution system (Item A. in Fig. 2.) operates in a static environment with fixed policies. To cope with a stochastic, nonstationary environment, the system

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

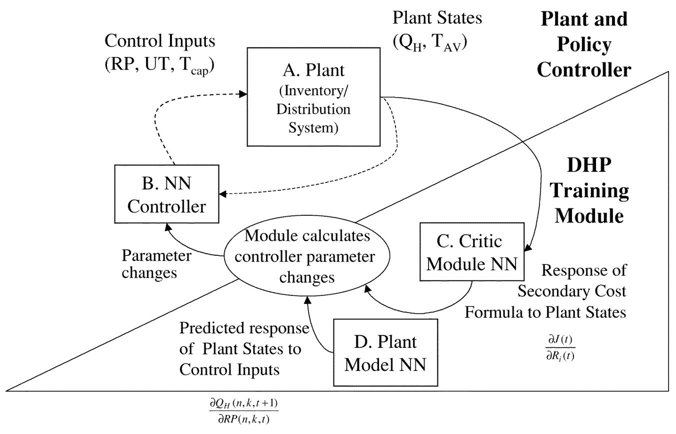

Plant States
(Q_H, T_{AV})
Plant and Policy Controller
Control Inputs
(RP, UT, T_{cap})
A. Plant (Inventory/ Distribution System)
B. NN Controller
DHP Training Module
Module calculates controller parameter changes
C. Critic Module NN
Parameter changes
Response of Secondary Cost Formula to Plant States
Predicted response of Plant States to Control Inputs
D. Plant Model NN
$\frac{\partial J(t)}{\partial R_i(t)}$
$\frac{\partial Q_H(n,k,t+1)}{\partial RP(n,k,t)}$

Fig. 2. Adaptive critic controller design system. A simple plant(A)/controller(B) combination(upper left) can be improved using a DHP adaptive critic controller design module (lower right). During normal operations (dashed arrows), the controller views the plant state outputs and changes the order policies as necessary. When the DHP training module is installed (solid arrows), the critic (C) estimates the response of the secondary cost function to changes in system states, and the neural model of the plant (D) estimates the response of the plant to control inputs. This information is used to calculate a new set of operating parameters for the controller.

will require a policy controller (Item B.) that will periodically adjust those policies. The new policies are only as good as the controller that created them, so the DHP technique [25] is used to adapt and improve the controller so it continues to produce near-optimal policies in a changing environment. DHP uses an estimate of the Jacobian of the coupled policy and discrete event system (Item D.) to train a critic function (Item C.). This critic function produces estimates of the gradient of the long-term cost (cost to go) associated with a particular policy controller. These derivatives are then used to adjust the design of the controller to reduce the long-term cost. As the state and control variables in our problem context are discrete valued, various approximations are used to implement differentiable functions needed by DHP.

II. Problem

In this section, we describe the supply chain management problem, provide details of the associated cost function, and list some of the constraints associated with the traditional solution techniques.

Because goods held in inventory cost money to purchase, store, and ship, an efficient organization maintains as small an inventory as possible. However, there are penalties for failing to keep enough goods on hand, ranging from delayed profits to loss of customers. In other situations, inappropriate policies can result in order-of-magnitude swings in inventories and orders [14], [17]. Therefore, an effective organization maintains an inventory that is only as large as necessary to meet the vagaries of demand. The problem suffers from the traditional curse of dimensionality, but problem size can sometimes be reduced through heuristics, and a satisficing manager may be content with a near-optimal solution [15]. Years of development, aided by increased computing power, have improved the ability to create such schedules.

Our three-node supply chain in Fig. 1 can be characterized as a discrete time dynamical system with 36 state variables describing stock on hand and in transit (collectively designated $R(t)$), 18 control inputs specifying the operating policies (reorder point RP, order up-to point UT, and transport capacity procured $\mathrm{T_{cap}}$—collectively designated $u(t)$. In addition, we have four exogenous, nonstationary, stochastic inputs $\mathrm{Q}_D(t)$ representing retail demand. All of these will be indexed for the specific node $\mathbf{n}$, stock $\mathbf{k}$, transport arc $\mathbf{a}$ or transport mode $\mathbf{m}$ involved. The stochastic demand variables remove stock quantities from the demand nodes. When a stock level at a node is reduced to or below the *r*eorder *p*oint policy value for that stock at that node RP(n,k,t), an order is placed for resupply from the next node up the supply chain. The order quantity is determined by the order-*u*p-*t*o quantity policy value UT(n,k,t). When the order is filled, it is shipped subject to the available capacity of the intervening transit arcs as determined by the *t*ransit *c*apacity policy values $\mathrm{T_{cap}(a, m, t)}$.

The underlying problem is one of cost minimization. The function to be minimized is total cost $\mathrm{C_{Total}}$

$$C_{\text{Total}} = C_{\text{Initial}} + C_{\text{Operating}} + C_{\text{Final}} \tag{1}$$

where

$$C_{\text{Operating}} = \sum_{t=0}^{T} \left(C_H(t) + C_P(t) + C_T(t) + C_X(t)\right) \tag{2}$$

i.e. operating cost is the sum of holding costs, purchase costs, transportation costs, and penalties for running out of stocks at demand nodes. A detailed derivation of the cost function is provided in Appendix I.

The system is simulated in the following manner. The system is driven by exogenous demand, $\mathrm{Q}_D(t)$, at each timestep

$$Q_H(n,k,t+1) = Q_H(n,k,t) - Q_D(n,k,t) \tag{3}$$

Orders are generated based on the two policy variables associated with inventory control: the *reorder point*, RP(n,k,t), and the *order up-to point*, $UT(n,k,t)$. When

$$RP(n,\ k,\ t) \geq Q_H(n,\ k,\ t) \tag{4}$$

in which case

$$Q_R(n,k,t) = UT(n,k,t) - (Q_H(n,k,t) + Q_O(n,k,t)) \tag{5}$$

where Q_O is the quantity already on order, but not delivered to the requesting node.

