Chapter 10
Other Applications of ADP

10.1 Introduction

As is known, both modern wireless networks and automotive engines are complex non-linear dynamical systems. The corresponding optimal control problems have been investigated for many years by many researchers. However, there are still not results referring to the optimal control based on the ADP method. Due to the advantages of ADP in solving the optimal feedback control in a forward-time fashion, the ADP method is introduced in this chapter to solve the optimal control problem of modern wireless networks and automotive engines.

In the first part, a self-learning call admission control algorithm is developed for signal-to-interference ratio (SIR)-based power-controlled direct-sequence (DS)-code division multiple access (CDMA) cellular networks that provide both voice and data services [21]. The idea is built upon a novel learning control architecture with only a single module instead of two or three modules in adaptive critic designs. The call admission controller can perform learning in real time as well as in off-line environments and the controller improves its performance as it gains more experience. Another important contribution in the present work is the choice of utility function for the present self-learning control approach which makes the present learning process much more efficient than existing learning control methods.

In the second part, we apply an existing control algorithm to a production vehicle. The algorithm is presented according to certain criteria and calibrated for vehicle operation over the entire operating regime [24]. Actually, the algorithm has been optimized for the engine in terms of its performance, fuel economy, and tailpipe emissions through a significant effort in the research and development of calibration process. To further improve the engine performance through controller design, one may go through the traditional calibration and control procedures today. But an alternative to this traditional approach is to use the neural network-based learning control approach. The final result of our neural network learning process is a controller that has learned to provide optimal control signals under various operating conditions. We emphasize that such a neural network controller will be obtained

H. Zhang et al., *Adaptive Dynamic Programming for Control*,
Communications and Control Engineering, DOI 10.1007/978-1-4471-4757-2_10,

after a special learning process that performs adaptive dynamic programming algorithm. Once a controller is learned and obtained (off-line or on-line), it will be applied to perform the task of engine control. The performance of the controller can be further refined and improved through continuous learning in real-time vehicle operations.

10.2 Self-Learning Call Admission Control for CDMA Cellular Networks Using ADP

10.2.1 Problem Formulation

In the DS-CDMA cellular network model used in this section, we assume that separate frequency bands are used for the reverse link and the forward link, so that the mobiles only experience interference from the base stations and the base stations only experience interference from the mobiles. We consider cellular networks that support both voice and data services. Assume that there are K classes of services provided by the wireless networks under consideration, where $K \geq 1$ is an integer. We define a mapping $\sigma : Z^+ \to \{1, \ldots, K\}$ to indicate the fact that the nth connection is from the service class $\sigma(n)$, where Z^+ denotes the set of non-negative integers. We assume that each connection in our network may be from a different service class, which requires a different quality of service target (e.g., in terms of different bit error rate for each service class). This includes the case when we allow each call to specify its own quality of service requirements. We assume that traffic from the same service class has the same data rate, the same activity factor, the same desired SIR (signal-to-interference ratio) value, and the same maximum power limit that can be received at the base station.

Consider a base station currently with N active connections. The power received at the base station from the user (mobile station) of the nth connection is denoted by S_n, $n = 1, \ldots, N$. In an SIR-based power-controlled DS-CDMA network [2, 7, 13, 29], the desired value of S_n is a function of the number of active home connections and total other cell interference. If we assume that the maximum received power at a base station is limited to H_k for connections from service class $k = \sigma(n)$, then S_n is a random variable in the range of $(0, H_k]$. The maximum power limits H_k, $k = 1, \ldots, K$, are determined by the power limit of mobile transmitters, the cell size, the path loss information, and the user's service class. They have been used in several previous works on call admission control [9, 17, 29].

In CDMA networks, the instantaneous bit SIR (or the bit energy-to-interference ratio) for the nth connection at the base station (in a cell) can be expressed in terms of the received powers of the various connections as [9]

$$(E_b/N_0)_n = \frac{S_n W}{I_n R_{\sigma(n)}}, \tag{10.1}$$

where S_n is the instantaneous power level of the nth connection received at the base station, W is the total spread bandwidth (or the chip rate), and $R_{\sigma(n)}$ is the data rate of service class $\sigma(n)$. I_n in (10.1) indicates the instantaneous total interference to the nth connection received at the base station and it is given by

$$I_n = (1+f)\sum_{i=1, i\neq n}^{N} \nu_{\sigma(i)} S_i + \eta_n,$$

where $\nu_{\sigma(i)}$ is the traffic (e.g., voice) activity factor of the ith connection, which is from the service class $\sigma(i)$, η_n is the background (or thermal) noise, N is the number of active connections in the cell, and f is called the intercell interference factor [32], having a typical value of 0.55. In the above, the value of f may not always be constant in a system. Its value can be calculated using existing measurements and can be updated periodically to reflect changes in traffic conditions and traffic distributions.

Assume that after the admission of a new call or a handoff call, the power control algorithm starts to evolve until convergence. Assume that the power control algorithm converges and it requires the power received at a base station from each connection in the system given by $S_n^*, n = 0, 1, \ldots, N$, where the total number of connections in the system is $N+1$ and connection 0 is the newly admitted caller. Obviously, if $S_n^* > H_{\sigma(n)}$ or $S_n^* \leq 0$, for some n, $0 \leq n \leq N$, the admission should not be granted since it leads to an outage. Only when $0 < S_n^* \leq H_{\sigma(n)}$ for all n, $n = 0, 1, \ldots, N$, the admission decision can be considered as a correct decision. The goal of the present study is to develop a self-learning control algorithm that learns to achieve the correct admission decisions under various, possibly changing, environment conditions and user behavior and to optimize the grade of service (GoS) measure.

The GoS in cellular networks is mainly determined by the new call blocking probability and the handoff blocking probability. The former determines the fraction of new calls that are blocked, while the latter is closely related to the fraction of already admitted calls that cannot maintain their required quality of service (bit error rate) and are dropped. For example, many works have chosen to use the following definition for the GoS [1]:

$$\text{GoS} = P(\text{call blocking}) + w \times P(\text{handoff failure}), \tag{10.2}$$

where $P(a)$ is the probability of event a and w is typically chosen as, e.g., 10. In our simulation studies, we fix $w = 10$. The GoS defined in (10.2) provides a trade-off between the new call blocking rate and the handoff call blocking rate. The parameter w is a weighting factor that decides how much emphasis is placed on handoff calls. Keeping the GoS defined in (10.2) under a desired target level would require one to give much higher priority to handoff calls than to new calls when $w = 10$. On the other hand, quality of service (QoS) is usually defined according to the bit error rate in digital transmission. For example, the quality of service requirement for voice users is usually expressed as a bit error rate less than 10^{-3} in order to guarantee the

quality of communication which can be satisfied by the power control mechanism keeping E_b/N_0 at a required value of 7 dB or higher [9, 17, 29].

For a given set of parameters, including traffic statistics and mobility characteristics, fixed call admission control schemes can sometimes yield optimal solutions [26] in terms of GoS. All such schemes (see [12, 23, 26, 27, 29]), however, by reserving a fixed part of capacity, cannot adapt to changes in the network conditions due to their static nature. Therefore, we develop in the present work a self-learning call admission control algorithm for CDMA wireless networks. The present call admission control algorithm based on adaptive critic designs has the capability to learn from the environment and the user behavior so that the performance of the algorithm will be improved through further learning.

10.2.2 A Self-Learning Call Admission Control Scheme for CDMA Cellular Networks

10.2.2.1 Adaptive Critic Designs for Problems with Finite Action Space

In the following, we provide a brief introduction to adaptive critic designs [18]. Suppose that one is given a discrete-time non-linear dynamical system

$$x(k+1) = F[x(k), u(k), k],$$

where $x \in \mathbb{R}^n$ represents the state vector of the system and $u \in \mathbb{R}^m$ denotes the control action. In the present section, the function F denotes a stochastic transition from the state $x(k)$ to the next state $x(k+1)$ under the given control action $u(k)$ at time k. Suppose that one associates with this system the cost functional

$$J[x(i), i] = \sum_{k=i}^{\infty} \gamma^{k-i} L[x(k), u(k), k], \tag{10.3}$$

where L is called the utility function and γ is the discount factor with $0 < \gamma \leq 1$. Note that J is dependent on the initial time i and the initial state $x(i)$, and it is referred to as the cost-to-go of state $x(i)$. The objective is to choose the control sequence $u(k)$, $k = i, i+1, \ldots$, so that the cost functional J in (10.3) is minimized.

