Improving quasi-optimal inventory and transportation policies using adaptive critic based approximate dynamic programming

Stephen Shervais
Eastern Washington University,
Cheney, WA 99004, USA

Thaddeus T. Shannon[1]
Portland State University,
Portland, OR 97207, USA

Abstract-- We demonstrate the possibility of optimal control of physical inventory systems in a non-stationary fitness terrain, based on the combined application of evolutionary search and adaptive critic terrain following. We show that adaptive critic based approximate dynamic programming techniques based on plant-controller Jacobeans can be used with systems characterized by discrete valued states and controls. Improvements upon a quasi-optimal policy found using a genetic algorithm in a high-penalty environment, average 66% under conditions both of stationary and non-stationary demand.

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

I. Introduction

This paper demonstrates the use of adaptive critic based approximate dynamic programming techniques to the tuning and adaptation of inventory control and transportation policies for a physical distribution system within a changing business environment. Such distribution systems feature warehouses/depots where inventory can be cheaply stored and which feed higher-cost retail outlets which satisfy final customer demand (Figure 1). Transportation resources are multimodal, but may be limited or difficult to change. Final demand may fluctuate, and average demand can change over time. Most prior studies in this application area have concentrated on inventory (economic order

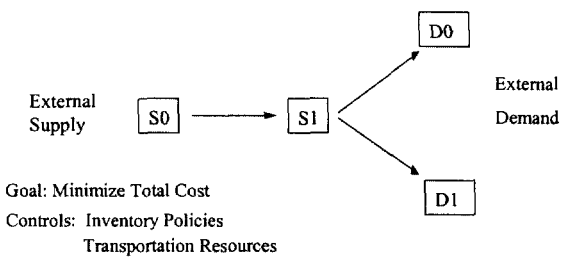

Figure 1. The Physical Distribution Problem. Inventory is held at S1 and distributed to inventories at D0 and D1 using limited transport resources.

quantity) [6][7][17], or on transportation (solid transportation problem) [1][5] allocations. This work addresses both: selection of an optimal set of policies for a multi-product, multi-echelon, multi-modal physical distribution system, in a non-stationary environment. The problem is highly multi-dimensional, even with a small system. Both state and control variables may be discrete valued and end use demand is often characterized by a random variable. The cost surface in policy space for such systems tends to be quite discontinuous, with low penalty and high penalty regions separated by no more than a single transport unit.

We start by performing a global search for a quasi-optimal policy using a genetic algorithm (GA) in a stationary environment characterized by a discrete event simulation. All GA results, not just the final optimum, are saved and used to develop a neural network plant-model. We then implement the adaptive critic based, dual heuristic programming (DHP) method to improve upon the quasi optimal policy found by the GA, and adapt it to a changing environment. DHP uses the Jacobian of the coupled policy-discrete event system to train a critic function. This critic function produces estimates of the partial derivatives of the long-term cost associated with a particular policy. These derivatives are then used to adjust the policy, reducing the long-term cost. As the state and control variables in our problem context are discrete valued, numerous approximations are used to implement differentiable functions needed by DHP.

II. The Problem

The task addressed by this paper is that of adjusting the control policies for the physical distribution system shown in Figure 1 so that it can continue to operate effectively in a stochastic, non-stationary environment. The underlying problem is usually presented as a cost minimization problem. The function to be minimized is total cost C_{Tot}, which consists of the initial and final costs, plus the incremental costs, summed over a planning time horizon T

$$C_{Tot} = C_{Init} + C_{Incr} + C_{Final}, \text{ where} \quad (1)$$

$$C_{Incr} = \sum_{t=0}^{T} \left(C_H(t) + C_P(t) + C_T(t) + C_X(t)\right), \text{where} \quad (2)$$

C_H is holding cost, C_P is purchase cost, C_T transport cost, and C_X stockout penalties, and

$$C_H(t) = \sum_{n=1}^{N} \sum_{k=0}^{K} C_H(t,n,k), \quad (3)$$

summed over N nodes and K stocks,

[1] This work was supported in part by the National Science Foundation under grant ECS-9904378.