Some nodes are supply nodes (denoted when necessary by the subscript $\mathbf{s}$) that supply the nodes affected by stochastic demand (subscripted $\mathbf{d}$). The quantity of stock $\boldsymbol{k}$ provided by supply node $\mathbf{n}_{\boldsymbol{s}'}$, $\mathrm{Q_S(n_s, k, t)}$, is constrained by $\mathrm{Q_H(n_s, k, t)}$, the quantity on hand at that node, and $Q_R(n_d,k,t)$ the quantity requested by demand node $\mathbf{n}_{\boldsymbol{d}}$

$$Q_S(n_d,k,t) = \min\left(Q_R(n_d,k,t), Q_H(n_s,k,t)\right). \tag{6}$$

This quantity is then shipped along a transit arc $\mathbf{a}$ via some transport mode $\mathbf{m}$, selected based on available capacity and estimated time to stock out of the receiving node. Two transport modes are available: high-speed/high-cost, and low-speed/low-cost. Transit times are fixed for any given mode over a given arc, but vary among the arcs. At any given timestep, the trans-

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

port cost is the sum of fixed costs for all transport capacity purchased $\mathrm{C_{TF}(a,m)}$, plus $\mathrm{C_{TO}(a,m,t)}$ the operating cost for that portion of the transport capacity of mode $\mathbf{m}$ actively involved in moving $\mathrm{Q_S}$ over arc $\mathbf{a}$.

Minimization of the cost function over some planning horizon requires finding a set of inventory and transport policies that are appropriate to the level of demand. Small-scale problems of this nature are traditionally solved analytically using Dynamic Programming (DP), or mixed-integer Linear Programming (LP) techniques [3], [4], [18]. There are three difficulties with these approaches. First, as the problem becomes larger and more complex, the solution space is subject to a combinatorial explosion that limits the effectiveness of traditional approaches. Second, both the DP and LP techniques use the expected value of demand (average demand) at each timestep, while a variant, stochastic dynamic programming, creates an even greater combinatorial explosion by requiring additional "recourse" terms. Unfortunately, real world (stochastic) fluctuations in demand can overwhelm a control policy tuned to the average. Finally, the average itself can change over time, so that policies that might have worked at the start of a planning period might not work toward the end. **What is needed is a methodology that will allow continuous, combined tuning of both inventory and transport policies to a nonstationary environment.**

III. Methodology

Over the last decade, a family of *approximate* dynamic programming techniques utilizing adaptive critics has been developed, techniques that do not directly suffer from the curse of dimensionality, and that can continuously adapt a control system in response to changes in the environment. These techniques work best when starting from a stable controller design. This section addresses three topics: first, we present a description of approximate dynamic programming and the use of dual heuristic programming (DHP) to implement these techniques. Second, we describe the methods we used to find quasioptimal initial controller designs. Finally, we describe the simulation techniques and environments in which these controllers were trained and tested.

A. Approximate Dynamic Programming

In dynamic programming, one develops an optimal policy for control of a system, a "plant" in the control literature, by comparing the costs of all alternative actions at all accessible points in state space through time. Adaptive critic based approximate dynamic programming (ADP) methods start with the assumption that there exists a smoothly parameterized family of controllers that includes the optimal controller. Gradient methods are then used to search for the optimal member of this family. The needed gradients are obtained from a function estimator called a *critic*.

For our problem we use such a critic function, one that estimates the gradient of the secondary utility function (cost to go) at any accessible point in state space. Under the assumption that the critic is accurately estimating the gradient of the cost to go of the policy specified by the controller's parameter values, the critic function can be used to adjust these parameters so as to arrive at a local optimum in the parameterized policy space. This process has been successfully operationalized using artificial neural networks for both the control and critic functions [2], [12], [19], [20], [23], [24].

The method is applied by formulating a "primary" utility function $U(t)$ that embodies a control objective for a particular context in one or more measurable variables. For our context, U(t) is the operating cost, so we have

$$U(t)=C_{\mathrm{Operating(t)}}=(C_H(t)+C_P(t)+C_T(t)+C_X(t))\,. \tag{7}$$

A *secondary* utility function J(t) is then formed

$$J(t)=\sum_{k=0}^{\infty}\gamma^k U(t+k) \tag{8}$$

which embodies the desired control objective, discounted by γ through time. This is Bellman's equation, and a useful identity that follows from it is the Bellman Recursion

$$J(t)=U(t)+\gamma J(t+1). \tag{9}$$

A promising collection of approximation techniques based on estimating the function $J(t)$ using this identity has been proposed by Werbos [25]–[27].

In DHP the critic's outputs are estimates of the gradient of $J(t)$. This method utilizes two distinct training loops, one for the controller and one for the critic estimator. The controller is adjusted to optimize the secondary utility function $J(t)$ for the problem context. A gradient based learning method for controller training requires estimates of the derivatives $(\partial J(t))/(\partial u_i(t))$, where u represents the various control inputs (operating policies): reorder point (RP), up-to-point (UT), and transport capacity $(\mathrm{T_{cap}})$. We are thus estimating the way changes in policies impact our estimated total cost over time. The critic function is trained based on the consistency of its estimates through time judged using the Bellman Recursion. For the DHP method, where the critic estimates the derivatives of J(t) with respect to the system states (stock levels, transport available, etc.) R, the critic's output is defined as

$$\lambda_i(t)=\frac{\partial J(t)}{\partial R_i(t)}. \tag{10}$$

We differentiate both sides of Bellman's Recursion

$$\frac{\partial}{\partial R_i(t)}J(t)=\frac{\partial}{\partial R_i(t)}\left(U(t)+\gamma J(t+1)\right) \tag{11}$$

to get the identity used for critic training

$$\begin{aligned}\lambda_i(t)=&\frac{\partial U(t)}{\partial R_i(t)}+\sum_j\frac{\partial U(t)}{\partial u_j(t)}\frac{\partial u_j(t)}{\partial R_i(t)}\\&+\sum_k\gamma\lambda_k(t+1)\left[\frac{\partial R_k(t+1)}{\partial R_i(t)}\right.\\&\left.+\sum_m\frac{\partial R_k(t+1)}{\partial u_m(t)}\frac{\partial u_m(t)}{\partial R_i(t)}\right].\end{aligned} \tag{12}$$

To evaluate the right hand side of this equation we need a model of the system dynamics that includes all the terms from

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

the Jacobian matrix of the coupled plant-controller system, e.g., all the $(\partial R_j(t+1))/(\partial R_i(t))$ and $(\partial R_j(t+1))/(\partial u_i(t))$. In terms of our specific problem, we would be estimating such things as $(\partial Q_H(n,k,t+1))/(\partial Q_H(n,k,t))$, and $(\partial Q_H(n,k,t+1))/(\partial UT(n,k,t))$.