The class of ACDs considered in the present section is called action-dependent heuristic dynamic programming, or ADHDP, which is shown in Fig. 10.1 (see [19]). The critic network in this case will be trained by minimizing the following error measure over time:

$$\begin{aligned} \|E_q\| &= \sum_k E_q(k) \\ &= \sum_k \left[Q(k-1) - L(k) - \gamma Q(k) \right]^2, \end{aligned} \tag{10.4}$$

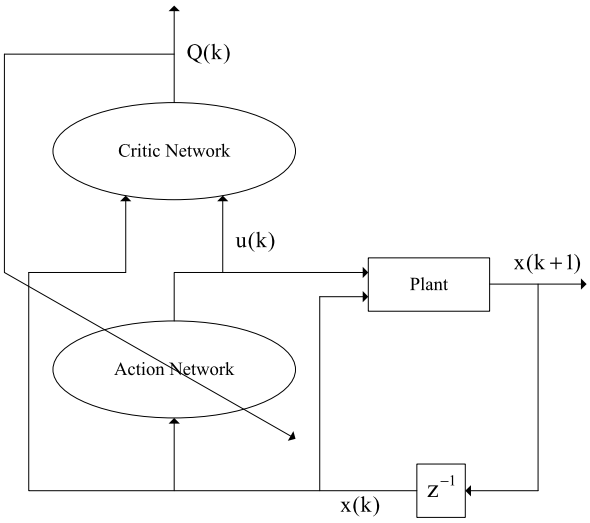

Fig. 10.1 A typical scheme of an action-dependent heuristic dynamic programming [19]

where $Q(k)$ is the critic network output at time k. When $E_q(k) = 0$ for all k, (10.4) implies that

$$\begin{aligned} Q(k-1) &= L(k) + \gamma Q(k) \\ &= L(k) + \gamma[L(k+1) + \gamma Q(k+1)] \\ &= \cdots \\ &= \sum_{i=k}^{\infty} \gamma^{i-k} L(i). \end{aligned} \tag{10.5}$$

Comparing (10.5) to (10.3), we see that when minimizing the error function in (10.4), we have a neural network trained so that its output becomes an estimate of the cost functional defined in dynamic programming for $i = k + 1$, i.e., the value of the cost functional in the immediate future [19].

The input–output relationship of the critic network in Fig. 10.1 is given by

$$Q(k) = Q\left[x(k), u(k), k, W_C^{(p)}\right],$$

where $W_C^{(p)}$ represents the weights of the critic network after the pth weight update. There are two methods to train the critic network according to the error function (10.4) in the present case, which are described in [19]. We will use the so-called forward-in-time method.

We can train the critic network at time $k - 1$, with the output target given by $L(k) + \gamma Q(k)$. The training of the critic network is to realize the mapping given by

$$C_f : \begin{Bmatrix} x(k-1) \\ u(k-1) \end{Bmatrix} \rightarrow \{L(k) + \gamma Q(k)\}. \tag{10.6}$$

In this case, the output from the network to be trained is $Q(k-1)$ and the input to the network to be trained is composed of $x(k-1)$ and $u(k-1)$. The target output value for the critic network training is calculated using its output at time k as indicated in (10.6). The goal of learning the function given by (10.6) is to have the critic network output satisfy

$$Q(k-1) \approx L(k) + \gamma Q(k) \quad \text{for all } k,$$

which is required by (10.5) for adaptive dynamic programming solutions.

The critic network training procedure is described as follows using the strategy of [16]:

1. Initialize two critic networks: cnet1 = cnet2;
2. Use cnet2 to get $Q(k)$, and then train cnet1 for 50 epochs using the Levenberg–Marquardt algorithm [11];
3. Copy cnet1 to cnet2, i.e., let cnet2 = cnet1;
4. Repeat Steps 2 and 3, e.g., four times;
5. Repeat Steps 1–4, e.g., ten times (start from different initial weights);
6. Pick the best cnet1 obtained as the trained critic network.

After the critic network's training is finished, the action network's training starts with the objective of minimizing the critic network output $Q(k)$. In this case, we can choose the target of the action network training as zero, i.e., we will update the action network's weights so that the output of the critic network becomes as small as possible. In general, a good critic network should not output negative values if $L(k)$ is non-negative. This is particularly true when $L(k)$ is chosen as the square error function in tracking control problems [31]. The desired mapping which will be used for the training of the action network in Fig. 10.1 is given by

$$A: \{x(k)\} \rightarrow \{0(k)\}, \tag{10.7}$$

where $0(k)$ indicates the target values of zero. We note that during the training of the action network, it will be connected to the critic network as shown in Fig. 10.1. The target in (10.7) is for the output of the whole network, i.e., the output of the critic network after it is connected to the action network as shown in Fig. 10.1.

There are many problems in practice that have a control action space that is finite. Typical examples include bang-bang control applications where the control signal only takes a few (finite) extreme values or vectors. When the application has only a finite action space, the decisions that can be made are constrained to a limited number of choices, e.g., a binary choice in the case of call admission control problem. When a new call or a handoff call arrives at a base station requesting admission, the decisions that a base station can make are constrained to two choices, i.e., to accept the call or to reject the call. Let us denote the two options by using $u(k) = +1$ for "accept" and $u(k) = -1$ for "reject". It is important to realize that in the present case the control actions are limited to a binary choice, or to only two possible options. Because of this, the ACDs introduced in Fig. 10.1 can be further simplified, so that only the critic network is needed. Our self-learning call admission control

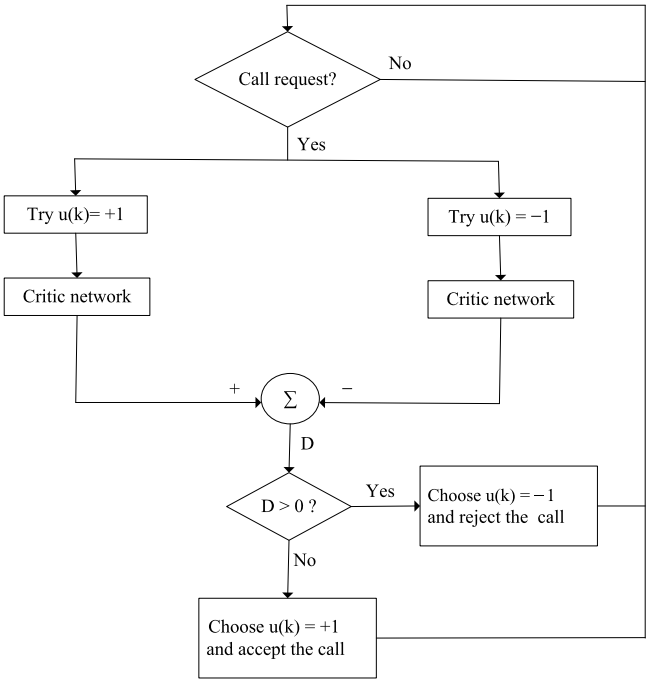

Fig. 10.2 The block diagram of the present adaptive critic approach

scheme for wireless cellular networks using adaptive critic designs is illustrated in Fig. 10.2. When a new call or a handoff call arrives at a base station requesting admission, we can first ask the critic network to see whether $u(k) = +1$ (accept) or $u(k) = -1$ (reject) will give a smaller output value. We will then choose the control action, choosing from $u(k) = +1$ and $u(k) = -1$, that gives a smaller critic network output. As in the case of Fig. 10.1, the critic network would also take the states of the system as input. We note that Fig. 10.2 is only a schematic diagram that shows how the computation takes place while making a call admission control decision. The two blocks for the critic network in Fig. 10.2 represent the same network or computer code in software. The block diagram in Fig. 10.2 indicates that the critic network will be used twice in calculations (with different values of $u(k)$) to make a decision on whether or not to accept a call.

The above description assumes that the critic network has been trained successfully. Once the critic network training is done, it can be applied as in Fig. 10.2. To guarantee that the overall system will achieve optimal performance now and in future environments which may be significantly different from what they are now, we will allow the critic network to perform further learning when needed in the future. In the next section, we describe how the critic network learning is performed. In particular, we describe how the training data are collected at each time step and how the utility function $L(k)$ is defined. We note that once the training data are collected, the training of the critic network can use, e.g., the forward-in-time method, described

in this section. We also note that the description here for the critic network training applies to both the initial training of the critic network and further training of the critic network when needed in the future.

10.2.2.2 Self-learning Call Admission Control for CDMA Cellular Networks

We use a utility function as reward or penalty to the action made by the call admission control scheme. When the call admission control scheme makes a decision about accepting a call, it will lead to two distinct results. The first is that the decision of accepting a call is indeed the right decision due to the guarantee of quality of service during the entire call duration. In this case we should give a reward to the decision of accepting a call. Otherwise, a penalty is assigned to this decision. On the other hand, if rejecting a call would have been the right decision due to call dropping or system outage after the call acceptance, we will also give a reward to the decision of rejecting a call. Otherwise, a penalty is assigned to this decision.

It is generally accepted in practice that handoff calls will be given higher priority than new calls [12, 26, 27, 29]. This is accomplished in our call admission control scheme by using different thresholds for new calls and handoff calls. A handoff call can be admitted if $0 < S_n^* \leq H_{\sigma(n)}$ for all $n = 0, 1, \ldots, N$. A new call can only be admitted if $0 < S_0^* \leq T_{\sigma(0)}$ and $0 < S_n^* \leq H_{\sigma(n)}$ for $n = 1, 2, \ldots, N$, where $T_{\sigma(0)} < H_{\sigma(0)}$ is the threshold for new calls (where connection 0 is the new caller).

