CHAPTER 23

ADAPTIVE FEATURE PURSUIT: ONLINE ADAPTATION OF FEATURES IN REINFORCEMENT LEARNING

Shalabh Bhatnagar[1], Vivek S. Borkar[2], and Prashanth L. A.[1]

[1]Department of Computer Science and Automation,

Indian Institute of Science, Bangalore 560 012, India

[2]Department of Electrical Engineering,

Indian Institute of Technology, Powai, Mumbai 400 076, India

Abstract

We present a novel feature adaptation scheme based on temporal difference learning for the problem of prediction. The scheme suitably combines aspects of *exploitation* and *exploration* by (a) finding the worst basis vector in the feature matrix at each stage and replacing it with the current best estimate of the normalized value function, and (b) replacing the second worst basis vector with another vector chosen randomly that would result in a new subspace of basis vectors getting picked. We apply our algorithm to a problem of prediction in traffic signal control and observe good performance over two different network settings.

Key Words: Online feature adaptation, temporal difference learning, traffic signal control.

23.1 INTRODUCTION

Markov decision process (MDP) [3, 4, 22] is a general framework for solving stochastic control problems. Classical solution approaches for MDP such as policy and value iteration solve the associated control problem precisely by identifying an optimal action to pick in each state. These approaches typically suffer from two major problems: (a) they require precise knowledge of the transition probabilities i.e., the system model, and (b) the amount of computation required to obtain a solution using these approaches grows exponentially with the size of the state and/or action space. Reinforcement learning (RL) [7], [5], [25] provides efficient solutions for both problems above. Many RL algorithms are incremental update stochastic approximation algorithms that work directly with real or simulated data. These algorithms make use of the averaging property of stochastic approximation as a result of which they work efficiently even when the system transition model is not known. For problems where the cardinality of the state/action space is so large that precise solutions cannot be obtained easily, one resorts to certain approximation methods. Often the value function is approximated as a function of certain parameter(s). The functional form of the approximator is also called an architecture. Architectures based on linear function approximators have been well studied in the literature because algorithms such as temporal difference (TD) learning [24] have been shown convergent when linear architectures are used [26], [27]. Here the value of a given state is approximated using the scalar product of the parameter vector with the feature associated with the state. The feature usually quantifies important state attributes. On the other hand, TD with nonlinear function approximators has been seen to diverge in some cases. An important problem that we address in this paper is to design an efficient scheme to adaptively select the 'best features' when linear function approximation is used.

The problem of feature adaptation has been studied recently for problems of prediction as well as control. In [18], features are assumed parameterized and the problem considered is one of finding the optimum feature parameter when TD learning is used. Two algorithms, one based on a gradient approach and the other based on a cross entropy method are then presented for tuning the feature parameter. Certain generalizations of the parameterized basis adaptation approach of [18] have been presented in [29]. The basis adaptation scheme there involves low order calculations. In [15], a procedure based on state aggregation and neighborhood component analysis is used for constructing basis functions assuming a linear approximation architecture. In [8], Krylov subspace basis functions, involving powers of the transition probability matrix of the associated Markov chain are used. Noisy samples of the basis functions are obtained as they cannot be computed exactly. In [17], a Laurent series expansion of the value function is used for basis construction. Another interesting work is [23] where through a recursive procedure, an 'ideal' basis function that is a representative of the value function is obtained. Whereas the above references deal with basis adaptation for the problem of prediction, in [13], the problem of control with adaptive bases is considered. Multiscale actor-critic algorithms are developed where on the 'slowest' timescale, the feature parameters are updated. Algorithms based on TD error, mean square Bellman error and mean square projected Bellman error are presented there. All of the above works consider features to be parameterized and the goal in these references is to find optimal parameters and thus the optimal features within the specific parameterized feature classes. In [20], feature adaptation for a problem of prediction is considered. Unlike in the above references, the basis functions there are not assumed parameterized. The Bellman error is included as an additional basis in each iteration, thereby increasing the dimension of the subspace at each iteration. In [14], an a priori approximation of the value function based on a simplified model as one of the features has been considered. In [1], an approach that makes use of both state aggregation and linear function approximation is presented. While the feature matrix is kept fixed there, state aggregation is done adaptively on a slower timescale using estimates of the approximate value function. In this case, one obtains locally optimal state clusters asymptotically.

