Reinforcement Control via Action Dependent Heuristic Dynamic Programming

K. Wendy Tang
Department of Electrical Engineering
SUNY at Stony Brook
Stony Brook, NY 11794-2350
wtang@ee.sunysb.edu

Govardhan Srikant
Department of Electrical Engineering
SUNY at Stony Brook
Stony Brook, NY 11794-2350

Abstract

Heuristic Dynamic Programming (HDP) is the simplest kind of Adaptive Critic which is a powerful form of reinforcement control [1]. It can be used to maximize or minimize any utility function, such as total energy or trajectory error, of a system over time in a noisy environment. Unlike supervised learning, adaptive critic design does not require the desired control signals be known. Instead, feedback is obtained based on a critic network which learns the relationship between a set of control signals and the corresponding strategic utility function. It is an approximation of dynamic programming [2].

Action Dependent Heuristic Dynamic Programing (ADHDP) system involves two subnetworks, the Action network and the Critic network. Each of these networks includes a feedforward and a feedback component. A flow chart for the interaction of these components is included. To further illustrate the algorithm, we use ADHDP for the control of a simple, 2-D planar robot.

1. Introduction

Neurocontrol is defined as the use of neural networks to emit control signals for dynamic systems. There are five basic schemes: the *supervised control, direct inverse control, neural adaptive control, Backpropagation of Utility*, and *Adaptive Critic Designs*. Werbos [2] provided a detailed summary of the five schemes including the pros and cons of each method. Of these techniques, both Backpropagation of Utility and Adaptive Critic are capable of maximizing a utility function. In an earlier report [3], we have demonstrated the application of Backpropagation of Utility. In this paper, we concentrate on Adaptive Critic Design.

Adaptive Critic is more powerful than Backpropagation of Utility as it is capable of optimal control in a noisy, nonlinear, or nonstationary environment [4]. Werbos [5] recently showed how a critic network can be used to solve the generalized maze problem, a difficult optimization problem, using a recurrent critic design. Within the scope of Adaptive Critic Designs, there are different levels of applications, including Heuristic Dynamic Programming (HDP) and the more advanced Dual Heuristic Programing (DHP) [4]. In a recent article, Prokhorov and Wunsch [6] demonstrated applications of the more advanced DHP and its variations.

In this paper, our objective is to illustrate the application of Action Dependent Heuristic Dynamic Programming (ADHDP), a simpler variation of HDP. We show that this simpler design is a cost-efficient alternative for less complex problems. This article is organized as follows: Section 2 is a description of the *Action Dependent Heuristic Dynamic Programming (ADHDP)*. Section 3 illustrates how the algorithm is used for the control of a 2-D planar robot. Finally, conclusions and a summary are included in Section 4.

2. ADHDP

The objective of the Adaptive Critic design is to provide a set of *control signals* to a dynamic system to maximize a utility function over time. The utility function can be total energy, cost-efficiency, smoothness of a trajectory, etc. For expository convenience, we assume the notation $u(t)$ for the control signal, and $U(t)$ for the utility function which is usually a function of the system state.

The simplest form of Adaptive Critic Design is Heuristic Dynamic Programming (HDP), of which the Action Dependent HDP (ADHDP) is a simple variation. ADHDP consists of two subnetworks, an *Action*

0-7803-4122-8/97 $10.00©1997 IEEE

network and a *Critic* network. The Action network is responsible for providing the control signal to maximize or minimize the utility function. This goal is achieved through adaptation of the internal weights of the Action network. Such adaptation is accomplished through backpropagation of various signals. For each iteration, there are feedforward and feedback components. In the feedforward mode, the Action network outputs a series of control signals, $u(t), t = 1, \ldots, T$ whereas adaptation of the internal weights is accomplished through the feedback mode.