For handoff calls, when $0 < S_n^* \leq H_{\sigma(n)}$ for $n = 0, 1, \ldots, N$, accepting a handoff call will be given a reward and rejecting a handoff call will be given a penalty. On the other hand, when $S_n^* > H_{\sigma(n)}$ for some n, $0 \leq n \leq N$, accepting a handoff call (i.e., the zeroth caller) will be given a penalty and rejecting a handoff call will be given a reward. Obviously, when $S_n^* \leq 0$ for some n, $0 \leq n \leq N$, the call should be rejected. We note that if the power control algorithm leads to either $S_n^* > H_{\sigma(n)}$ or $S_n^* \leq 0$ for some n, the network will enter an outage, i.e., some calls will have to be terminated prematurely since they cannot maintain required quality of service (bit error rate). In this case, the action of rejection is given a reward and the action of acceptance is given a penalty. Note that S_n^*, $n = 0, 1, \ldots, N$, are power levels for all connections after the call admission decision and the power control algorithm convergence.

We first define the cost functional for handoff calls as follows ($S_n^* \geq 0$ for $n = 0, 1, \ldots, N$):

$$E_n = \xi \max \left\{ u(k) \left[\frac{S_n^*}{H_{\sigma(n)}} - 1 \right], 0 \right\}, \tag{10.8}$$

where $\xi > 0$ is a coefficient, and $u(k) = 1$ represents accepting a call and $u(k) = -1$ represents rejecting a call. We emphasize that the conditions, $0 \leq S_n^* \leq H_{\sigma(n)}$ for $n = 0, 1, \ldots, N$, must hold for the entire duration of all calls in order for the system to give reward to the action of accepting a handoff call.

For new calls, when $0 < S_0^* \le T_{\sigma(0)}$ and $0 < S_n^* \le H_{\sigma(n)}$ for $n = 1, 2, \ldots, N$, we give a reward to the action of accepting a new call, and we give a penalty to the action of rejecting a new call. When $S_0^* > T_{\sigma(0)}$, or $S_n^* > H_{\sigma(n)}$ for some n, $n = 1, 2, \ldots, N$, or $S_n^* \le 0$ for some n, $n = 0, 1, \ldots, N$, we give penalty for accepting a new call and we give a reward for rejecting a new call. The cost functional for new calls is defined as ($S_n^* \ge 0$ for $n = 0, 1, \ldots, N$):

$$E_n = \begin{cases} \xi \max\left\{\left[\dfrac{S_n^*}{T_{\sigma(n)}} - 1\right], 0\right\}, & \text{when } u(k) = 1, \\ \xi \max\left\{\left[1 - \dfrac{S_n^*}{H_{\sigma(n)}}\right], 0\right\}, & \text{when } u(k) = -1, \end{cases} \tag{10.9}$$

where $T_{\sigma(n)} < H_{\sigma(n)}, n = 0, 1, \ldots, N$. We note again that the conditions, $0 < S_n^* \le H_{\sigma(n)}$ for $n = 0, 1, \ldots, N$, must hold for the entire duration of all calls in order for the system to give reward to the action of accepting a new call, even though the admission decision is according to the condition $0 < S_0^* \le T_{\sigma(0)}$ for the new call.

The functions defined above satisfy the condition that $E_n \ge 0$ for all n, $n = 0, 1, \ldots, N$. From (10.8) and (10.9), we see that when the action is "accept", if the value of the utility function of any connection is larger than 0, this action should be penalized. Also, when the action is "reject", if the value of the utility function of any connection is zero, this action should be rewarded. Therefore, from the system's point of view, the cost function should be chosen as

$$E = \begin{cases} \max\limits_{0 \le n \le N} (E_n), \text{ if } u(k) = 1, \\ \min\limits_{0 \le n \le N} (E_n), \text{ if } u(k) = -1. \end{cases} \tag{10.10}$$

The cost function defined in (10.10) indicates that the goal of our call admission control algorithm is to minimize the value of function E, i.e., to reach its minimum value of zero and to avoid its positive values.

The utility function L (used in (10.3)) in our present work is chosen as

$$L(u) = \frac{E}{1 + E}. \tag{10.11}$$

Figures 10.3 and 10.4 show plots of this utility function for handoff calls and new calls, respectively, when $\xi = 10$. The parameter ξ is used to obtain a desired shape of the utility function. Since our algorithm will search for the optimal performance that corresponds to small values of the utility function, the shape of the utility function will have some effects on the optimization process. When $\xi = 10$, we see that the utility function becomes more selective than $\xi = 1$ for any condition indicated by signal power. From Figs. 10.3 and 10.4 we see that the choice of the present utility functions in (10.11) clearly shows minimum points (the flat areas) that our

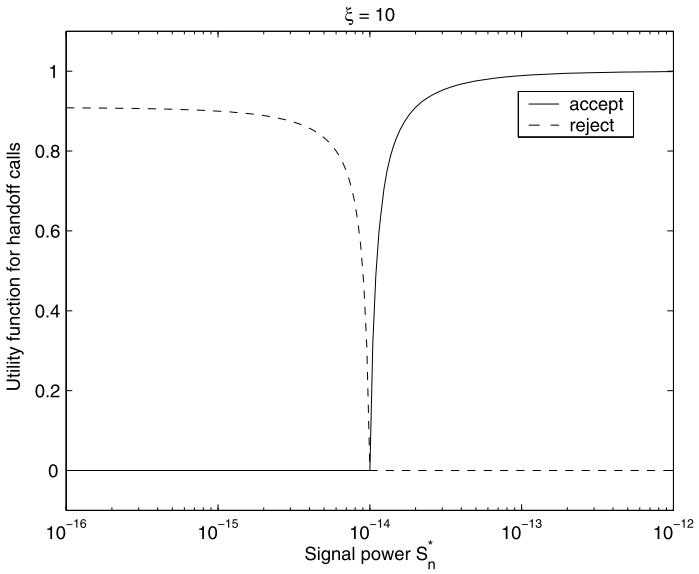

Fig. 10.3 Utility function for handoff calls

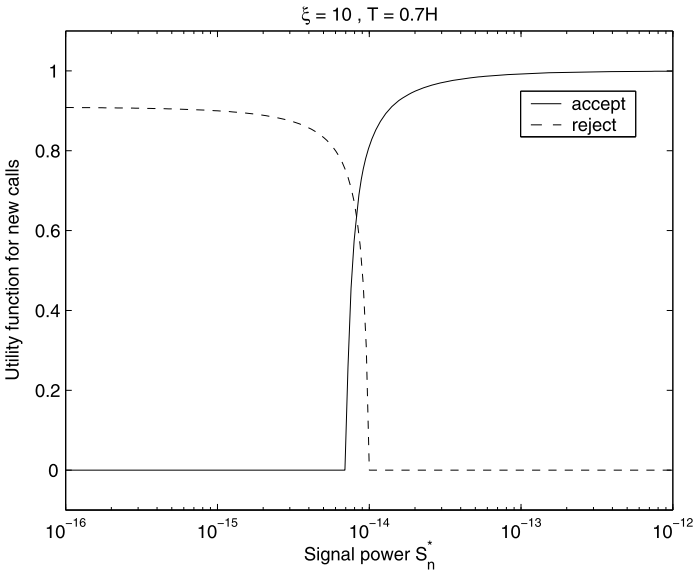

Fig. 10.4 Utility function for new calls

call admission control scheme tries to reach and the points with high penalty that our scheme should avoid. In addition, the conversion from E to U guarantees the convergence of the performance index of dynamic programming, which is defined

as in (10.3). With the present utility function given by (10.10) and (10.11), we have

$$0 < J(k) < \frac{1}{1-\gamma},$$

since L in (10.11) satisfies $0 \leq L < 1$. Without the conversions in (10.11), there is no guarantee that the infinite summation in (10.3) will be bounded. We note that the present critic network produces an output that approximates the cost functional $J(k)$ in (10.3), and the admission action chosen each time will minimize the critic network output, to achieve approximate optimal control.

In stationary environment, where user traffic statistics (patterns) remain unchanged, a simple static call admission control algorithm [20] will be able to achieve the admission objective described above. However, traffic patterns including user arrival rate, call holding times, user mobility patterns, etc., may show significant changes from time to time. To deal with changing environments, static call admission control algorithm would not be appropriate. The present call admission control algorithm based on adaptive critic designs will be able to deal with environment changes through further learning in the future. Another benefit of the present self-learning call admission control algorithm is its ability to improve performance through further learning as the controller gains more and more experience.