The control policy is updated using the chain rule and the system model to translate critic outputs into estimates of $(\partial J(t))/(\partial u_i(t))$, i.e.

$$\frac{\partial J(t)}{\partial u_k(t)} = \frac{\partial U(t)}{\partial u_k(t)} + \gamma \sum_i \lambda_i(t+1) \frac{\partial R_i(t+1)}{\partial u_k(t)}. \tag{13}$$

This gradient can then be used with any local optimization technique to optimize a policy (control) function, e.g. standard LMS adaptation for linear filters and fuzzy systems or Backpropagation for neural networks. The entire process can be characterized as a simultaneous optimization problem: gradient-based optimization of the critic function estimator together with gradient based optimization of control policy parameters based on the $\nabla J(t)$ estimates obtained from the critic. Different strategies have been utilized to get both these optimizations to converge. One possibility is to alternate between optimizing the critic estimator and optimizing the control policy [19]. It is also possible to do both optimization processes simultaneously [12], [22].

It is known that if the DHP method converges, it converges to the correct (unbiased) solution when solving continuous time, deterministic or stochastic, linear quadratic regulator problems [28]. Quadratic convergence of the method for control system design in an online environment cannot be guaranteed due to potential instability of the plant from resonant or pathological control input sequences during adaptation. For offline optimization, plant instability can be handled by setting boundaries in state space for acceptable plant operation, and resetting the simulation when the boundaries are violated.

As DHP relies on gradient-based optimization of $J(t)$, on nonlinear problems it inherently suffers from the problem of (unsatisfactory) local optima. Finding the global optimum of $J(t)$ is in general subject to the "No Free Lunch Theorem." Approximate dynamic programming techniques offer tractable methods for local hill climbing on the $J(t)$ landscape of policy parameter space. Initialized at a random point in parameter space, these methods may be trapped by a local optimum at an unsatisfactory policy. One can attempt to avoid this by applying problem specific knowledge to the choice of initial controller parameters, in the hope of being near a satisfactorily high hill (or deep valley). In this paper we avoid this problem by starting with a quasioptimal initial policy (obtained by some other method) that is already in some sense satisfactory, and apply the DHP method to improve this starting policy.

B. Optimization and Search for an Initial Controller

As stated earlier, use of an LP model is subject to a number of difficulties. Nevertheless, it is a proven technique that will produce an optimal solution for a single deterministic demand scenario. For that reason, this study uses an LP solution as a baseline. The quasioptimal starting solution mentioned in the previous paragraph was obtained using a genetic algorithm (GA) to search the policy space. This section describes both the LP and GA tools used in the study. The GA- and LP-derived policies were developed based on a fixed demand schedule, and the demand amounts were based on the expected value of the demand for a specific stock at a specific demand node, e.g. the first demand node, node two had 2 units of demand for stock 1, and 2.5 units of demand for stock 2 every timestep for 90 simulation days.

1) Linear Programming Model: Our LP was a 4,883 constraint mixed integer multiperiod model. The large number of constraints is indicative of the seriousness of the combinatorial explosion problem. In order to make the system viable beyond just the planning horizon, we imposed the constraint that the amount of stock on hand at each location on the final day be equal to the amount on hand on the first day.

The LP model was solved using LINDO, a commercial modeling tool, and represents an analytically optimal result. For the *fixed demand* scenario, the cost of the LP-derived optimal policy turned out to be only 28% that of the fixed policy found by the GA. We note however, that in general, the LP solution is not an optimal solution to the problem with stochastic *demand*.

2) Genetic Algorithm: While the LP is guaranteed to produce an optimal solution to the deterministic demand problem, the GA is not. The fitness terrain is spiky, and the search problem is difficult, so the GA solution is generally only quasioptimal. The evaluation function used for the GA was a discrete event simulation of the full problem. The demand schedule used for the GA was the same as for the LP; demand was assumed fixed at some expected value for the duration of the experiment.

The GA implemented for the present work used a chromosome with three genes for each node—initial stock level, reorder point, and order-up-to point. It also had one gene representing the transport capacity purchased for each transport arc/mode combination.

In this GA, the top half of the population reproduces stochastically (roulette wheel selection), with reproduction probability based on sigma scaling [16]. Crossover is a version of uniform crossover. The probability of crossover was set at 0.30, and the mutation rate was set at 0.01. Business constraints necessary to the operation of the simulation were handled by repairing the chromosome as it was being created.

C. The Logistics System Simulation Process

Because the research involved modeling the operation of a physical distribution system in a manner that replicates the complexity that exists in the real world, it was necessary to use a discrete event simulation (Item A in Fig. 2) for both the GA fitness function [6], and the testbed for the controller. The simulation tracked the passage of the stocks through the various nodes, the allocation of transport resources, and the total cost of operating the supply chain. Stock inventory levels in a typical run are shown in Fig. 3.

At the end of the simulation, penalties were assessed for holding stocks in excess of requirements or for not having stocks on hand equal to the initial quantities. This is an artificial policy penalty, similar to the constraints on the LP, and is intended to ensure that the system can continue to operate beyond the planning horizon.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

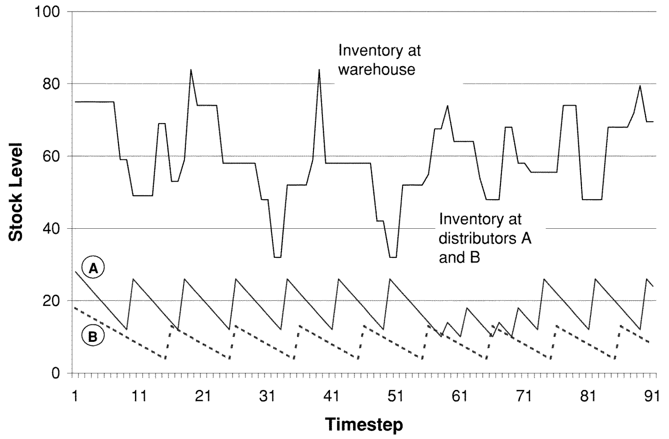


Fig. 3. Simulation output. A single stock is held at a warehouse (upper solid line), which supports two distributors, A (lower solid line), and B (lower dashed line). The inventory levels fluctuate based on the demand, the order policies, transport available, and the delivery time. As an example of the process, the warehouse shipped an order to distributor A on day seven, delivered on day nine, and shipped another to B on day nine, delivered on day 15. Between days 60 and 70, a shortage of transport kept distributor A from getting a full replenishment.