The Critic network provides the relationship between a given set of control signals and the strategic utility function, $J(t)$. Its function is two folded: (i) in the feedforward mode, it predicts the strategic utility function for a given set of control signals from the action network; and (ii) in the feedback mode, it provides feedback to the action network on how to change its control signal to achieve maximum or minimum utility. The basic idea is that once the critic network learns the relationship between a given set of control signals and the corresponding strategic utility function, it can be used to provide proper feedback to the action network to generate the desired control signals. Figure 1 shows a block-diagram representation of the system. The dashed lines represent the feedback mode, or derivative calculations. The description and pseudocodes of the forward and backward components can be found in [3, 7] and are not repeated here because of page limitation.

In other words, the application of ADHDP involves repeated iterations of two steps: (i) training of the Critic Network to learn the relationship between a given set of control signals and the corresponding strategic utility function; and (ii) training of Action network to generate a set of control signals. The interaction of these components are described in a flow chart in Figure 5. For practical application of ADHDP, it is often necessary that we start with some initial control signals. These control signals can be generated by the classical PID controller or through Backpropagation of Utility [3]. Once such initial control signals are generated, an Action network is trained to provide these initial signals and an utility function is computed from these signals. Then the HDP algorithm begins by repeated iterations of several components as presented in Figure 5 and to be described in the next paragraph. Such iterations repeat until the utility function is less than or greater than a threshold value, depending on whether we are minimizing or maximizing the utility function.

Referring to Figure 5, for each iteration of the ADHDP, the forward component of the Action network is used to generate a set of control signals, $u(t), t = 0, 1, \ldots T-1$. The detailed description of the forward component can be found in [3, 7] and is not repeated here. These initial control signals are used to compute the current utility and create a target strategic utility function J for the critic network. More specifically,

$$J(t) = U(t) + mJ(t+1), \quad t = T-2, \ldots, 0 \qquad (1)$$

where $0 \leq m \leq 1$ is a scalar and $U(t)$ is the utility for time t and $J(T-1) = U(T-1)$.

Once a target J is created, the Critic network is trained to match its output with J. When such training is complete, the Critic network can be used to provide the derivatives of J through backpropagation. This is achieved through a simple modification of the backward components. Pseudo code for such feedback is included in [3, 7] and is not repeated here. This feedback of the derivative of J is then used in the backward components of Action network to find the weight gradient. Once the weight gradient is computed, standard weight adaptation algorithms such as Delta-Bar-Delta [8] or the RPROP [9] can be used to update the internal weights of the Action network. The Action network can then be used to generate a new set of control signals and the algorithm repeats until the utility is less than or bigger than a prescribed threshold value.

3. An Example: 2-D Planar Robot

As an example, we consider a simple planar manipulator with two rotational joints (Figure 2). We assume, without loss of generality, that the robot links can be represented as point-masses concentrated at the end of the link. The link masses and lengths are respectively: $m_1 = m_2 = 0.1$ kg and $d_1 = d_2 = 1$ m. This simple dynamic system is governed by the dynamic equations:

$$\begin{aligned}
\tau_{d1}(t) = \; & [(m_1 + m_2)d_1^2 + m_2 d_2^2 \\
& + 2m_2 d_1 d_2 \cos(\theta_2(t))]\ddot{\theta}_1(t) \\
& + [m_2 d_2^2 + m_2 d_1 d_2 \cos(\theta_2(t))]\ddot{\theta}_2(t) \\
& - 2m_2 d_1 d_2 \sin(\theta_2(t))\dot{\theta}_1(t)\dot{\theta}_2(t) \\
& - m_2 d_1 d_2 \sin(\theta_2(t))\dot{\theta}_2^{\,2}(t) \\
& + (m_1 + m_2) g d_1 \sin(\theta_1(t)) \\
& + m_2 g d_2 \sin(\theta_1(t) + \theta_2(t)) \\
\tau_{d2}(t) = \; & [m_2 d_2^2 + m_2 d_1 d_2 \cos(\theta_2(t))]\ddot{\theta}_1(t) \\
& + m_2 d_2^2 \ddot{\theta}_2(t) \\
& - 2m_2 d_1 d_2 \sin(\theta_2(t))\dot{\theta}_1(t)\dot{\theta}_2(t) \\
& - m_2 d_1 d_2 \sin(\theta_2(t))\dot{\theta}_1^{\,2}(t) \\
& + m_2 g d_2 \sin(\theta_1(t) + \theta_2(t))
\end{aligned} \qquad (2)$$