The development of the present self-learning call admission control scheme involves the following four steps:

1. Collecting data: During this phase, when a call comes, we can accept or reject the call with any scheme and calculate the utility function for the system as presented above. In the present section, we simply accept and reject calls randomly with the same probability of 0.5. At the same time, we collect the states corresponding to each action. The states (environment) collected for each action include total interference, call type (new call or handoff call), call class (voice or data), etc.
2. Training critic network: Using the data collected to train the critic network as mentioned in the previous section. Examples of input variables chosen for the critic network will be given in our simulation examples.
3. Applying critic network: The trained critic network is then applied as shown in Fig. 10.2.
4. Further updating critic network: The critic network will be updated as needed while it is used in application to accommodate environment changes, for example, user pattern and behavior changes or new requirements for the system. Data collection has to be performed again and the training of the critic network as well. In this case, the above three steps will be repeated.

The critic network will be updated when there are changes in call admission requirements or if the already trained ACD scheme does not satisfy new requirements. In fact, ACD is going to learn the rules imposed by the utility function of the system. Therefore, rule changes can be accommodated by modifying the utility function to accommodate new requirements. For example, to satisfy certain requirements, we

modify the cost function in (10.9) to become

$$E_n = \begin{cases} \xi \max\left\{\left[\dfrac{S_n^*}{H_{\sigma(n)}} - 1\right], 0\right\}, & \text{when } u(k) = 1 \\ & \text{and } n_a \le N_h, \\ \xi \max\left\{\left[\dfrac{S_n^*}{T_{\sigma(n)}} - 1\right], 0\right\}, & \text{when } u(k) = 1 \\ & \text{and } n_a > N_h, \\ \xi \max\left\{\left[1 - \dfrac{S_n^*}{H_{\sigma(n)}}\right], 0\right\}, & \text{when } u(k) = -1, \end{cases} \tag{10.12}$$

where n_a is the number of active handoff calls in the cell and N_h is a fixed parameter indicating the threshold for low traffic load.

When a call arrives at the base station, the admission decision would be either "accept" or "reject". If the decision is taken to accept a call, there will be two kinds of data collected. One is that the decision is incorrect due to call dropping, and the other one is that the decision is correct since $0 < S_n^* \le H_{\sigma(n)}$, $n = 0, 1, \ldots, N$, is maintained for the entire duration of all calls. In the former case, a penalty is recorded and in the latter case, a reward is recorded. If the decision is to reject a call, there will be also be two kinds of data collected that correspond to a reward and a penalty. Note that in the case of a "reject" decision, the value of the utility function is determined as in (10.8)–(10.11) where the values of S_n^*, $n = 0, 1, \ldots, N$, are calculated according to [3, 20, 30].

10.2.3 Simulations

We first conduct simulation studies for a network with a single class of service (e.g., voice). The network parameters used in the present simulation are taken similarly as the parameters used in [13, 29] (see Table 10.1).

The arrival rate consists of the new call attempt rate λ_c and the handoff call attempt rate λ_h. The new call attempt rate λ_c depends on the expected number of subscribers per cell. The handoff call attempt rate λ_h depends on such network parameters as traffic load, user velocity, and cell coverage areas [10, 12]. In our simulation, we assume that $\lambda_c : \lambda_h = 5 : 1$ [12]. A channel is released by call completion or handoff to a neighboring cell. The channel occupancy time is assumed to be exponentially distributed [10, 12] with the same mean value of $1/\mu = 3$ minutes. For each neural network training in our simulation studies, we generated 35000 data points according to the data collection procedure in the previous section.

In the following, we conduct comparison studies between the present self-learning call admission control algorithm and the algorithm developed in [20] with fixed thresholds for new calls given by $T = H$, $T = 0.8H$, and $T = 0.5H$, respectively. The arrival rate in all neighboring cells is fixed at 18 calls/minutes.

Table 10.1 Network parameters

Parameters	Values	Parameters	Values
W	1.2288 Mcps	R	9.6 kbps
η	1×10^{-14} W	H	1×10^{-14} W
E_b/N_0	7 dB	v	3/8

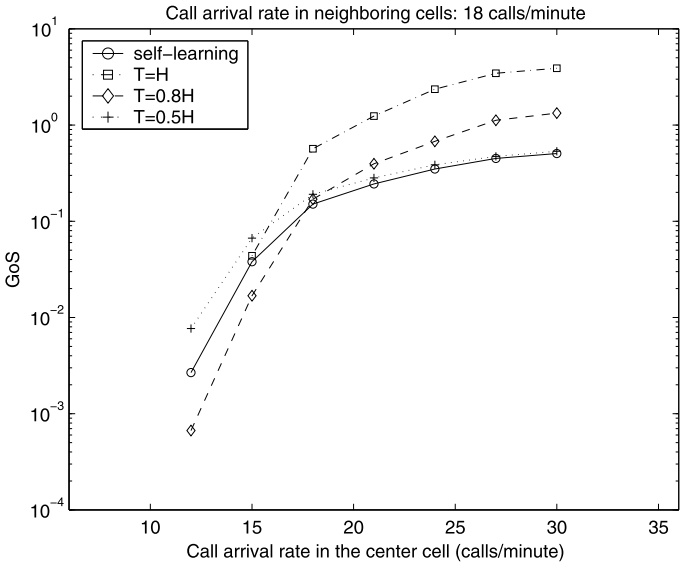

Fig. 10.5 Comparison study using utility function defined in (10.9)

The training data are collected as mentioned in the previous section. We choose $T_{\sigma(n)} = 0.5H_{\sigma(n)}$ and $\xi = 10$ in (10.9). The critic network has three inputs. The first is the total interference received at the base station, the second is the action (1 for accepting, -1 for rejecting), and the third is the call type (1 for new calls, -1 for handoff calls). The critic network is a multilayer feedforward neural network with 3–6–1 structure, i.e., three neurons at the input layer, six neurons at the hidden layer, and one neuron at the output layer. Both the hidden and the output layers use the hyperbolic tangent function as the activation function. Figure 10.5 shows the simulation results. We see from the figure that the performance of the self-learning algorithm is similar to the case of static algorithm with $T = 0.5H$, because we choose $T_{\sigma(n)} = 0.5H_{\sigma(n)}$ in (10.9) for our learning control algorithm. When the call arrival rate is low, the self-learning algorithm is not so good because it reserves too much for handoff calls and as a result it rejects too many new calls. That is why the GoS is worse than the other two cases of static algorithms ($T = 1.0H$ and $T = 0.8H$). In this case, the self-learning algorithm is trained to learn a call admission control scheme that gives higher priority to handoff calls.

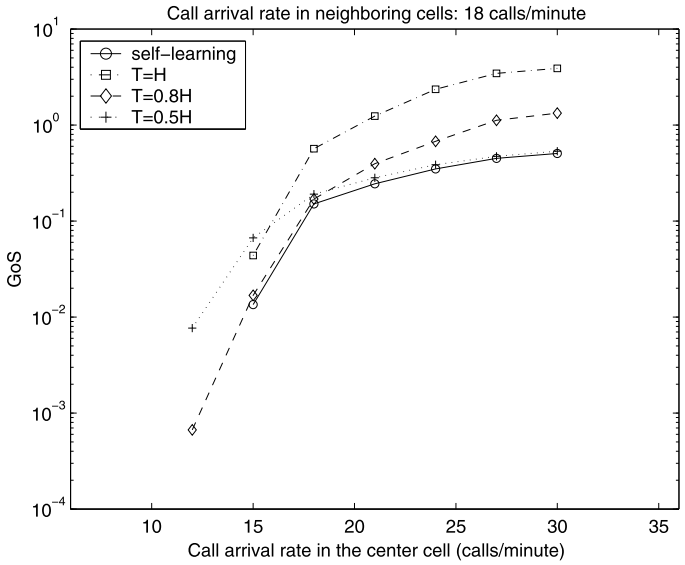

Fig. 10.6 Comparison study using utility function defined in (10.12)

In order to improve the GoS when the call arrival rate is low, we use the modified cost function for new calls as in (10.12), where we choose $N_h = 15$ in our simulation. Using this new utility function we collect the training data using one of the static algorithms with fixed threshold or the previous critic network. Then we train a new critic network with the newly collected data. In this time the critic network has four inputs. Three of them are the same as in the previous critic network. The new input is equal to 1 when $n_a \leq N_h$ and otherwise it is equal to -1. The critic network in this case has a structure given by 4–8–1. Figure 10.6 shows the result of applying the new critic network to the same traffic pattern as in Fig. 10.5. From the figure we see that the self-learning algorithm using the new critic network has the best GoS. We see that by simply changing the cost function from (10.11) to (10.12), the self-learning algorithm can significantly improve its performance to outperform static admission control algorithms. One of the benefits of self-learning call admission control algorithm is that we can easily and efficiently design call admission control algorithms by modifying the cost function (equivalently, the utility function) to satisfy the requirements or to accommodate new environment changes.