$$C_H(t,n,k) = P_H(n,k)\,Q_H(t,n,k) \text{ if } Q_H(t,n,k) \geq 0, \quad 0 \text{ else, where} \tag{4}$$

P_H is holding price per Q_H unit quantity on hand;

$$C_P(t) = \sum_{n=1}^{N} \sum_{k=0}^{K} C_P(t,n,k) \tag{5}$$

$$C_P(t,n,k) = \sum_{n=1}^{N} \sum_{k=0}^{K} P_P(k) Q_P(t,n,k) \text{ if } Q_P(t,n,k), \quad 0 \text{ else,} \tag{6}$$

where $P_P(k)$ is purchase price per Q_P unit quantity purchased;

$$C_T(t) = C_{TF}(t) + C_{TO}(t) + C_{TX}(t), \text{ where} \tag{7}$$

C_{TF} is fixed transport cost (the cost of owning the transport resource over the decision period), C_{TO} is operating cost (only for transport units actually employed), and C_{TX} is the penalty for transport shortfalls, and

$$C_{TF}(t) = \sum_{a=0}^{A} \sum_{m=0}^{M} C_{TF}(t,a,m), \tag{8}$$

summed over A arcs and M modes of transport, where

$$C_{TF}(t,a,m) = P_{TF} T_{Cap}(t,a,m) \text{ if } T_{Cap}(t,a,m) > 0, \quad 0 \text{ else,} \tag{9}$$

where P_{TF} is the price per T_{Cap}, the unit capacity of transport hired during the decision period;

$$C_{TO}(t) = \sum_{a=0}^{A} \sum_{m=0}^{M} C_{TO}(t,a,m), \text{ where} \tag{10}$$

$$C_{TO}(t,a,m) = P_{TO}(T_{CAP}(t,a,m) - T_{CAV}(t,a,m)), \quad \text{if } T_{CAP}(t,a,m) > T_{CAV}(t,a,m), \quad 0 \text{ else,} \tag{11}$$

where P_{TO} is the cost of operating the units which are on the road (the difference between capacity T_{Cap} and capacity available T_{Cav});

$$C_{TX}(t) = \sum_{a=0}^{A} \sum_{m=0}^{M} C_{TX}(t,a,m), \tag{12}$$

$$C_{TX}(t,a,m) = P_{TX}\,(Q_S(t,a,m) - T_{Cav}(t,a,m)), \quad \text{if } Q_S(t,a,m) > T_{Cav}(t,a,m), \quad 0 \text{ else, where} \tag{13}$$

Q_S is the quantity provided by a supply node n_s, which is related to Q_R the quantity requested by a demand node, n_d

$$Q_S(t,a,m) = Q_R(t,n_d,k) \quad \text{if } Q_H(t,n_s,k) > Q_R(t,n_d,k), \quad Q_H(t,n_s,k) \text{ else,} \tag{14}$$

the quantity requested being driven by the two policies which control the reorder point, RP and the order up-to point UT, such that

$$Q_R(t,n_d,k) = UT(t,n_d,k) - (Q_H(t,n_d,k) + Q_O(t,n_d,k)) \quad \text{if } Q_H(t,n_d,k) < RP(t,n_d,k) - (Q_H(t,n_d,k) + Qo(t,n_d,k)), \quad 0 \text{ else, where} \tag{15}$$

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

$$C_X(t) = \sum_{n=1}^{N} \sum_{k=0}^{K} C_X(t,n,k), \tag{17}$$

$$C_X(t,n,k) = \sum_{n=1}^{N} \sum_{k=0}^{K} P_X Q_H(t,n,k), \quad \text{if } Q_H(t,n,k) < 0, \quad 0 \text{ else,} \tag{18}$$

where P_X is the penalty for stockout.