D. Test Regime

External demand is the sole exogenous variable. Changes in demand are defined in two ways: the overall pattern of demand (demand schedule), and the characteristics of the variance of that pattern (noise regime). We generated three different demand schedules, and combined each with two different noise regimes. The final test was to use a real-world data set, modified to fit our test conditions. The datasets used are included as worksheets in an Excel file as Attachment 1, and are described in Tables I and II.

1) Demand Schedule: The three demand schedules are as follows.

Baseline—BL (stationary demand): Demand was fixed for a 360-day test period. This is the demand schedule (without noise) that was used to develop the startup policies using the LP and GA. For controller training and testing, we applied the noise described below and in Table I.

Increasing Average Demande—IAD (smooth increase): In the second demand schedule, we assumed a fixed increase in demand at every timestep throughout the period, sufficient to increase demand by 20% over the 360-day test period.

Delta Demand—DD (step increase): The final demand schedule assumed an environment where demand increased by a 10% step at the very beginning of the test period. This new demand level persisted unchanged throughout the 360-day test period.

The IAD and DD schedules were used only for test, not for training, and contained the added noise described below and in Table II.

2) Noise Regime: Three different kinds of noise were added to the demand schedules described above.

Poisson-Derived Noise: The Poisson distribution is a discrete distribution that describes the number of occurrences of an event during a time interval. It is often used to describe the distribution of demand at the retail level [10], [13]. It has the characteristic that it produces an integer stream, and that the mean and the variance are equal.

For demand schedules with noise, we drew from a Poisson distribution with a mean ten times larger than our target, rescaling the numbers as necessary to meet the desired noninteger demand schedule. This produced a Poisson-derived data set, with a variance one-tenth of the mean.

One over f: The assumption of the Poisson distribution is that there are no underlying forces driving daily changes in demand about the mean. In many situations this may not be true. For example, daily demand might be cyclically affected by day of the week, month, and season.

The 1/f distribution is the result of the superposition of many different rates of demand, and has the characteristic that a logarithmic plot of its power spectrum falls on a straight line. We used a 1/f distribution scaled to provide the same average demand used by the LP tool and the GA. We believe that this study represents its first use in an inventory control problem.

Real World: The acid test is to see how a system will perform under real world demand conditions. We obtained a 360-day sequence of demand for a machine part at a local manufacturer. The power spectrum of this data set appears similar to that of the 1/f distribution. The distribution is decidedly nonnormal, and the variance is large. This data was rescaled to provide the same overall averages as that found in the baseline case.

IV. Implementation

This section addresses implementation details and some limitations of the study.

A. Limitations

This is a proof-of-concept study and the goal was to implement a system that was as simple as possible while retaining a level of complexity that allows demonstration of the method. To that end, the number of stocks, nodes, and modes are limited; the delivery times, although different for each arc/mode, are fixed; and standard physical distribution techniques such as skip- or cross-echelon transfers of stock are disallowed. In addition, we made no effort to optimize the structure of the neural nets, selecting hidden layer sizes and node activation functions arbitrarily, or according to rough rules of thumb.

B. System Identification

We need a model that describes the responses of a system to control and state changes, where the system model is capable of approximating the partial derivatives $(\partial R_j(t+1))/(\partial R_i(t))$ and $(\partial R_j(t+1))/(\partial u_i(t))$. So, for example, our specific problem requires estimates of $(\partial Q_H(n,k,t+1))/(\partial Q_H(n,k,t))$, and $(\partial Q_H(n,k,t+1))/(\partial RP(n,k,t))$.

To obtain these, we trained a simple multi-layer perceptron (MLP) Neural Net (item D. in Fig. 2.) to model the supply chain system responses, then used the model NN to develop an ordered array of derivatives.

By saving all the interim results of the GA during the search for an optimal policy set (including those generated using far-from-optimal inputs) we obtain a data set that not only spans the search space, but also provides a higher proportion of I/O pairs

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

TABLE I
NOISE CHARACTERISTICS FOR FIXED AND STATIONARY DEMAND SCHEDULES

	Node 2 Stock 0	Node 2 Stock 1	Node 3 Stock 0	Node 3 Stock 1
TRAIN, 90-day: via Fixed Demand.	1.0	1.5	2.0	2.5
TRAIN, 90-day: via Poisson-derived Stationary Stochastic Demand.	Mean: 1.0 Variance: 0.11	Mean: 1.5 Variance: 0.15	Mean: 1.9 Variance: 0.16	Mean: 2.5 Variance: 0.27
TEST, 360-day: via Poisson-derived Stationary Stochastic Demand.	Mean: 1.0 Variance: 0.10	Mean: 1.5 Variance: 0.16	Mean: 2.0 Variance: 0.21	Mean: 2.5 Variance: 0.26
TEST, 360-day: via 1/f Stationary Stochastic Demand.	Mean: 1.0 Variance: 0.07	Mean: 1.5 Variance: 0.16	Mean: 2.0 Variance: 0.35	Mean: 2.5 Variance: 0.44
TEST, 360-day: via Real World Stationary Stochastic Demand.	Mean: 1.0 Variance: 16.41	Mean: 1.4 Variance: 23.76	Mean: 1.9 Variance: 32.64	Mean: 2.6 Variance: 59.76