The traffic load in telephony systems is typically time varying. Figure 10.7 shows a pattern concerning call arrivals during a typical 24 hour business day, beginning at midnight [8]. It can be seen that the peak hours occur around 11:00 am and 4:00 pm. Next, we use our newly trained critic network above for this traffic pattern. Figure 10.8 gives the simulation results under the assumption that the traffic load was spatially uniformly distributed among cells, but followed the time-varying pattern given in Fig. 10.7. Figure 10.8 compares the four call admission control algorithms and shows that the self-learning algorithm has the best GoS among all the algorithms tested. We note that the self-learning call admission control algorithm was

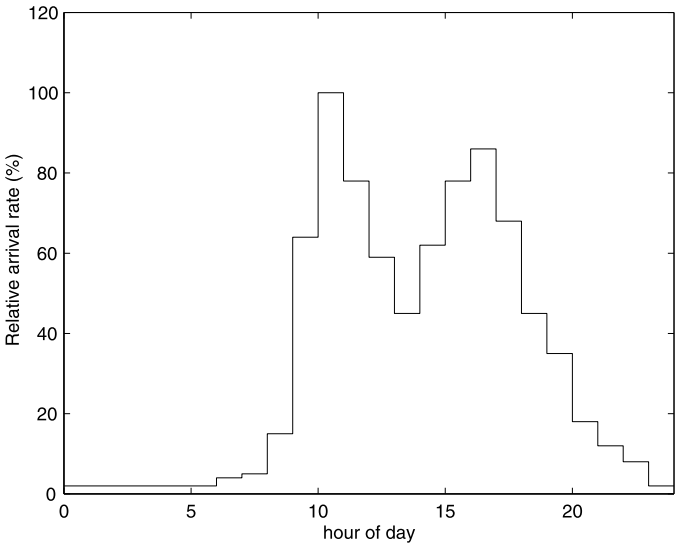

Fig. 10.7 A traffic pattern of a typical business day

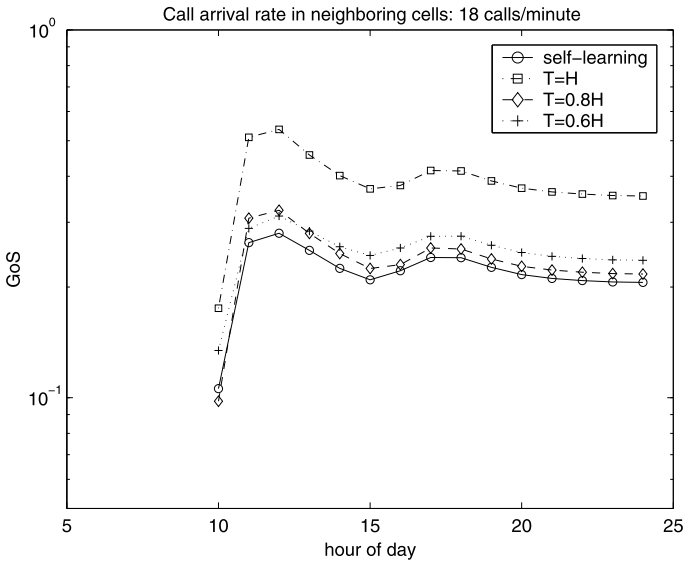

Fig. 10.8 Comparison study for varying traffic

not retrained from the previous case, i.e., we used the same critic network in the simulation results of Fig. 10.8 as in Fig. 10.6.

In the following, we conduct comparison studies between the present self-learning call admission control algorithm and that of [29]. Using the algorithm

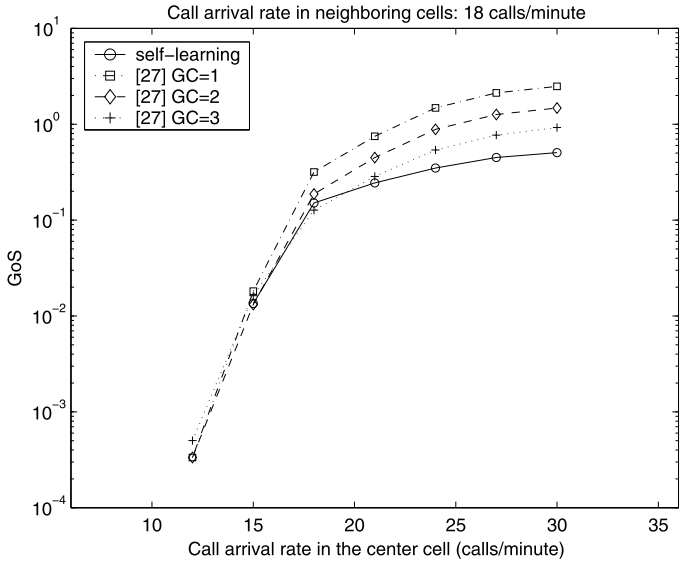

Fig. 10.9 Comparison studies with the algorithm in [29]

in [29], the base station controller reads the current interference from the power strength measurer. It then estimates the current interference margin (CIM) and handoff interference margin (HIM), where CIM $<$ HIM. A total interference margin (TIM) is set according to the quality of service target. If CIM $>$ TIM, reject the call admission request. If HIM $<$ TIM, accept the call request. If CIM $<$ TIM $<$ HIM, then only handoff calls will be accepted. Figure 10.9 compares the present self-learning call admission control algorithm with the algorithm in [29] that reserves one, two, and three channels for handoff calls, respectively. The arrival rate in all neighboring cells is fixed at 18 calls/minute. We assume the use of a hexagonal cell structure. From Fig. 10.9, we see that the present algorithm has the best GoS. That is because the algorithm in [29] is a kind of guard channel algorithm used in CDMA systems. Therefore, when the load is low, GC $= 1$ performs the best, and when the load is high, GC $= 3$ performs the best. However, our algorithm can adapt to varying traffic load conditions. It has the best overall performance under various traffic loads. Again, we used the same critic network in the simulation results of Fig. 10.9 as in Fig. 10.6.

Finally, we conduct simulation studies for cellular networks with two classes of services. One class is voice service and the other is data service. Network parameters in our simulations are chosen in reference to the parameters used in [14, 28] (see Table 10.2). In our simulation, the data traffic is similar to that in [14], i.e., low resolution video or interactive data. In this case, the data traffic can be specified by a constant transmission rate. The background noise in this case is chosen the same as in Table 10.1. The utility function is defined for voice and data calls as in (10.9) and (10.11). In (10.12), we choose $T_{\sigma(n)} = 0.6H_{\sigma(n)}$ and $\xi = 10$ for both voice calls and

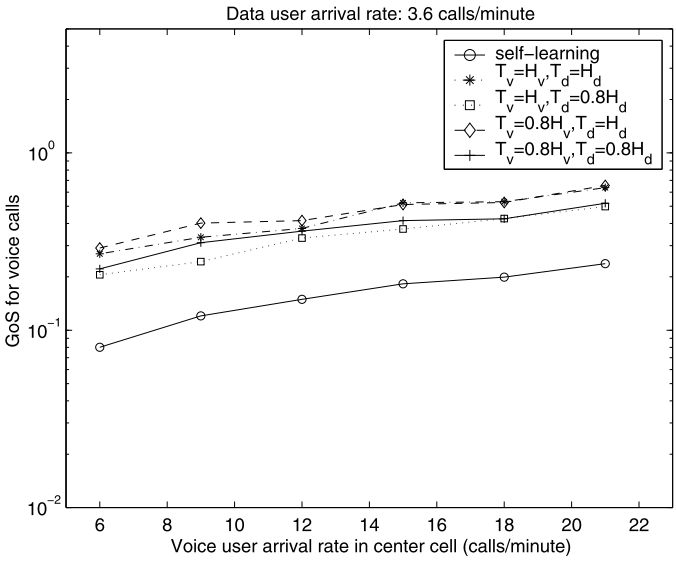

Fig. 10.10 GoS for voice calls

Table 10.2 Network parameters

Voice users		Data users	
Parameters	Values	Parameters	Values
W_v	4.9152 Mcps	W_d	4.9152 Mcps
R_v	9.6 kbps	R_d	38.4 kbps
H_v	1×10^{-14} W	H_d	1×10^{-13} W
$(E_b/N_0)_v$	7 dB	$(E_b/N_0)_d$	9 dB
ν_v	3/8	ν_d	1

data calls. N_h is chosen as 20 and 4 for voice calls and data calls, respectively. The critic network now has five inputs. The newly added input is the call class which is 1 for voice calls and -1 for data calls. The critic network structure is chosen as 5–10–1. Figures 10.10 and 10.11 compare our self-learning call admission control algorithm and the static algorithm [20] with fixed thresholds given by $T = H$ and $T = 0.8H$. The arrival rates of voice users and data users in all neighboring cells are fixed at 20 calls/minute and 3 calls/minute, respectively. From Figs. 10.10 and 10.11, we see that the present self-learning algorithm has the best GoS for almost all call arrival rates tested. We conclude that the present self-learning algorithm performs better than the fixed algorithms due to the fact that the self-learning algorithm can adapt to varying traffic conditions and environment changes.