For small problems, minimizing the cost function is often done using mixed-integer linear programming (LP) techniques[9]. Our study instead uses a Genetic Algorithm as a tool for finding an initial policy set that will minimize costs, reserving the use of the LP as a comparison tool. Because the fitness terrain is spiky, the search problem is difficult, and the GA solution is only quasi-optimal. In the specific case of the problem used here, the cost of the LP-derived optimal starting policy set was only 28% that of the GA solution. The purpose of the Genetic Algorithm is twofold. Not only does it search for a quasi-optimal starting point, but it also saves both the optimal and off-optimal data generated by the search process for use as a source of input/output pairs to train the Plant Model NN (section III).

The evaluation function for the GA is a discrete event simulation. Business constraints necessary to the operation of the simulation are handled by repairing the chromosome as it is being created. Other business rules are enforced by adjustment of the penalties associated with breaking them.

III. METHODOLOGY

Small scale artificially constrained examples of our inventory and transportation problem can be solved exactly using Dynamic Programming [3][4]. Unfortunately, very few supply nodes, stock levels and transport arcs can be included before the classical approach becomes intractable due to the "curse of dimensionality". Over the last decade, a family of approximate dynamic programming techniques utilizing adaptive critics has been developed, that do not suffer from dimensional blow up.

In Dynamic Programming one develops an optimal policy by comparing the costs of all alternative actions at all accessible points in state space through time. This search is made efficient by limiting the options using the principle of optimality: that an optimal trajectory has the property that no matter how an intermediate point is reached, the rest of the trajectory must coincide with an optimal trajectory as calculated with the intermediate point as the starting point.

Adaptive critic based approximate dynamic programming methods start with the assumption that the optimal policy can be written as a continuously differentiable function of the state variables and some number of policy parameters. A critic function is then constructed that estimates the value

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 long term cost of the policy specified by the control function's parameter values, the gradient of the critic function can be used to adjust the policy 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][10][11] and more recently using fuzzy systems[13].

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. A *secondary* utility function is then formed

$$J(t) = \sum_{k=0}^{\infty} \gamma^k U(t+k), \tag{19}$$

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

$$J(t) = U(t) + \gamma J(t+1). \tag{20}$$

A promising collection of approximation techniques based on estimating the function *J(t)* using this identity with neural networks as function approximators was proposed by Werbos [19][20]. As the gradient of the estimated *J(t)* is used to train or tune the control policy, some techniques use critics that estimate the derivatives of *J(t)* instead of the function value itself.

In Dual Heuristic Programming (DHP) the critic's outputs are estimates of the derivatives of *J(t)*. This method utilizes two distinct training loops, one for the control policy and one for the critic estimator. The control policy is adjusted to optimize the secondary utility function *J(t)* for the problem context. Since the controller outputs control actions u(t), a gradient based learning algorithm requires estimates of the derivatives $\frac{\partial J(t)}{\partial u_i(t)}$ for controller training. 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, i.e. $\lambda_i(t) = \frac{\partial J(t)}{\partial R_i(t)}$, we differentiate both sides of Bellman's Recursion

$$\frac{\partial}{\partial R_i(t)} J(t) = \frac{\partial}{\partial R_i(t)} \left(U(t) + \gamma J(t+1) \right), \tag{21}$$

to get the identity used for critic training (in tensor notation)

$$\lambda_i(t) = \frac{\partial U(t)}{\partial R_i(t)} + \frac{\partial U(t)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial R_i(t)} + \gamma \lambda_k(t+1) \left[\frac{\partial R_k(t+1)}{\partial R_i(t)} + \frac{\partial R_k(t+1)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial R_i(t)} \right]. \tag{22}$$

To evaluate the right hand side of this equation we need a model of the system dynamics that includes all the terms from the Jacobian matrix of the coupled plant-controller system, e.g. $\frac{\partial R_j(t+1)}{\partial R_i(t)}$ and $\frac{\partial R_j(t+1)}{\partial u_i(t)}$. (23)

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

$$\frac{\partial J(t)}{\partial u_k(t)} = \frac{\partial U(t)}{\partial u_k(t)} + \gamma \sum \lambda_i(t+1) \frac{\partial R_i(t+1)}{\partial u_k(t)}. \tag{24}$$

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 *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 optimization of the control policy. It is also possible to do both optimization processes simultaneously [8][12].