TABLE II
NOISE CHARACTERISTICS FOR NON-STATIONARY DEMAND SCHEDULES

Demand Distribution	Node 2 Stock 0	Node 2 Stock 1	Node 3 Stock 0	Node 3 Stock 1
TEST, 360-day: via Poisson-derived Non stationary Demand, with increasing averages.	Mean: 1.1 Variance: 0.12	Mean: 1.7 Variance: 0.20	Mean: 2.2 Variance: 0.27	Mean: 2.7 Variance: 0.34
TEST, 360-day: via 1/f Non stationary Demand, with increasing averages.	Mean: 1.0 Variance: 0.09	Mean: 1.7 Variance: 0.21	Mean: 2.2 Variance: 0.44	Mean: 2.8 Variance: 0.55
TEST, 360-day: via Real World Non stationary Demand, with increasing averages.	Mean: 1.5 Variance: 16.42	Mean: 1.9 Variance: 23.57	Mean: 2.3 Variance: 33.21	Mean: 3.1 Variance: 60.66
TEST, 360-day: via Poisson-derived Non stationary Demand, with delta demand changes.	Mean: 1.1 Variance: 0.11	Mean: 1.7 Variance: 0.18	Mean: 2.2 Variance: 0.22	Mean: 2.7 Variance: 0.29
TEST, 360-day: via 1/f Non stationary Demand, with delta demand changes.	Mean: 1.1 Variance: .09	Mean: 1.7 Variance: 0.20	Mean: 2.2 Variance: 0.43	Mean: 2.7 Variance: 0.54
TEST, 360-day: via Real World Non stationary Demand, with delta demand changes.	Mean: 1.2 Variance: 21.70	Mean: 1.6 Variance: 31.43	Mean: 2.2 Variance: 43.16	Mean: 3.0 Variance: 79.04

in the vicinity of good solutions. We believe this is the first time such an approach has been used.

The plant model NN has 62 inputs (36 states, 18 controls, and eight exogenous demand variables (representing the sample mean and standard deviations of demand $\mathrm{Q}_D(\mathrm{n},\mathrm{k})$ from the previous seven days)), an arbitrary 102-neuron hidden layer, and 36 system state outputs. The hidden layer neurons have hyperbolic tangent activation functions, while the output layer neurons have linear activation functions. Forty-eight examples of the randomly-initialized NN were trained for 20 passes through the training data. The resulting nets had an average training set $\mathrm{RMSE} = 0.60$ and an average test set $\mathrm{RMSE} = 0.62$. While this does not appear to be a sterling result, Shannon [22] has shown that even very poor predictive models can provide derivatives that are adequate for successful adaptive critic learning. This turned out to be the case for the present experiment.

The trained model NN may be used to provide information on the required partial derivatives as follows: first, pass the current state and control information $R(t)$ (e.g. Q_H(n,k,t) and $u(t)$ (e.g. UT(n,k,t) forward through the NN, and save the slope of the activation function for each of the NN's nodes into a structure known as the dual of the NN. Then, for each state R_i, a 1 is passed through the dual of the NN, with all other inputs to the dual NN held to zero [26]. The derivative information will appear as outputs of the dual NN (equivalent to the input nodes, $R(t)$ and $u(t)$, of the original NN). This information, combined with the differentiable cost equations and the Critic Neural Network's (item C. in Fig. 2.) estimate

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

of λ, is used to locally optimize toward an appropriate policy function.

C. Policy Representation

We need a smoothly parameterized family of policies over which to search for an optimal policy. We formulated such a family using a neural net policy controller, trained using Poisson-derived demand with a stationary mean equal to the value used by the GA and LP. The policy controller NN has 44 inputs, and 18 outputs, one output for each control variable. The number of neurons in the hidden layer was arbitrarily set at 68. The hidden layer neurons used hyperbolic tangent activation functions, while the output neurons used linear activation functions. The inputs included the state variables and eight demand descriptors, including the average and standard deviation of demand over the previous period for each stock at each retail node. The NN's output represents recommended deltas to the current reorder and order-up-to points for each node/stock pair and the maximum capacity allocated for each arc/mode. With this arrangement, the starting policy values are equivalent to a set of biases on the output layer. The network was trained using standard backpropagation methods.

D. Critic Estimator

The critic network has 36 inputs, and 36 outputs, one for each state variable. The number of neurons in the hidden layer was arbitrarily set at 80. The hidden layer neurons used hyperbolic tangent activation functions, while the output neurons used linear activation functions. The inputs are the state variables $R(t)$ and the network's outputs, $\lambda(t)$, are estimates of the partial derivatives of the components of the operating cost $C_{\text{OPERATING}}$ with respect to the state variables. The critic network was trained using a zero discount rate ($\gamma = 1$), using standard Backpropagation methods.

V. Results

We tested the resulting neural control system against the policies found by the GA and LP across the full range of demand variation described in Section III-D. The NN controller design achieved using the DHP adaptive critic method was able to improve on all fixed policies (Table III). These results were highly consistent—the coefficient of variation (Std Dev/Avg) in all test series was 0.05% ($\mathrm{n} = 250$).

1) Poisson-Derived Noise: In this set of experiments, we first trained the NN controller on 90 days of Poisson-derived demand, then tested it in a 360-day simulation against both Poisson—derived demand (Fig. 4) and against real-world demand (Fig. 5). The DHP-designed NN controller, starting from both an LP-derived and a GA-derived policy set, turned in significant improvements over the LP- and GA-designed fixed-policy controllers.

2) One-Over-F Noise: In this set of experiments, we first trained the NN controller on 90 days of 1/f-distributed demand, then tested it in a 360-day simulation against both 1/f-distributed demand (Fig. 6) and against real-world demand (Fig. 7). The NN controller, starting from both a GA-derived and an LP-derived policy set, again turned in significant improvements over the fixed-policy controllers.

TABLE III
PERFORMANCE OF NEURAL VS. FIXED POLICIES. COLUMNS SHOW THE CONTROLLER TYPE (NEURAL NET (NN) OR FIXED POLICY (FP)) AND THE SOURCE OF THE INITIAL POLICIES (GENETIC ALGORITHM (GA) OR LINEAR PROGRAM (LP)). ROWS SHOW DEMAND SCENARIO AND NOISE REGIME. SCORES ARE BASED ON 1000 CONTROLLERS IN EACH CATEGORY, AND ARE IN ARBITRARY PENALTY POINTS

	NNLP	NNGA	FPLP	FPGA
Trained on Normally distributed noise Tested on Normally distributed noise				
Baseline	778	455	1220	818
DeltaDemand	505	423	2131	916
Increasing Average Demand	966	379	2209	1162
Average	*750*	*419*	*1853*	*966*
Trained on 1/f noise Tested on 1/f noise				
Baseline	744	412	1675	1043
DeltaDemand	976	418	3858	1360
Increasing Average Demand	743	446	1874	2180
Average	*821*	*425*	*2469*	*1528*
Trained on Normally distributed noise Tested on real world data				
Baseline	720	477	2745	2395
DeltaDemand	921	403	2069	2665
Increasing Average Demand	937	443	3859	5371
Average	*859*	*441*	*2891*	*3477*
Trained on 1/f noise Tested on real world data				
Baseline	683	458	2745	2395
DeltaDemand	816	447	2069	2665
Increasing Average Demand	843	490	3859	5371
Average	*781*	*465*	*2891*	*3477*

VI. Conclusions

We have demonstrated that adaptive critic based approximate dynamic programming techniques can be effectively used with systems characterized by discrete valued states and controls. The improvements over fixed policies (found using either a LP model or a genetic algorithm) average 83% under conditions both of stationary and nonstationary demand in a high penalty environment.