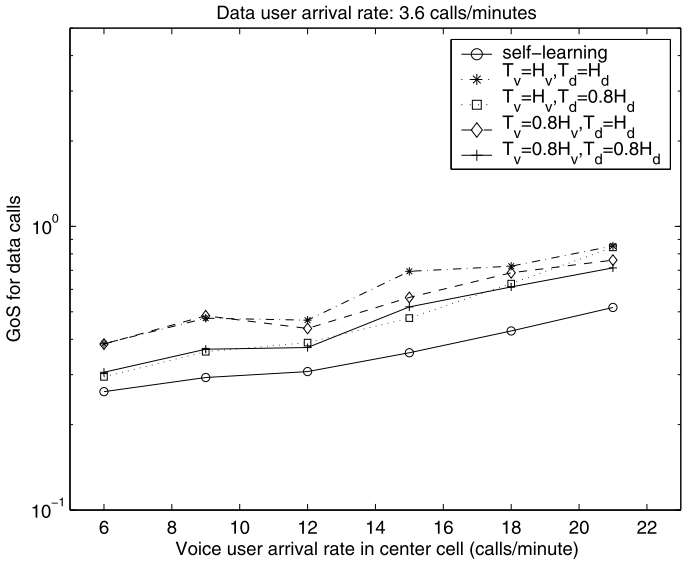

Fig. 10.11 GoS for data calls

10.3 Engine Torque and Air–Fuel Ratio Control Based on ADP

10.3.1 Problem Formulation

A test vehicle with a V8 engine and 4-speed automatic transmission is instrumented with engine and transmission torque sensors, wide-range air–fuel ratio sensors in the exhaust pipe located before and after the catalyst on each bank, as well as exhaust gas pressure and temperature sensors. The vehicle is also equipped with a dSPACE rapid prototyping controller for data collection and controller implementation. Data are collected at each engine event under various driving conditions, such as the Federal Test Procedure (FTP cycles), as well as more aggressive driving patterns, for a length of about 95,000 samples during each test. The engine is run under closed-loop fuel control using switching-type oxygen sensors. The dSPACE is interfaced with the power-train control module (PCM) in a by-pass mode.

We build a neural network model for the test engine with a structure compatible with the mathematical engine model developed by Dobner [5, 6, 22] and others. Due to the complexity of modern automotive engines, in the present work, we use the time-lagged recurrent neural networks (TLRNs) for engine modeling. In practice, TLRNs have been used often for function approximation and it is believed that they are more powerful than the networks with only feedforward structures (cf. [25, 33]).

For the neural network engine model, we choose air–fuel ratio (AFR) and engine torque (TRQ) as the two outputs. We choose throttle position (TPS), electrical fuel pulse width (FPW), and spark advance (SPA) as the three control inputs. These are

input signals to be generated using our new adaptive critic learning control algorithm. We choose intake manifold pressure (MAP), mass air flow rate (MAF), and engine speed (RPM) as reference input. The time-lagged recurrent neural network used for the engine combustion module has six input neurons, a single hidden layer with eight neurons, and two output neurons.

Validation results for the output TRQ and AFR of the neural network engine model indicate a very good match between the real vehicle data and the neural network model output during the validation phase [15].

10.3.2 Self-learning Neural Network Control for Both Engine Torque and Exhaust Air–Fuel Ratio

Suppose that one is given a discrete-time non-linear dynamical system

$$x(k+1) = F[x(k), u(k), k], \tag{10.13}$$

where $x \in \mathbb{R}^n$ represents the state vector of the system and $u \in \mathbb{R}^m$ denotes the control action. Suppose that one associates with this system the cost functional (or cost)

$$J[x(i), i] = \sum_{k=i}^{\infty} \gamma^{k-i} L[x(k), u(k), k], \tag{10.14}$$

where L is called the utility function or local cost function and γ is the discount factor with $0 < \gamma \leq 1$. Note that J is dependent on the initial time i and the initial state $x(i)$, and it is referred to as the cost-to-go of state $x(i)$. The objective is to choose the control sequence $u(k)$, $k = i, i+1, \ldots$, so that the cost functional J (i.e., the cost) in (10.14) is minimized.

Adaptive critic designs (ACDs) are defined as designs that approximate dynamic programming in the general case, i.e., approximate optimal control over time in noisy, non-linear environments. A typical design of ACDs consists of three modules—Critic (for evaluation), Model (for prediction), and Action (for decision). When in ACDs the critic network (i.e., the evaluation module) takes the action/control signal as part of its input, the designs are referred to as action-dependent ACDs (ADACDs). We use an action-dependent version of ACDs that does not require the explicit use of the model network in the design. The critic network in this case will be trained by minimizing the following error measure over time:

$$\begin{aligned} \|E_q\| &= \sum_k E_q(k) \\ &= \sum_k \left[Q(k-1) - L(k) - \gamma Q(k) \right]^2, \end{aligned} \tag{10.15}$$

where $Q(k) = Q[x(k), u(k), k, W_C]$. When $E_q(k) = 0$ for all k, (10.15) implies that

$$\begin{aligned} Q(k-1) &= L(k) + \gamma Q(k) \\ &= L(k) + \gamma[L(k+1) + \gamma Q(k+1)] \\ &= \cdots \\ &= \sum_{i=k}^{\infty} \gamma^{i-k} L(k). \end{aligned} \tag{10.16}$$

We see that when minimizing the error function in (10.15), we have a neural network trained so that its output at time k becomes an estimate of the cost functional defined in dynamic programming for $i = k + 1$, i.e., the value of the cost functional in the immediate future [19].

The input–output relationship of the critic network is given by

$$Q(k) = Q\left[x(k), u(k), k, W_C\right],$$

where W_C represents the weight vector of the critic network.

We can train the critic network at time $k - 1$, with the desired output target given by $L(k) + \gamma Q(k)$. The training of the critic network is to realize the mapping given by

$$C_f : \left\{ \begin{matrix} x(k-1) \\ u(k-1) \end{matrix} \right\} \rightarrow \{L(k) + \gamma Q(k)\}. \tag{10.17}$$

We consider $Q(k-1)$ in (10.15) as the output from the network to be trained and the target output value for the critic network is calculated using its output at time k.

After the critic network's training is finished, the action network's training starts with the objective of minimizing $Q(k)$. The goal of the action network training is to minimize the critic network output $Q(k)$. In this case, we can choose the target of the action network training as zero, i.e., we will train the action network so that the output of the critic network becomes as small as possible. The desired mapping which will be used for the training of the action network in the present ADHDP is given by

$$A : \{x(k)\} \rightarrow \{0(k)\}, \tag{10.18}$$

where $0(k)$ indicates the target values of zero. We note that during the training of the action network, it will be connected to the critic network to form a larger neural network. The target in (10.18) is for the output of the whole network, i.e., the output of the critic network after it is connected to the action network.

After the action network's training cycle is completed, one may check the system's performance, then stop or continue the training procedure by going back to the critic network's training cycle again, if the performance is not acceptable yet.

Assume that the control objective is to have $x(k)$ in (10.13) track another signal given by $x^*(k)$. We define in this case the local cost function $L(k)$ as

$$L(k) = \frac{1}{2}e^{\mathrm{T}}(k)e(k) = \frac{1}{2}[x(k) - x^*(k)]^{\mathrm{T}}[x(k) - x^*(k)].$$

Using the ADHDP introduced earlier in this section, we can design a controller to minimize

$$J(k) = \sum_{i=k}^{\infty} \gamma^{i-k} L(i),$$

where $0 < \gamma < 1$. We note that in this case our control objective is to minimize an infinite summation of $L(k)$ from the current time to the infinite future, while in conventional tracking control designs, the objective is often to minimize $L(k)$ itself.

10.3.3 Simulations

The objective of the present engine controller design is to provide control signals so that the torque generated by the engine will track the torque measurement as in the data and the air–fuel ratio will track the required values also as in the data. The measured torque values in the data are generated by the engine using the existing controller. Our learning controller will assume no knowledge about the control signals provided by the existing controller. It will generate a set of control signals that are independent of the control signals in the measured data. Based on the data collected, we use our learning controller to generate control signals TPS, FPW, and SPA, with the goal of producing exactly the same torque and air–fuel ratio as in the data set. That is to say, we keep our system under the same requirements as the data collected, and we build a controller that provides control signals which achieve the torque control and air–fuel ratio control performance of the engine.

As described in the previous section, the development of an adaptive critic learning controller involves two stages: the training of a critic network and the development of a controller/action network. We describe in the rest of the present section the learning control design for tracking the TRQ and AFR measurements in the data set. This is effectively a torque-based controller, i.e., a controller that can generate control signals given the torque demand. The block diagram of the present adaptive critic engine control (including air–fuel ratio control) is shown in Fig. 10.12. The diagram shows how adaptive critic designs can be applied to engine control through adaptive dynamic programming.