As this technique relies on gradient based optimization of *J(t)*, it inherently suffers from the problem of (unsatisfactory) local optima. Global optimization of *J(t)* in general is subject to the "No Free Lunch Theorem". What approximate dynamic programming techniques offer is a tractable method 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 control law. 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 seek to avoid this problem by starting with a quasi-optimal initial policy that is already in some sense satisfactory.

IV. IMPLEMENTATION

To implement Dual Heuristic Programming to refine our quasi-optimal policies we need:

a) a differentiable secondary utility function,
b) a system model capable of approximating the partial differential equations $\frac{\partial R_j(t+1)}{\partial R_i(t)}$ and $\frac{\partial R_j(t+1)}{\partial u_i(t)}$, and
c) a method for expressing policies using a differentiable function of some set of policy parameters.

Unfortunately, our current problem context does not come thus equipped. We are therefore required to make a number of approximations.

A. Strategic Utility Function

The strategic utility function U(t) defines the effectiveness of the controller. In many control situations this is a simple error function that measures how far the system is from a desired state. The current problem is a bit more complex. The utility function is set equal to the cost equation (1), and objective is to set policies so as to minimize the total cost of executing those policies. The utility function therefore describes the impact on costs of each aspect of that execution.

Complicating the situation further is the fact that policy execution is discontinuous – if, for example, an order is created, then purchase costs are incremented, if no order is created, then they are not. However, in order to apply the DHP methodology, we must have a differentiable J(t), and therefore need a differentiable U(t), and in order to do that, we must make the approximations described below.

Rather than detail all elements of U(t), we will limit this presentation to the three cost equations (4, 6, and 13) presented above, the first example being holding cost, C_H.

The curve representing holding cost has a 'knee' at $Q_H = 0$, so the derivative is not continuous at that point:

$$\partial C_H / \partial Q_H = \begin{cases} P_H \text{ if } Q_H > 0, \\ 0 \text{ else,} \end{cases} \quad (25)$$

so, for the purpose of calculating derivatives only, we use the approximation:

$$C_H = P_H Q_H \text{sig}(Q_H), \quad (26)$$

(where sig(x) is the logistic sigmoid function $\text{sig}(x) = 1/(1+e^x)$), which has the partial derivative

$$\partial C_H / \partial Q_H = P_H Q_H \text{sig}(Q_H)(1 - \text{sig}(Q_H)). \quad (27)$$

In the case of purchase cost, the function itself is non-continuous (a purchase is only made when the associated control conditions are met), so when we calculate the partial derivatives we start from:

$$C_P = P_P Q_P \text{sig}((UT - (Q_H + Q_O))(\text{sig}(RP - (Q_H + Q_O))). \quad (28)$$

Similarly, the starting point for calculation of the derivatives of the transport shortfall penalty, is the approximation:

$$C_{TX} = P_{TX}(UT - (Q_H + Q_O))(\text{sig}(RP - (Q_H + Q_O)) - T_{Cav}) \\ (\text{sig}(UT - (Q_H + Q_O)) - T_{Cav}). \quad (29)$$

B. System Identification

Creation of a function which describes the responses of the plant to control and state inputs is a task called System Identification. In this study we train a simple MLP Neural Net as a model of the plant responses, then use the net as an ordered array of derivatives.

Training the Plant Model NN is a straightforward task, provided one has a suitable collection of I/O pairs. There are a number of ways of obtaining such pairs -- all involve providing a set of inputs which span the search space, processing the plant response, and collecting the resultant output. Two ways that were considered and rejected are Monte Carlo search, and complete enumeration. The first involves generating inputs at random, feeding them to the simulation, and saving the outputs. The second involves complete enumeration of all possible inputs at some sampling interval. The first has the drawback that it might miss significant features of the fitness landscape, the second that it will waste time on *non*-significant feature.