where τ_{d1}, τ_{d2} are the desired joint forces needed to drive the manipulator to the prescribed trajectory, and $g = 9.81\text{m/s}^2$ is the gravitational constant. We consider that initially, at time $t = 0$ second, the state of the manipulator is

$$\theta_1(t=0) = \dot{\theta}_1(t=0) = \ddot{\theta}_1(t=0) = 0$$
$$\theta_2(t=0) = \dot{\theta}_2(t=0) = \ddot{\theta}_2(t=0) = 0.$$

The HDP's task is to generate a series of control signals $u_1(t) = \tau_1(t)$, $u_2(t) = \tau_2(t)$, $t = \delta t, 2\delta t, \ldots, t_f = T\delta t = 5$ seconds ($\delta t = 0.05, T = 100$) to drive the manipulator from the initial configuration $\theta_0 = 0°$ for both joints to $\theta_f = \theta(t = t_f) = 60°, 90°$ for Joint 1 and 2 respectively. The desired trajectory for both joints is specified by the quintic polynomials [10]:

$$\begin{aligned}
\theta_d(t) = \; & \theta_0 + 10(\theta_f - \theta_0)(t/t_f)^3 - 15(\theta_f - \theta_0)(t/t_f)^4 \\
& +6(\theta_f - \theta_0)(t/t_f)^5 \\
\dot{\theta}_d(t) = \; & 30(\theta_f - \theta_0)(t^2/t_f^3) - 60(\theta_f - \theta_0)(t^3/t_f^4) \\
& +30(\theta_f - \theta_0)(t^4/t_f^5) \\
\ddot{\theta}_d(t) \; & = 60(\theta_f - \theta_0)(t/t_f^3) - 180(\theta_f - \theta_0)(t^2/t_f^4) \\
& +120(\theta_f - \theta_0)(t^3/t_f^5)
\end{aligned} \tag{3}$$

We used Backpropagation of Utility [3] to generate a set of initial forces. These initial forces (τ_1, τ_2) and the corresponding desired forces (τ_{d1}, τ_{d2}) are shown in Figure 4. We emphasize that the desired forces are used here for comparison only. Their values are not used in HDP learning. For tracking control, the utility function is:

$$U(t) = (\theta(t) - \theta_d(t))^2 + (\dot{\theta}(t) - \dot{\theta}_d(t))^2 + (\ddot{\theta}(t) - \ddot{\theta}_d(t))^2$$

where $\theta = [\theta_1 \; \theta_2]^{\mathrm{T}}$ and $\theta_d = [\theta_{d1} \; \theta_{d2}]^{\mathrm{T}}$ are the actual and desired joint position.

After approximately 200-250 iterations, the final forces obtained from HDP becomes very close to the desired values. Figure 3 plots the absolute errors of these forces ($|\tau_1 - \tau_{d1}|$, $|\tau_2 - \tau_{d2}| < 0.05$). For further investigation, we have applied the ADHDP algorithm to a second trajectory that drives the two joints to $\theta_f = 30°, 60°$, respectively. Again, the initial forces are generated by Backpropagation of Utility. Figures 6-7 show the initial and desired forces and the final force error after ADHDP learning.

In our implementation of HDP, we found that it is helpful to use multiple critic networks for different time horizons. We used 10 Critic networks, each learning ten different sampling points with two additional overlapping points at the begining and end of a time period. The action and critic networks are two-hidden layered networks. For the action network there are 5 hidden nodes in each layer while each critic network has 3 hidden nodes in each layer. Furthermore, in our implementation of the target J function (Equation 1), we set $m = 0$. Werbos has suggested that for more complex systems, it is necessary to set $m = 0$ initially and then gradually relax m to a non-zero value [11].