The cost structure of the simulation was designed to maintain pressure on both the GA evolutionary search and the NN learning process, by penalizing severely any failure to maintain

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

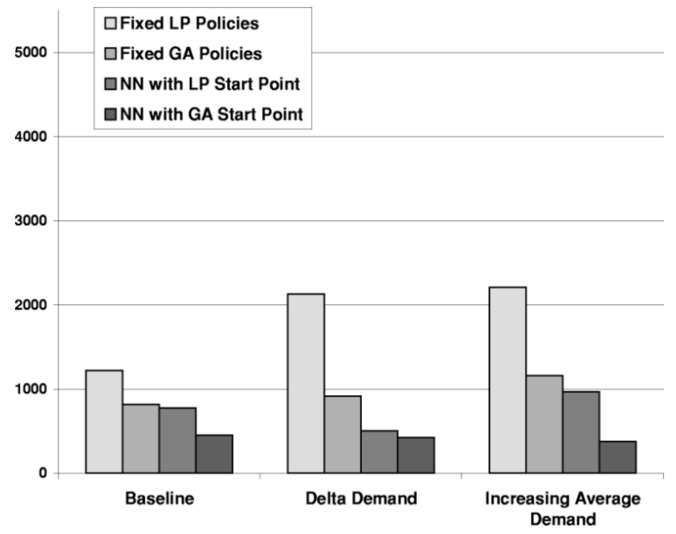


Fig. 4. Performance Comparison—Neural Controller vs. Fixed Policy Controllers. The NN controller was trained and tested on normally-distributed demand. The Y-axis units are arbitrary penalty points, consistent across all tests.

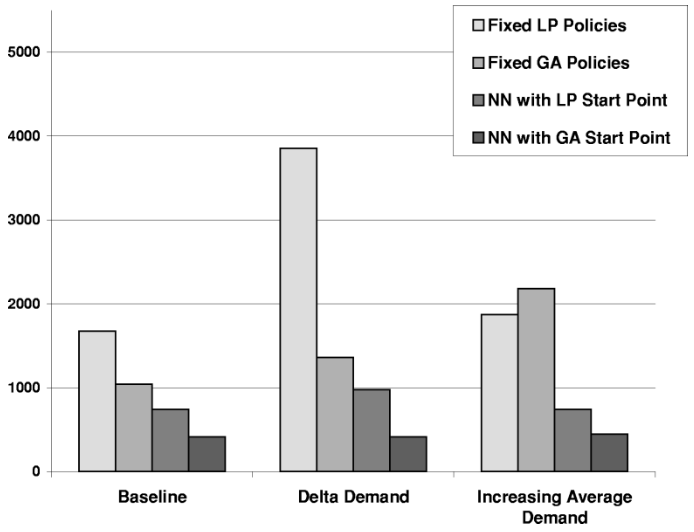


Fig. 6. Performance Comparison (1/f test)—Neural Controller vs. Fixed Policy Controllers. The NN controller was trained and tested on 1/f-distributed demand. The Y-axis units are arbitrary penalty points, consistent across all tests.

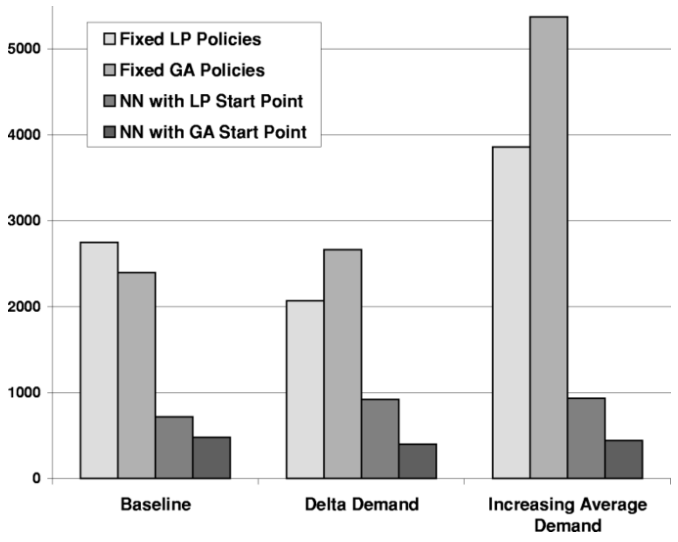


Fig. 5. Performance Comparison (Real-world test)—Neural Controller vs. Fixed Policy Controllers. The NN controller was trained on normally-distributed demand and tested against a real-world data set. The Y-axis units are arbitrary penalty points, consistent across all tests.

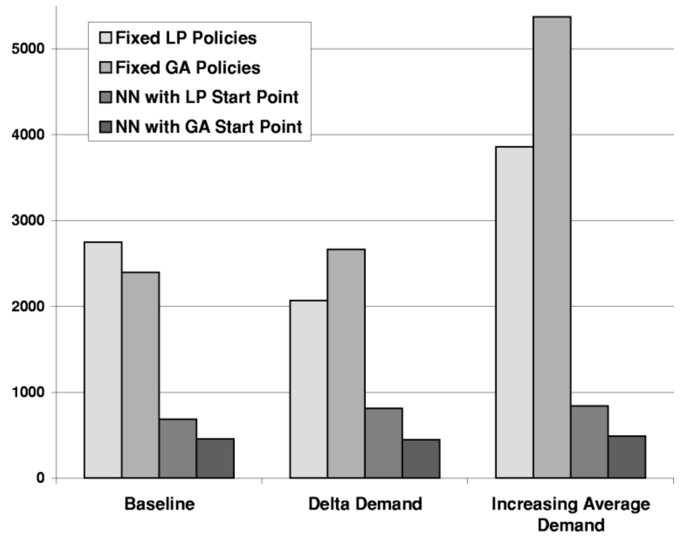


Fig. 7. Performance Comparison (Real-world test)—Neural Controller vs. Fixed Policy Controllers. The NN controller was trained on 1/f-distributed demand and tested against a real-world data set. The Y-axis units are arbitrary penalty points, consistent across all tests.

stocks or transport resources at a level appropriate to the scenario. Costs reported, therefore, are not directly comparable to normal business operating costs.