The third method, used here, is to save the output generated by the GA during its search process. If all output is saved, even that generated by less-than-optimal inputs, then the data set so obtained will not only span the search space, it will also provide a higher proportion of I/O pairs in the vicinity of good solutions. We believe this is the first time such an approach has been used.

Two data sets were collected for this study. The training set totaled 90,000 records, and the test set contained 108,000 records. The data were collected by running the GA with a population of 100 for the standard 1,000 generations, and saving the I/O data every 97 generations.

The plant model NN has 62 inputs (40 state variables, 18 control variables, and 4 exogenous demand variables), a 102-neuron hidden layer, and 40 state variable outputs. Forty-eight examples of the NN were trained for 20 full epochs. The resulting nets had a training set RMSE = 0.60 and a test set RMSE = 0.62. (n=48, coefficient of variance < 0.000 for both). This is not a sterling result, but Shannon [12] has shown that very poor predictive models can provide outputs that are good enough for an adaptive critic to use.

Once the trained plant model NN is in place, it may be used to provide information on the partial derivatives $\frac{\partial R(t+1)}{\partial u(t)}$ and $\frac{\partial R(t+1)}{\partial R(t)}$ by first passing the current state and control information R_t and u_t forward through the NN and saving the slope of the activation function at each node. Then, for each R, a 1 is backpropagated from the output node, with all other inputs at the output nodes held to zero. The derivative information will appear as outputs from the input nodes, R and u.

C. Differentiable Policy Representation

If we are to do gradient based search across a policy space, we have to be able to express those policies in such a way that gradient is meaningful. To do this, we are going to use another artificial neural network – the Action Net – since a NN can be thought of as a differentiable function of the combination of its inputs and network weights. This will allow us to produce deltas from the initial policy variable set points, giving us outputs that are both meaningful and differentiable.

V. RESULTS

The experiments described below tested the neural control system against the quasi-optimal policies found by the GA.

The GA policies were developed based on a fixed demand schedule, and the amounts demanded were based on the expected value of the demand for a specific stock at a specific node. Demand node 1, for example, required 2.0 units of stock 0 every timestep for 90 simulation days.

The NN controller, which requires persistence of excitation to train well, was trained using Poisson-distributed demand with a stationary mean equal to the value used by the GA.

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

Both the fixed, GA-derived and the NN controller-adjusted policies were then tested using demand schedules that ran for 360 simulation days. There were three test data sets, labeled *Baseline, Delta Demand (DD),* and *Increasing Average Demand (IAD).* Baseline was the training data set, extended to 360 days. Delta Demand caused a 10% step increase in demand, which held for the full 360-day test period. Increasing Average Demand added a fixed increment to demand each timestep, sufficient to raise the demand at the 360 day point by 20% over the starting demand. All three demand schedules had a Poisson distribution laid on top of the underlying trend.

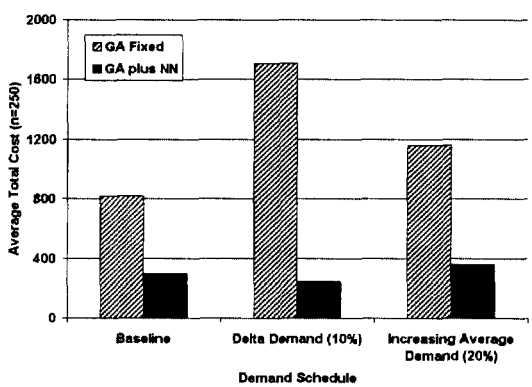

Figure 2. Summary of GA/NN Controller Comparisons. Performance comparisons of fixed quasi-optimal GA-derived policies (GA, dark columns), with adaptive Neural Control policies (NN, light columns) for three different demand schedules: stationary (Baseline), Increasing Average Demand (IAD), and Delta Demand (DD). All demand schedules exhibit Poisson-distributed noise.