4. Conclusions

Backpropagation of Utility and *Adaptive Critic Designs (ADCs)* are potentially powerful techniques as they are capable of maximizing and minimizing a utility function over time. Action Dependent Heuristic Dynamic Programming (ADHDP) is the simplest ADC.

In our previous work [3], we have demonstrated how to use Backpropagation of Utility for a simple control problem. However, its main drawback is that it requires the exact emulation of the dynamic system by a neural network, the *model network* [3]. Adaptive Critic Designs, as an approximation of dynamic programming, are capable of optimal control in a noisy environment.

In this paper we showed that the simple ADHDP combined with Backpropagation of Utility is a cost-effective alternative for less complex systems. ADHDP is composed of two subnetworks, the *Action* net and the *Critic* net. While the former generates control signals for the system, the latter provides a judgement or feedback to evaluate how effective the generated controls signals are. Such feedback, in the form of derivative of the judgement, is used to update the internal weights of the Action net and eventually drive the control signals to their desired values.

As an illustration, we used the ADHDP design for control of a 2-D planar robot. Albeit a simple case, the 2-D example involves various dynamic effects and coupling of joint forces. In our previous attempt of using Backpropagation of Utility for controlling such system, we are unable to adequately emulate the system without prolonged training of the model network. In this paper, we demonstrated that even though the model net used by Backpropagation of Utility is not adequate, the ADHDP design is capable of driving the control signals to the desired values through proper feedback of reward and punishment, the strategic utility function J (Equation 1). A flow chart is provided to illustrate the interaction of the various components of the ADHDP design.

5. Acknowledgement

The authors acknowledge and appreciate discussions with Paul Werbos and Daniel Prokhorov. Suggestions from the anonymous reviewers to improve the paper are greatly appreciated. This research was supported by the National Science Foundation under Grant No. ECS-9626655.

References

[1] Paul J. Werbos. Approximate Dynamic Programming for Real-Time Control and Neural Modeling. In White D, A and D.A. Sofge, editors, *Handbook of Intelligent Control*, pages 493–525. Van Nostrand Reinhold, 1992.

[2] Paul J. Werbos. Neurocontrol and Supervised Learning: an Overview and Evaluation. In D.A. White and D.A. Sofge, editors, *Handbook of Intelligent Control*, pages 65–89. Van Nostrand Reinhold, 1992.

[3] K. W. Tang and Girish Pingle. "Neuro-Remodeling via Backpropagation of Utility.". *Journal of Mathematical Modeling and Scientific Computing*, May 1996. Accepted for Publication.

[4] Paul J. Werbos. A Menu of Designs for Reinforcement Learning Over Time. In R.S. Sutton W. T Miller, III and P.J. Werbos, editors, *Neural Networks for Control*, pages 67–95. MIT Press, 1990.

[5] Paul J. Werbos and X. Pang. "Generalized Maze Navigation: SRN Critic Solve What Feedforward or Hebbian Nets Cannot". In *World Congress on Neural Networks*, pages 88–93, San Diego, CA, September 15-18 1996.

[6] D.V. Prokhorov and D.C. Wunsch II. "Advanced Adaptive Critic Designs". In *World Congress on Neural Networks*, San Diego, CA, September 15-18 1996.

[7] Paul J. Werbos. "Backpropagation Through Time: What It Does and How to Do It". *Proceedings of the IEEE*, 78(10):1550–1560, October 1990.

[8] Robert A. Jacobs. "Increased Rates of Convergence Through Learning Rate Adaptation". *Neural Networks*, 1:295–307, 1988.

[9] M. Riedmiller and H. Braun. "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm". In *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, April 1993.

[10] J. J. Craig. *Introduction to Robotics, mechanics and Control.* Addison-Wesley Publishing Co., New York, NY, 1986.

[11] Paul J. Werbos. Verbal communication, 1996.