In this paper, we propose a new algorithm for feature adaptation. We consider the problem of prediction (i.e., estimating the value function corresponding to a given policy) under the infinite horizon discounted cost criterion and employ the TD learning scheme with linear function approximation for this purpose. We let all feature components to take values only between 0 and 1. Upon convergence of the TD scheme for a fixed set of features, we consider the converged weights in the weight vector (i.e., the parameter whose scalar product with the state-feature approximates the value of that state). We then replace two of the columns of the feature matrix that correspond to the two smallest components of the weight vector as follows. The column of the feature matrix corresponding to the smallest component of the weight vector is replaced by the most recent estimate of the value function (suitably normalized so that all its entries are between 0 and 1) obtained after running the TD scheme for a given large number of iterations using the feature matrix of the previous step, while the column corresponding to the second smallest component of the weight vector is replaced by independent and uniformly generated random numbers between 0 and 1. Thus while the worst performer (amongst the columns of the feature matrix) is replaced by the normalized value function estimate, the second-worst performer is replaced with a random search direction to aid in exploration of better features and thereby a better subspace than the previous. The remaining columns of the feature matrix are left unchanged. The TD algorithm is then run again with the new feature matrix and the process repeated.

As an application setting, we consider the problem of estimating the value of a policy in a problem of traffic signal control [21]. The general control problem in this domain is to obtain a policy for signal switching across various sign configurations so that traffic flow is maximized and levels of congestion and hence delays at the traffic junctions are minimized. In [21], Q-learning with function approximation for a given feature matrix is applied for this problem. By keeping the optimal policy (obtained from the Q-learning algorithm in [21]) corresponding to the original feature matrix fixed, we apply our algorithm for feature adaptation and observe consistent improvement in performance over “episodes” during each of which the feature matrix is held fixed and TD is applied. We study the performance of our scheme on two different network settings and observe performance improvements in both.

The rest of the paper is organized as follows: In Section 23.2, we give the framework. The feature adaptation scheme is presented in Section 23.3. In Section 23.4, we present a partial analysis of the convergence behaviour of our scheme. In Section 23.5, the results of numerical experiments in the traffic signal control setting are shown. Finally, Section 23.6 presents the concluding remarks.

23.2 THE FRAMEWORK

An MDP is a stochastic process $\{X_n\}$ taking values in a set S (called the state space) that is governed by a control sequence $\{Z_n\}$. Let $A(i)$ be the set of feasible controls or actions in state i and $A \overset{\triangle}{=} \cup_{i\in S} A(i)$ be the set of all actions or the action space. We assume that both S and A are finite sets. The process $\{X_n\}$ satisfies the *controlled Markov property*

$$P(X_{n+1} = j \mid X_m, Z_m, m \leq n) = p(X_n, Z_n, j) \text{ a.s.},$$

where $p : S \times A \times S \rightarrow [0,1]$ is a given function for which $\sum_{j\in S} p(i,a,j) = 1$, $\forall a \in A(i), i \in S$.

A policy is a decision rule for selecting actions. We call the sequence $\bar{\pi} \overset{\triangle}{=} \{\mu_0, \mu_1, \ldots\}$ of maps $\mu_n : S \rightarrow A$ an admissible policy when $\mu_n(i) \in A(i)\ \forall i \in S$. In other words, each map μ_n in an admissible policy assigns feasible controls to any state. When $\mu_n \equiv \mu$, $\forall n \geq 0$, where μ is independent of n, we call $\bar{\pi}$ or by abuse of notation, μ itself a stationary deterministic policy (SDP).

In what follows, we consider the MDP $\{X_n\}$ to be governed by a given (fixed) SDP μ. In such a case $\{X_n\}$ is in fact a Markov chain taking values in S. Let $c(i,a)$ denote the single-stage cost when the Markov chain is in state i and action a is picked. Given $\gamma \in (0,1)$, let

$$V^\mu(i) = E\left[\sum_{m=0}^{\infty} \gamma^m c(X_m, \mu(X_m)) \mid X_0 = i\right] \tag{23.1}$$

denote the value function under SDP μ when the initial state of $\{X_n\}$ is i. The value function can be obtained by solving the system of equations

$$V^\mu(i) = c(i,\mu(i)) + \gamma \sum_{j\in S} p(i,\mu(i),j)V^\mu(j),\ i \in S, \tag{23.2}$$

or in vector notation,

$$V^\mu = c^\mu + \gamma P^\mu V^\mu, \tag{23.3}$$

where $P^\mu = [[p(i,\mu(i),j)]]_{i,j\in S}$ is the transition probability matrix of $\{X_n\}$, $c^\mu = (c(i,\mu(i)), i \in S)^T$ is the vector of single-stage costs, and $V^\mu = (V^\mu(i), i \in S)^T$ is the vector of 'values' over individual states respectively, under policy μ. The system of equations (23.3) yield the solution

$$V^\mu = (I - \gamma P^\mu)^{-1} c^\mu, \tag{23.4}$$

where I denotes the $(|S| \times |S|)$–identity matrix. When the size of the state space ($|S|$) is large, obtaining the inverse of the matrix $(I - \gamma P^\mu)$ is computationally hard. An alternative is to solve (23.3) using a value iteration procedure. However, since (23.3) corresponds to a linear system of $|S|$ equations, one expects such a procedure to be slow as well for large $|S|$.