Note that the comparison to be made is not really between the NN controller and the GA-derived policies. Instead, it is between fixed policies (however arrived at) and learning policies (whatever their start point). The results (Figure 2) can be summarized by stating that, in this instance, the NN adaptive control solution was able to improve on the fixed policy set. The NN controller results were highly consistent – the coefficient of variation (Std Dev/Avg) in all three test series was 0.000% (n=250).

A. Baseline

This set of experiments (Figure 3) was designed to establish a reference behavior for the system.

First, the GA-derived policies were used, without modification, to operate the simulation on a 360-day run. Then the NN controlled-simulation was run against the same demand schedule, using the adaptive critic techniques discussed above to correct the way the NN responded to changes in the environment. At the end of the 360-day generalization test period the NN controller provided a savings of 64 percent over the GA-derived policies.

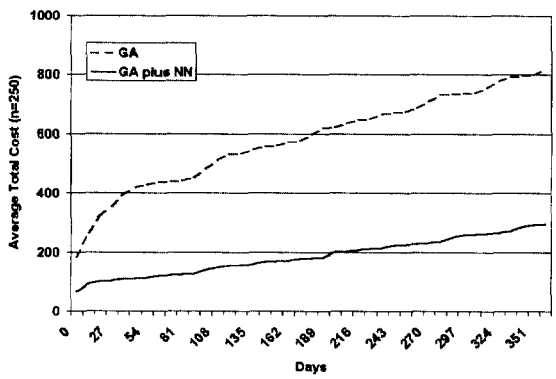

Figure 3. Average (n = 250) cumulative total cost of logistics system with fixed, GA-derived policies (dashed line) and with policies adjusted by the neural-controller (solid line). Baseline demand schedule.

B. Increasing Average Demand

The second series of tests (Figure 4.) compared the effectiveness of the (unchanging) GA-generated policies with those of the NN controller when operating in a changing environment In this case, the change was a fixed increase in demand at every timestep throughout the period, sufficient to increase demand by 20% over the 360-day test period. Final cost reduction at the end of the 360-day generalization period was 69%.

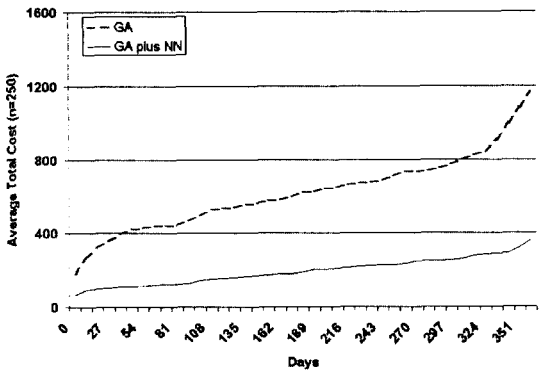

Figure 4. Average (n = 250) cumulative total cost of logistics system with fixed, GA-derived policies (dashed line) and with policies adjusted by the neural-controller (solid line). Baseline demand schedule with increasing average demand.

C. Delta Demand

The final test series (Figure 5) compared the effectiveness of the (unchanging) GA-generated policies with those of the

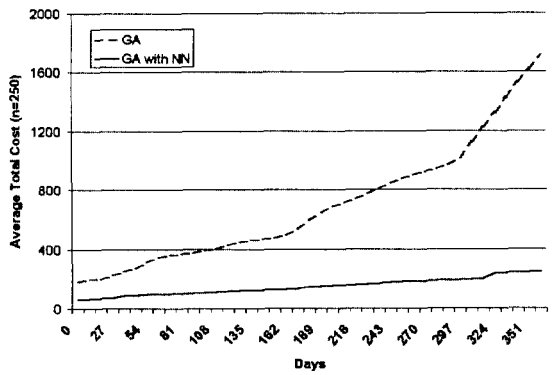

Figure 5. Average (n = 250) cumulative total cost of logistics system with fixed, GA-derived policies (dashed line) and with policies adjusted by the neural-controller (solid line). Baseline demand schedule plus 10% step increase.