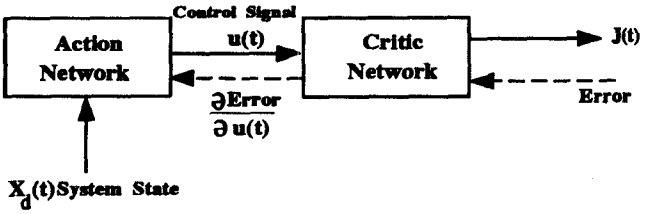

Figure 1: A Simple HDP System

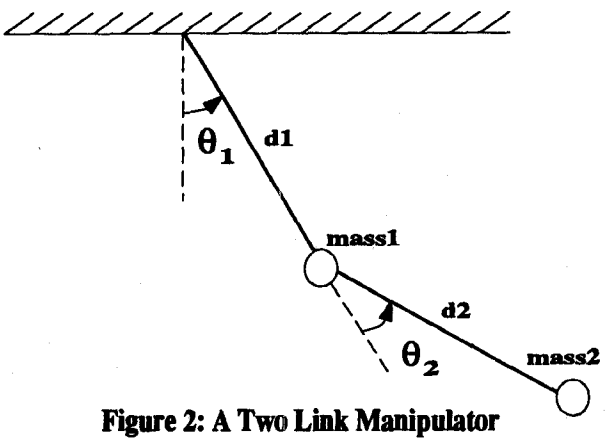

Figure 2: A Two Link Manipulator

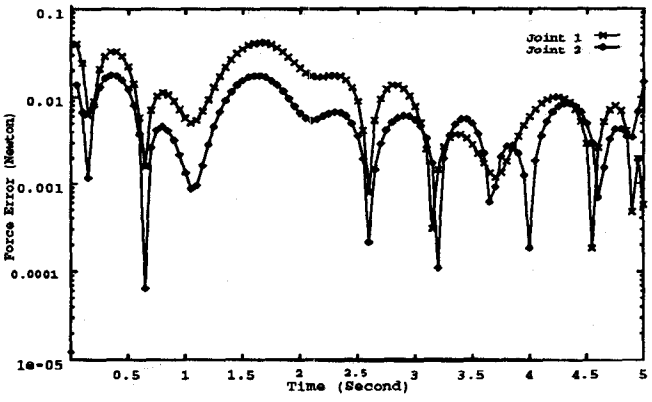

Figure 3: Resultant Force Error After HDP.