10.3.3.1 Critic Network

The critic network is chosen as a 8–15–1 structure with eight input neurons and 15 hidden layer neurons:

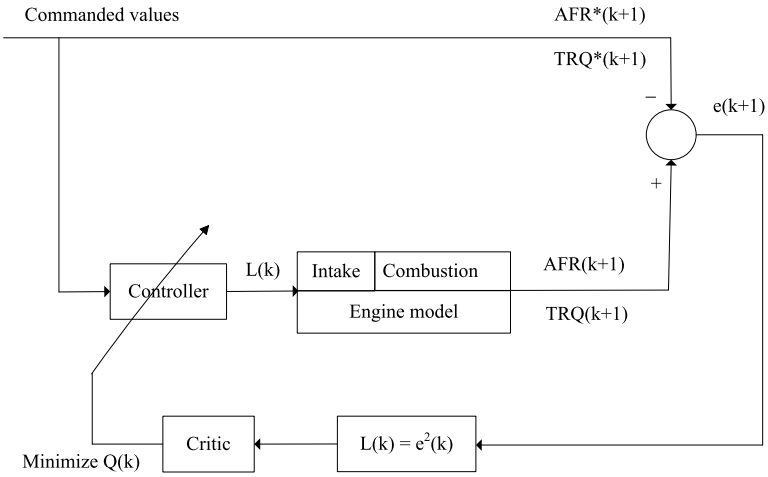

Fig. 10.12 Structure of adaptive critic learning engine controller

- The eight inputs to the critic network are TRQ, TRQ*, MAP, MAF, RPM, TPS, FPW, and SPA, where TRQ* is read from the data set, indicating the desired torque values for the present learning control algorithm to track.
- The hidden layer of the critic network uses a sigmoidal function, i.e., the `tansig` function in MATLAB [4], and the output layer uses the linear function purelin.
- The critic network outputs the function Q, which is an approximation to the function $J(k)$ defined as in (10.14).
- The local cost functional L defined in (10.14) in this case is chosen as

$$L(k) = \frac{1}{2}[\text{TRQ}(k) - \text{TRQ}^*(k)]^2 + \frac{1}{2}[\text{AFR}(k) - \text{AFR}^*(k)]^2,$$

  where TRQ and AFR are the engine torque and air–fuel ratio generated using the proposed controller, respectively, and TRQ* and AFR* are the demanded TRQ value and the desired AFR value, respectively. Both TRQ* and AFR* are taken from the actual measured data in the present case. The utility function chosen in this way will lead to a control objective of TRQ following TRQ* and AFR following AFR*.
- Utilizing the MATLAB Neural Network Toolbox, we apply `traingdx` (gradient descent algorithm) for the training of the critic network. We note that other algorithms implemented in MATLAB, such as `traingd`, `traingda`, `traingdm`, `trainlm` are also applicable. We employ batch training for the critic network, i.e., the training is performed after each trial of a certain number of steps (e.g., 10000 steps). We choose $\gamma = 0.9$ in the present experiments.

10.3.3.2 Controller/Action Network

The structure of the action network is chosen as 6–12–3 with six input neurons, 12 hidden layer neurons, and three output neurons:

- The six inputs to the action network are TRQ, TRQ*, MAP, MAF, THR, and RPM, where THR indicates the driver's throttle command.
- Both the hidden layer and the output layer use the sigmoidal function `tansig`.
- The outputs of the action network are TPS, FPW, and SPA, which are the three control input signals used in the engine model.
- The training algorithm we choose to use is `traingdx`. We employ batch training for the action network as well.

10.3.3.3 Simulation Results

In the present simulation studies, we first train the critic network for many cycles with 500 training epochs in each cycle. At the end of each training cycle, we check the performance of the critic network. Once the performance is found to be satisfactory, we stop critic network training. This process usually takes about 6–7 hours.

After the critic network training is finished, we start the action network training. We train the controller network for 200 epochs after each trial. We check to see the performance of the neural network controller at the end of each trial.

We choose to use 4000 data points from the data (16000–20000 in the data set) for the present critic and action network training.

We first show the TRQ and AFR output due to the initial training of our neural network controller when TRQ* and AFR* are chosen as random signals during training. Figures 10.13 and 10.14 show the controller performance when it is applied with TRQ* and AFR* chosen as the measured values in the data set. The neural network controller in this case is trained for 15 cycles using randomly generated target signal TRQ* and AFR*. Figures 10.13 and 10.14 show that very good tracking control of the commanded torque signal (TRQ) and the exhaust AFR are achieved. We note that at the present stage of the research we have not attempted to regulate the AFR at the stoichiometric value but to track a given command. In these experiments we simply try to track the measured engine-out AFR values so that the control signal obtained can directly be validated against the measured control signals in the vehicle. In Fig. 10.16, it appears that better tracking of AFR was achieved on the rich side of stoichiometric value, possibly due to more frequent rich excursions encountered during model training. This could also have been caused by intentional fuel enrichments (i.e., wall-wetting compensation) during vehicle accelerations.

Figures 10.15 and 10.16 show the TRQ and AFR output after refined training when TRQ* and AFR* are chosen as the measured values in the data. The neural network controller in this case is trained for 15 cycles using target signal TRQ* and AFR* as in the data. Figures 10.15 and 10.16 show that excellent tracking control results for the commanded TRQ and AFR are achieved.

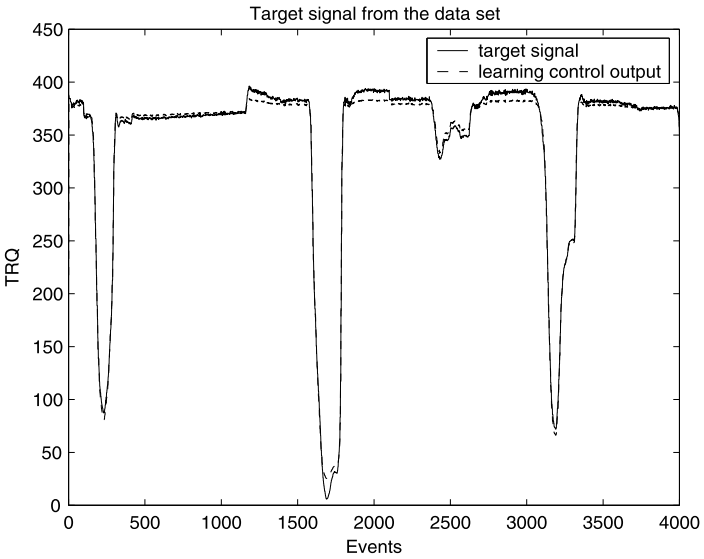

Fig. 10.13 Torque output generated by the neural network controller

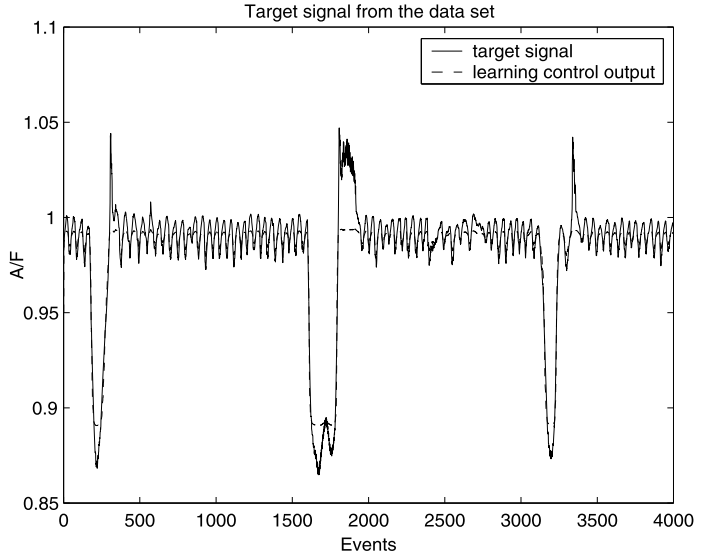

Fig. 10.14 Air–fuel ratio output generated by the neural network controller

The simulation results indicate that the present learning controller design based on adaptive dynamic programming (adaptive critic designs) is effective in training a neural network controller to track the desired TRQ and AFR sequences through proper control actions.