NN controller when operating in an environment where demand had been increased by a 10% step which persisted throughout all time periods. Final cost reduction at the end of the 360 day generalization period was 69%.

VI. CONCLUSIONS

We have demonstrated the possibility of optimal control of physical inventory systems in both stationary and non-stationary demand conditions, based on the combined application of evolutionary search and adaptive critic controller training. As part of this demonstration, we showed that adaptive critic based approximate dynamic programming techniques based on plant-controller Jacobeans can be used with systems characterized by discrete valued states and controls. In addition, it has proven feasible to use off-optimal data from the evolutionary search to provide system identification inputs for a plant model neural net. Improvements over a fixed, quasi-optimal policy found using a genetic algorithm, average 66% under conditions both of stationary and non-stationary demand in a high penalty environment.

VII. REFERENCES:

[1] Aneja, Y. and K. Nair (1979). "Bicriteria Transportation Problem." *Management Science*. 25: 73-78.

[2] Barto, A., Sutton, R. and Anderson, C. (1983) "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions On Systems, Man, And Cybernetics* Vol. SMC-13, No. 5, pp. 834-846.

[3] Bellman, R. (1957). *Dynamic Programming*. Princeton, Princeton University Press.

[4] Bellman, R.E., and Dreyfus, S. (1962) *Applied Dynamic Programming*. Princeton, Princeton University Press.

[5] Bit, A., M. Biswal, et al. (1993). "Fuzzy programming approach to multiobjective solid transportation problem." in *Fuzzy Sets and Systems*. 57: 183-194.

[6] Gullu, R. and N. Erkip (1996). "Optimal allocation policies in a two-echelon inventory problem with fixed shipment costs." *International Journal of Production Research*, . 46-47: 311-321.

[7] Jonsson, H., E. Silver, et al. (1986). "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.

[8] Lendaris, G. and Shannon, T. (1998) "Application Considerations for the DHP Methodology," in *Proceedings of the International Joint Conference on Neural Networks '98* (IJCNN '98) Anchorage, AK, IEEE, May.

[9] Oh, S. and A. Haghani (1997). "Testing and Evaluation of a Multi-Commodity Multi-Modal Network Flow Model for Disaster Relief Management." *Journal of Advanced Transportation*. 31: 249-282.

[10] Prokhorov, D., *Adaptive Critic Designs and their Application*, Ph.D. Dissertation, Department of Electrical Engineering, Texas Tech University, 1997.

[11] Prokhorov, D. & D. Wunsch, "Adaptive Critic Designs", *IEEE Transactions On Neural Networks*, vol.8(5), 1997, pp. 997-1007.

[12] Shannon, T., (1999) "Partial, Noisy and Qualitative Models for DHP Adaptive Critic Neuro-control," in *Proceedings of the International Joint Conference on Neural Networks '99* (IJCNN'99), Washington DC, July.

[13] Shannon, T.T. & Lendaris, G.G., (2000), "Adaptive Critic Based Approximate Dynamic Programming for Tuning Fuzzy Controllers", in *Proceedings of IEEE-FUZZ 2000, IEEE.*

[14] Werbos, P. (1992) "Neurocontrol and Supervised Learning: An Overview and Evaluation," in White, D. and Sofge, D., *Handbook of Intelligent Control*. New York, Van Nostrand Rheinhold.

[15] Werbos, P. (1995) "Optimization Methods for Brain-like Intelligent Control," *Proceedings of the 34th Conference on Decision and Control*. IEEE Press, pp. 579-584.

[16] Werbos, P. (1990) "Neurocontrol and related techniques," in Maren, A., Harston, C., and Pap, R. (eds.) *Handbook of Neural Computing Applications*. Academic Press, Inc., New York., pp. 345-380.

[17] Zhang, V. (1996). "Ordering Policies for an Inventory System with Three Supply Modes." *Naval Research Logistics*. 43: 691-708.