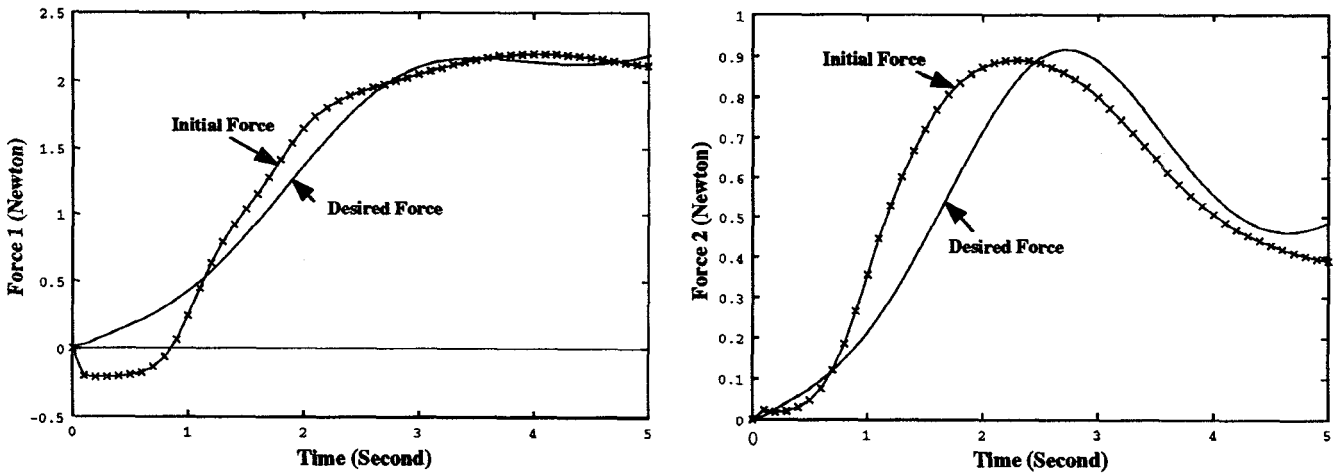

Figure 4: Initial and Desired Forces

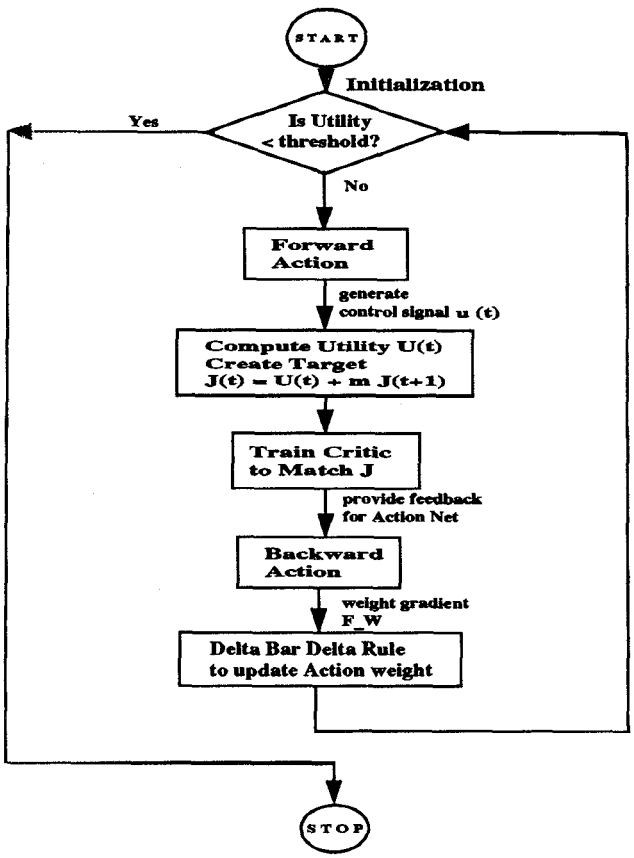

Figure 5: Flow Chart for HDP Training

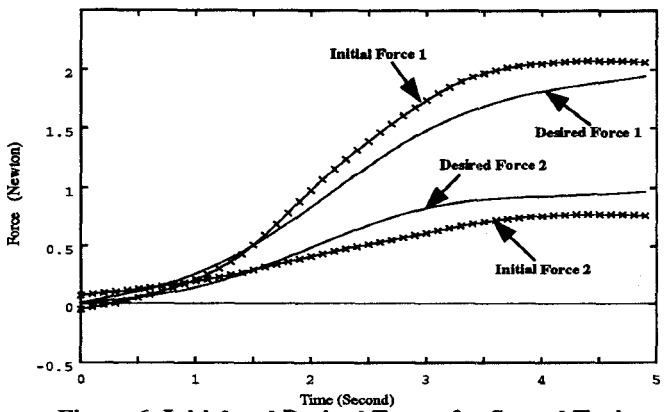

Figure 6: Initial and Desired Forces for Second Trajectory

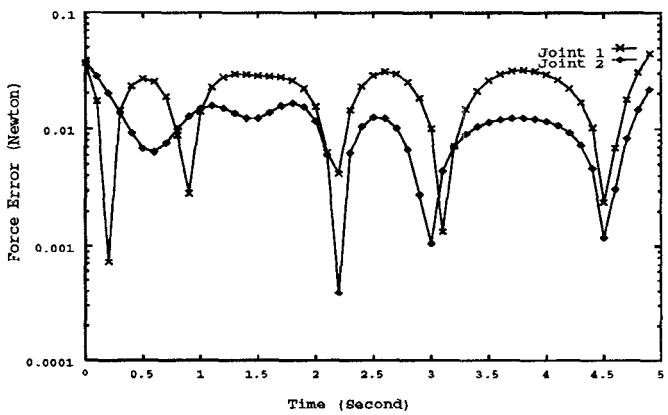

Figure 7: Force Error for Second Trajectory