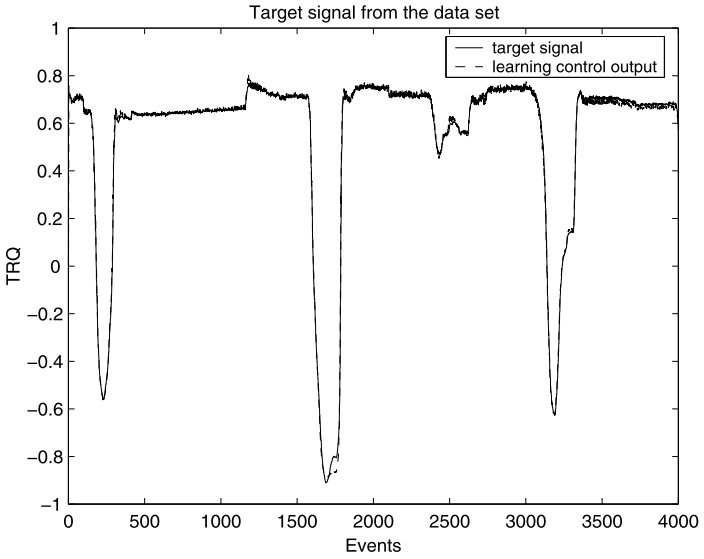

Fig. 10.15 Torque output generated by the refined neural network controller

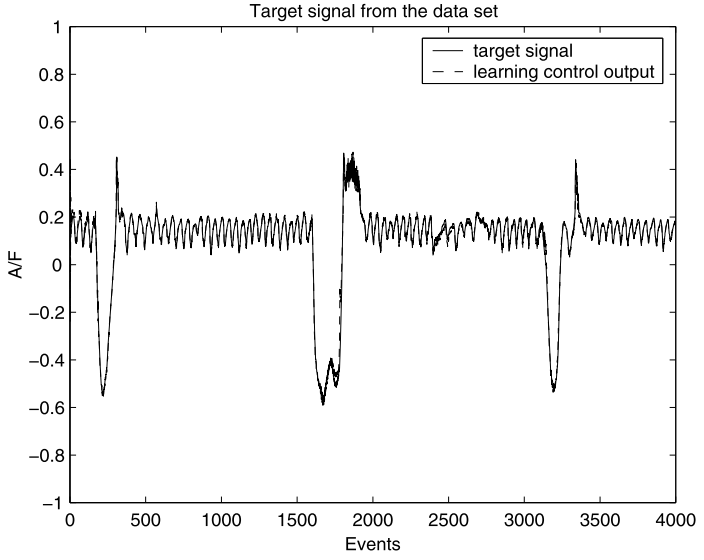

Fig. 10.16 Air–fuel ratio output generated by the refined neural network controller

10.4 Summary

In this chapter, we have investigated the optimal control problem of modern wireless networks and automotive engines by using ADP methods. First, we developed

a self-learning call admission control algorithm based on ADP for multiclass traffic in SIR-based power-controlled DS-CDMA cellular networks. The most important benefit of our self-learning call admission control algorithm is that we can easily and efficiently design call admission control algorithms to satisfy the system requirement or to accommodate new environments. We note that changes in traffic conditions are inevitable in reality. Thus, fixed call admission control policies are less preferable in applications. Simulation results showed that when the traffic condition changes, the self-learning call admission control algorithm can adapt to changes in the environment, while the fixed admission policy will suffer either from a higher new call blocking rate, higher handoff call blocking rate, or interference being higher than the tolerance.

Next a neural network learning control using adaptive dynamic programming was developed for engine calibration and control. After the network was fully trained, the present controller may have the potential to outperform existing controllers with regard to the following three aspects. First, the technique presented will automatically learn the inherent dynamics and non-linearities of the engine from real vehicle data and, therefore, do not require a mathematical model of the system to be developed. Second, the developed methods will further advance the development of a virtual power train for performance evaluation of various control strategies through the development of neural network models of the engine and transmission in a prototype vehicle. Third, the present controllers can learn to improve their performance during the actual vehicle operations, and will adapt to uncertain changes in the environment and vehicle conditions. This is an inherent feature of the present neural network learning controller. As such, these techniques may offer promise for use as real-time engine calibration tools. Simulation results showed that the present self-learning control approach was effective in achieving tracking control of the engine torque and air–fuel ratio control through neural network learning.

References

1. A guide to DECT features that influence the traffic capacity and the maintenance of high radio link transmission quality, including the results of simulations. ETSI technical report: ETR 042, July 1992. Available on-line at http://www.etsi.org
2. Ariyavisitakul S (1994) Signal and interference statistics of a CDMA system with feedback power control—Part II. IEEE Trans Commun 42:597–605
3. Bambos N, Chen SC, Pottie GJ (2000) Channel access algorithms with active link protection for wireless communication networks with power control. IEEE/ACM Trans Netw 8:583–597
4. Demuth H, Beale M (1998) Neural network toolbox user's guide. MathWorks, Natick
5. Dobner DJ (1980) A mathematical engine model for development of dynamic engine control. SAE paper no 800054
6. Dobner DJ (1983) Dynamic engine models for control development—Part I: non-linear and linear model formation. Int J Veh Des Spec Publ SP4:54–74
7. Dziong Z, Jia M, Mermelstein P (1996) Adaptive traffic admission for integrated services in CDMA wireless-access networks. IEEE J Sel Areas Commun 14:1737–1747

8. Freeman RL (1996) Telecommunication system engineering. Wiley, New York
9. Gilhousen KS, Jacobs IM, Padovani R, Viterbi AJ, Weaver LA, Wheatley CE III (1991) On the capacity of a cellular CDMA system. IEEE Trans Veh Technol 40:303–312
10. Guerin RA (1987) Channel occupancy time distribution in a cellular radio system. IEEE Trans Veh Technol 35:89–99
11. Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. IEEE Trans Neural Netw 5:989–993
12. Hong D, Rappaport SS (1986) Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoff procedures. IEEE Trans Veh Technol 35:77–92
13. Kim DK, Sung DK (2000) Capacity estimation for an SIR-based power-controlled CDMA system supporting ON–OFF traffic. IEEE Trans Veh Technol 49:1094–1100
14. Kim YW, Kim DK, Kim JH, Shin SM, Sung DK (2001) Radio resource management in multiple-chip-rate DS/CDMA systems supporting multiclass services. IEEE Trans Veh Technol 50:723–736
15. Kovalenko O, Liu D, Javaherian H (2001) Neural network modeling and adaptive critic control of automotive fuel-injection systems. In: Proceedings of IEEE international symposium on intelligent control, Taipei, Taiwan, pp 368–373
16. Lendaris GG, Paintz C (1997) Training strategies for critic and action neural networks in dual heuristic programming method. In: Proceedings of international conference on neural networks, Houston, TX, pp 712–717
17. Liu Z, Zarki ME (1994) SIR-based call admission control for DS-CDMA cellular systems. IEEE J Sel Areas Commun 12:638–644
18. Liu D, Zhang Y (2002) A new learning control approach suitable for problems with finite action space. In: Proceedings of international conference on control and automation, Xiamen, China, pp 1669–1673
19. Liu D, Xiong X, Zhang Y (2001) Action-dependent adaptive critic designs. In: Proceedings of INNS-IEEE international joint conference on neural networks, Washington, DC, pp 990–995
20. Liu D, Zhang Y, Hu S (2004) Call admission policies based on calculated power control setpoints in SIR-based power-controlled DS-CDMA cellular networks. Wirel Netw 10:473–483
21. Liu D, Zhang Y, Zhang H (2005) A self-learning call admission control scheme for CDMA cellular networks. IEEE Transactions on Neural Networks 16:1219–1228
22. Liu D, Hu S, Zhang HG (2006) Simultaneous blind separation of instantaneous mixtures with arbitrary rank. IEEE Trans Circuits Syst I, Regul Pap 53:2287–2298
23. Liu D, Xiong X, DasGupta B, Zhang HG (2006) Motif discoveries in unaligned molecular sequences using self-organizing neural networks. IEEE Trans Neural Netw 17:919–928
24. Liu D, Javaherian H, Kovalenko O (2008) Adaptive critic learning techniques for engine torque and air–fuel ratio control. IEEE Trans Syst Man Cybern, Part B, Cybern 38:988–993
25. Puskorius GV, Feldkamp LA, Davis LL (1996) Dynamic neural network methods applied to on-vehicle idle speed control. Proc IEEE 84:1407–1420
26. Ramjee R, Towsley D, Nagarajan R (1997) On optimal call admission control in cellular networks. Wirel Netw 3:29–41
27. Rappaport SS, Purzynski C (1996) Prioritized resource assignment for mobile cellular communication systems with mixed services and platform types. IEEE Trans Veh Technol 45:443–458
28. Sampath A, Holtzman JM (1997) Access control of data in integrated voice/data CDMA systems: benefits and tradeoffs. IEEE J Sel Areas Commun 15:1511–1526
29. Shin SM, Cho CH, Sung DK (1999) Interference-based channel assignment for DS-CDMA cellular systems. IEEE Trans Veh Technol 48:233–239
30. Veeravalli VV, Sendonaris A (1999) The coverage-capacity tradeoff in cellular CDMA systems. IEEE Trans Veh Technol 48:1443–1450

31. Visnevski NA, Prokhorov DV (1996) Control of a nonlinear multivariable system with adaptive critic designs. In: Proceedings of conference on artificial neural networks in engineering, St Louis, MO, pp 559–565
32. Viterbi AJ, Viterbi AM, Zehavi E (1994) Other-cell interference in cellular power-controlled CDMA. IEEE Trans Commun 42:1501–1504
33. Werbos PJ, McAvoy T, Su T (1992) Neural networks, system identification, and control in the chemical process industries. In: White DA, Sofge DA (eds) Handbook of intelligent control: neural, fuzzy, and adaptive approaches, Van Nostrand Reinhold, New York, NY