Extension of this approach from retail inventory control to inventory control in a production environment appears simple—many of the techniques of sales inventory control are directly applicable to production inventory control. The key differences include covariant demand among the various stock items, the requirement for a specified dwell time at each node traversed, and setup costs for each stock/assemblage at each node it traverses.

In summary, we used a version of approximate dynamic programming to find a quasioptimal solution to a difficult business problem. In addition, we have demonstrated successful use of off-optimal data from an initial evolutionary search to provide system identification inputs for a system model neural net. Techniques taken from the engineering context were here successfully applied to day-to-day business applications. This appears to be a fruitful source of additional solutions for business.

APPENDIX NOTATION

In the interest of compactness, we are using notation from both the control and inventory literatures, where each is appropriate. For example, when discussing operations and operating policies, we follow the inventory literature and address such things as stock levels Q_H, stock reorder points RP, and transport capacity leased T_{cap}; when deriving a control equation we follow the control literature and consider all these variables as simply different system states R or control inputs u.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

A. Nomenclature

$\mathrm{C_{Final}}$	Final penalty costs based on the state of the system at the end of the planning period.
$\mathrm{C_H}$	Holding cost.
$\mathrm{C_{Initial}}$	Initial cost of setting up the system.
$\mathrm{C_{Operating}}$	Daily operating costs.
$\mathrm{C_P}$	Purchase cost of goods purchased after the start of the operation (only applies to node 1).
$\mathrm{C_T}$	Transport cost.
$\mathrm{C_{TF}}$	Fixed cost of transport.
$\mathrm{C_{TO}}$	Operating cost of transport.
$\mathrm{C_{Total}}$	Total cost.
$\mathrm{C_{TX}}$	Penalty cost for not having enough transport to move filled stock orders.
$\mathrm{C_X}$	Stock out penalties for not being able to meet demand.
D	Exogenous demand
P_H	Holding cost per unit of stock.
P_P	Purchase price per unit of stock.
P_{TF}	Fixed price per unit of transport.
P_{TO}	Operating price per unit of transport engaged in movement of goods.
P_{TX}	Penalty price for transport shortfall.
P_X	Penalty price per unit of stock shortfall.
Q_D	Exogenous demand.
Q_H	Quantity of stock on hand.
Q_P	Quantity of stock purchased.
Q_R	Quantity requested by a node.
Q_S	Quantity supplied for transport.
Q_{XS}	Stock shortfall quantity.
Q_{XT}	Quantity of stock supplied that cannot be shipped.
T_{CAP}	Transport capacity procured for the current control period.
RP	Reorder point.
UT	Order-up-to point.

Cost Function Derivation

The cost equation may be broken down as shown

$$C_{\mathrm{Total}} = C_{\mathrm{Initial}} + C_{\mathrm{Operating}} + C_{\mathrm{Final}} \tag{14}$$

$$C_{\mathrm{Initial}} = \sum_{n=1}^{N}\sum_{k=0}^{K} C_P(n,k)C_T(n,k) \tag{15}$$

$$C_{Operating} = \sum_{t=0}^{T}(C_H(t)+C_P(t)+C_T(t)+C_X(t)) \tag{16}$$

$$C_{Final} = Cx\sum_{n=1}^{N}\sum_{k=0}^{K}\max\left(Q_H(n,k,t)\right.$$
$$\left.-Q_R(n,k,t),0\right), \tag{17}$$

$$C_H(t) = \sum_{n=1}^{N}\sum_{k=0}^{K} P_H(n,k)Q_H(n,k,t) \tag{18}$$

$$C_P(t) = \sum_{k=0}^{K} P_p(k)Q_P(n,k,t) \tag{19}$$

$$C_X(t) = \sum_{n=1}^{N}\sum_{k=0}^{K} P_X(n,k)Q_{XS}(n,k,t) \tag{20}$$

$$C_T(t) = C_{TF}(t) + C_{TO}(t) + C_{TX}(t) \tag{21}$$

$$C_{TF}(t) = \sum_{a=0}^{A}\sum_{m=0}^{M} P_{TF}(m)T_{CAP}(a,m,t) \tag{22}$$

$$C_{TO}(t) = \sum_{a=0}^{A}\sum_{m=0}^{M} P_{TO}(a,m)Q_s(a,m,t) \tag{23}$$

$$C_{TX}(t) = \sum_{a=0}^{A}\sum_{m=0}^{M} P_{TX}Q_{XT}(a,m,t) \tag{24}$$

where the summations in the stock equations are over N locations and K types of stock, and the summations in the transport equations are over A transport arcs and S transport steps.

References

[1] Y. Aneja and K. Nair, "Bicriteria transportation problem," *Manage. Sci.*, vol. 25, pp. 73–78, 1979.

[2] A. Barto, R. Sutton, and C. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 834–846, 1983.

[3] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.

[4] R. Bellman and S. Dreyfus, *Applied Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1962.

[5] A. Bit and M. Biswal *et al.*, "Fuzzy programming approach to multiobjective solid transportation problem," in *Fuzzy Sets Syst.*, vol. 57, 1993, pp. 183–194.

[6] J. Clymer, "Optimization of simulated system effectiveness using evolutionary algorithms," *Simulation*, vol. 73, no. 6, pp. 334–340, 1999.

[7] R. Gullu, R. Erkip, and N. Erkip, "Optimal allocation policies in a two-echelon inventory problem with fixed shipment costs," *Int. J. Prod. Res.*, vol. 46–47, pp. 311–321, 1996.

[8] D. Han and S. N. Balakrishnan, "Midcourse guidance law with neural networks," in *Proc. AIAA Guidance, Navigation, and Control Conference*, Denver, Co., 2000, AIAA #2000-4072.

[9] H. Jonsson and E. Silver, "Overview of a stock allocation model for a two-echelon push system having identical units at the lower echelon," in *Multi-Stage Production Planning and Inventory Control*. New York: Springer-Verlag, 1986.

[10] D. Kelton, R. Sadowski, and D. Sadowski, *Simulation With Arena*. New York: McGraw-Hill, 1997.

[11] G. Lendaris, L. Schultz, and T. Shannon, "Adaptive critic design for intelligent steering and speed control of a 2 axle vehicle," in *Proc. IJCNN*, Como, Italy, July 2000.

[12] G. Lendaris and T. Shannon, "Application considerations for the DHP methodology," in *Proc. IEEE Int. Joint Conf. Neural Networks* Anchorage, AK, May 1998.

[13] C. Lewis, *Demand Analysis and Inventory Control*. Hants: Saxon House, 1975.

[14] R. Metters, "Quantifying the bullwhip effect in supply chains," *J. Oper. Manage.*, vol. 15, pp. 89–109, 1997.

[15] T. Morton and D. Pentico, *Heuristic Scheduling Systems*. New York: Wiley, 1993.

[16] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996.

[17] E. Mosekilde, E. Larsen, and J. Sterman, "Coping with complexity: deterministic chaos in human decision-making systems," in *Beyond Belief: Randomness, Prediction and Explanation in Science*, J. Casti and K. Anders, Eds. Boca Raton, FL: CRC, 1990.

[18] S. Oh and A. Haghani, "Testing and evaluation of a multi-commodity multi-modal network flow model for disaster relief management," *J. Advanced Transportation*, vol. 31, no. 3, pp. 249–282, 1997.

[19] D. Prokhorov, "Adaptive Critic Designs and their Application," Ph.D. dissertation, Dept. Electrical Engineering, Texas Tech University, Lubbock, TX, 1997.

[20] D. Prokhorov and D. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Networks*, vol. 8, pp. 997–1007, 1997.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.

[21] A. Sarmiento and R. Nagi, “A review of integrated analysis of production-distribution systems,” *IIE Trans.*, vol. 31, no. 11, pp. 1061–1075, 1999.

[22] T. Shannon, “Partial, noisy and qualitative models for DHP adaptive critic neuro-control,” in *Proc. Int. Joint Conf. Neural Networks*, Washington D.C., July 1999.

[23] S. Shervais, “Developing improved inventory and transportation policies for distribution systems using genetic algorithm and neural network methods,” in *Proc. World Conf. Systems Sciences*, Toronto, Canada, July 2000, pp. 200 059-1–200 059-17.

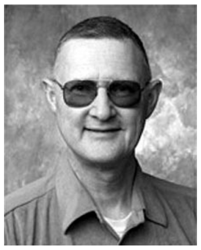
[24] S. Shervais and T. Shannon, “Improving theoretically-optimal and quasioptimal inventory and transportation policies using adaptive critic based approximate dynamic programming,” in *Proc. International Joint Conf. Neural Networks*, Washington D.C., July 2001.

[25] P. Werbos, “Neurocontrol and supervised learning: An overview and evaluation,” in *Handbook Intelligent Control*, D. White and D. Sofge, Eds. New York: Van Nostrand Rheinhold, 1992.

[26] ——, “Optimization methods for brain-like intelligent control,” in *Proc. 34th Conf. Decision Control*. Piscataway, NJ: IEEE Press, 1995, pp. 579–584.

[27] ——, “Neurocontrol and related techniques,” in *Handbook Neural Comput. Applicat.*, A. Maren, C. Harston, and R. Pap, Eds. New York: Academic, 1990, pp. 345–380.

[28] ——, “Stable adaptive control using new critics designs,” in *Handbook Applied Comput. Intell.*, N. Karayiannis, M. Padgett, and L. Zadeh, Eds. Boca Raton, FL: CRC, 2000.



Stephen Shervais (M’01) received the Ph.D. degree in systems science from Portland State University, Portland, OR, in 2000.

He has 20 years experience in Air Force Intelligence, including eight with the Defense Intelligence Agency. In 1989, he joined the staff of Electrospace Corporation, assisting in the development of the Military Intelligence Information Processing System, and in 1992, he moved to CACI Corporation, to manage development of a simulations and modeling database for the Air Force Studies and Analyzes Agency. In 1999 he became an Assistant Professor of Management Information Systems in the College of Business and Public Administration, Eastern Washington University, Cheney, Washington.

Dr. Shervais is a Senior Member of the American Institute of Aeronautics and Astronautics, and a member of IEEE Computer, ACM, INNS, and the International Society for Artificial Life.

Thaddeus T. Shannon (S’99) received the B.A. degree in mathematics from Reed College, Portland, OR, in 1986, the M.S. degree in systems science from Portland State University, Porland, OR, in 2001, and is currently pursueing the Ph.D. degree in systems science at Portland State University.

He has worked as an analyst in the wholesale electric utility industry and in the entertainment industry as a lighting designer, stage manager, and rigger for theater and dance companies, major festivals, corporate tradeshows, televised special events and concerts.

Mr. Shannon is a member of IATSE Local #28, Phi Kappa Phi, and the AMS.

George G. Lendaris (F’82) received the B.S., M.S., and Ph.D. degrees from the University of California, Berkeley.

He then joined the GM Defense Research Laboratories, where he did extensive work in control systems, neural networks, and pattern recognition. In 1969, he went to academia, first joining the Oregon Graduate Institute, where he was Chair of the Faculty, and two years later, moved to Portland State University (PSU), Portland, OR, where he became one of the founders and developers of their Systems Science Ph.D. Program, where he has served for the past 30-plus years. In this context, he expanded his academic and research activities into general system theory and practice, and, more recently, to computational intelligence. He has served a number of capacities at PSU, including President of the Faculty Senate. He is currently Director, Systems Science Ph.D. Program and the NW Computational Intelligence Laboratory. He has been active in neural network research since the early 1960’s, and, in particular, the past 15 years. During recent years, his research has focused on application of the adaptive critic and dynamic programming methods to control system design.

Dr. Lendaris developed the optical Diffraction Pattern Sampling methods of pattern recognition and was declared “Father of Diffraction Pattern Sampling” by the SPIE in 1977, and was elevated to Fellow of the IEEE in 1982 in recognition of this seminal work. He is past Vice President of the SMC Society, and more recently, on its AdCom. He has been active in the neural network professional arena, serving a number of capacities, such as General Chair of the 1993 IJCNN, up to his just-completed term as President of the International Neural Network Society.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:15:32 EST from IEEE Xplore. Restrictions apply.