Another problem in using a purely numerical approach is that in most real life situations, the transition probabilities $p(i, \mu(i), j)$, $i, j \in S$ are in fact not known. A way out is to use a combination of value function approximation (via a so-called approximation architecture) and stochastic approximation. The former helps to make the computational complexity manageable while the latter helps to learn the (approximated) value function with only real or simulated measurements and without any knowledge of the transition probabilities.

We approximate $V^\mu(j) \approx \phi(j)^T\theta$, where $\phi(j) = (\phi_1(j), \ldots, \phi_d(j))^T$ is a d-dimensional feature associated with state i. Also, $\theta = (\theta(1), \ldots, \theta(d))^T$ is an associated parameter. Let Φ denote the $|S| \times d$ feature matrix with $\phi(j)^T$, $j \in S$, as its rows. Thus $\Phi = [[\phi_k(s)]]_{k=1,\ldots,d, s\in S}$. Let $\phi_k \stackrel{\triangle}{=} (\phi_k(s), s \in S)^T$ denote the kth column of Φ, $k \in \{1, \ldots, d\}$.

We make the following assumptions.

Assumption 1 *The Markov chain $\{X_n\}$ under SDP μ is irreducible.*

Assumption 2 *The d columns of the matrix Φ, i.e., $\phi_1, \ldots, \phi_d$ are linearly independent. Further, $d \leq |S|$.*

From Assumption 1, $\{X_n\}$ is also positive recurrent (since $|S| < \infty$). Let $d^\mu(i)$, $i \in S$ be the stationary distribution of $\{X_n\}$ under policy μ. Further, let D^μ be a diagonal matrix with entries $d^\mu(i)$, $i \in S$ along the diagonal. The regular TD(0) algorithm is as follows:

The TD(0) Learning Algorithm

$$\theta_{n+1} = \theta_n + a(n)\delta_n \phi(X_n), \tag{23.5}$$

where δ_n is the *temporal difference term* that is defined according to

$$\delta_n = (c(X_n, \mu(X_n)) + \gamma \phi(X_{n+1})^T \theta_n - \phi(X_n)^T \theta_n),\ n \geq 0.$$

Further, $a(n), n \geq 0$ is a sequence of step-sizes that satisfy

$$\sum_n a(n) = \infty,\ \sum_n a^2(n) < \infty. \tag{23.6}$$

The parameter θ_n is d-dimensional and has components (say) $\theta_n(1), \ldots, \theta_n(d)$. Thus, $\theta_n = (\theta_n(1), \ldots, \theta_n(d))^T$. The algorithm (23.5) is called the temporal difference learning (TD(0)) algorithm for the discounted cost case and has been analyzed for its convergence in a more general setting (TD(λ) with $\lambda \in [0, 1]$) in [26]. The average cost version of this algorithm has also been analyzed in [27, 9].

It can be shown (see [26]) that the TD(0) algorithm (23.5) converges according to $\theta_n \to \theta^*$ with probability one, as $n \to \infty$, where

$$\theta^* \overset{\triangle}{=} (\theta_1^*, \ldots, \theta_d^*)^T = (\Phi^\top D^\mu (\gamma P^\mu - I)\Phi)^{-1} \Phi^\top D^\mu c^\mu.$$

In the next section, we describe our feature adaptation scheme.

23.3 THE FEATURE ADAPTATION SCHEME

We let all entries of the feature matrix Φ to take values between 0 and 1. Recall that we approximate $V^\mu(i) \approx \phi(i)^T \theta$. Thus, $V^\mu = (V^\mu(1), \ldots, V^\mu(|S|))^T$ can be approximated as

$$V^\mu \approx \Phi\theta = \begin{bmatrix} \sum_{j=1}^d \phi_j(1)\theta_j \\ \sum_{j=1}^d \phi_j(2)\theta_j \\ \cdot \\ \cdot \\ \cdot \\ \sum_{j=1}^d \phi_j(|S|)\theta_j \end{bmatrix}.$$

Alternatively, note that

$$V^\mu \approx \sum_{j=1}^d \phi_j \theta_j.$$

We run the algorithm using a nested loop sequence with the outer loop being run for a total of R iterations. The feature matrix is updated in the outer loop procedure. In between two successive updates of the outer loop, i.e., for a fixed feature matrix update, the inner loop comprising of TD recursions is run for a given (large) number M of iterates in order to obtain a final parameter corresponding to the aforementioned feature matrix (update). We call each such M-iterate run of TD for a given feature matrix an episode. Thus our scheme is run for a total of R episodes. The feature adaptation scheme is described in detail below:

The Feature Adaptation Scheme:

- *Step 0 (Initialize): Select a $|S| \times d$ feature matrix Φ^0 with columns $\phi_j^0 = (\phi_j^0(s), s \in S)^T$, $j = 1, \ldots, d$. Let Φ^0 have all its entries between 0 and 1 and let it satisfy Assumption 2. Set $r := 0$ and $n := 0$, respectively. Set $V_0^{\mu,0} = 0$ to be the initial estimate of the value function. Let $\bar{V}_0^{\mu,0} = 0$ be the initial estimate of the normalized value function. Also set M and R to be given large integers. Set θ_0^0 as the initial parameter value.*

- *Step 1: Let Φ^r denote the feature matrix during the rth step of the algorithm. The $(n+1)$st update of TD in the rth step of the algorithm is given as follows:*

$$\theta^r_{n+1} = \theta^r_n + a(n)\delta^r_n \phi^r(X_n), \tag{23.7}$$

where $a(n)$, $n \geq 0$ satisfy (23.6) and δ^r_n, $n \geq 0$ are defined according to

$$\delta^r_n = (c(X_n, \mu(X_n)) + \gamma \phi^r(X_{n+1})^T \theta^r_n - \phi^r(X_n)^T \theta^r_n).$$

Set $n := n+1$. If $n = M-1$, set $V^{\mu,r}_M = \Phi^r \theta^r_M$, and go to Step 2; else repeat Step 1.

- *Step 2: Let $\theta^r_M \stackrel{\triangle}{=} (\theta^r_{M,1}, \ldots, \theta^r_{M,d})^T$. Find $\theta^r_{M,k}$, $\theta^r_{M,l}$, $k, l \in \{1, \ldots, d\}$, $k \neq l$ such that*

$$\theta^r_{M,k} \leq \theta^r_{M,l} \leq \theta^r_{M,j} \ \forall j \in \{1, \ldots, d, j \neq k, j \neq l\}.$$

(Any tie is resolved according to some prescribed rule). Obtain a new feature matrix Φ^{r+1} as follows: First normalize $V^{\mu,r}_M$ (by first letting any negative component to be zero and then dividing every component by one with the largest value) to obtain $\bar{V}^{\mu,r}_M$. Thus, all entries of $\bar{V}^{\mu,r}_M$ lie between 0 and 1. Set $\phi^{r+1}_k := \bar{V}^{\mu,r}_M$. Next obtain ϕ^{r+1}_l by picking all its entries independently according to the uniform distribution on $[0,1]$. Retain the remaining columns in the Φ^{r+1} matrix from the matrix Φ^r itself, i.e., set $\phi^{r+1}_i := \phi^r_i$ for all $i \in \{1, \ldots, d\}$ with $i \neq k, l$. Set Φ^{r+1} to be the matrix with columns ϕ^{r+1}_j, $j = 1, \ldots, d$. Set $r := r+1$. If $r < R$, go to Step 1; else go to Step 3.

- *Step 3 (Termination): Output θ^R_M as the final parameter value and $V^{\mu,R}_M = \Phi^R \theta^R_M$ as the corresponding value function estimate.*

We summarize the notation used with regards to the value function below (as this will also be used in the convergence analysis):

- V^μ: exact value function under policy μ,
- $V_n^{\mu,r} = \Phi^r \theta_n^r$: estimate of the value function at the rth iterate of algorithm after the nth run of Step 1 ($M \geq n \geq 0$),
- $V_M^{\mu,r} = \Phi^r \theta_M^r$: estimate of the value function after completion of Step 1 during the rth iterate of algorithm (i.e., with $n = M$),
- $V_*^{\mu,r} = \Phi^r \theta_*^r$: estimate of the value function from Step 1 of algorithm at the rth iterate if Step 1 was run until convergence of TD, i.e., where $\theta_*^r = \lim\limits_{M \to \infty} \theta_M^r$ with probability one,
- $\bar{V}_M^{\mu,r}$: normalized estimate of value function after completion of Step 1 during the rth iterate of algorithm,
- $\check{V}_r^\mu$: projection of V^μ to S_r.

Note above that we obtain the normalized value function $\bar{V}_M^{\mu,r}$ at each iterate r of the algorithm only after termination of Step 1 on that iterate. Convergence of the TD scheme for a given feature matrix Φ^r, as $M \to \infty$, has been shown in the literature [26], [7] under the weighted Euclidean norm $\| \cdot \|_{D^\mu}$ defined according to

$$\| x \|_{D^\mu} = \sqrt{x^\top D^\mu x},$$

for any $x \in \mathcal{R}^{|S|}$. Let S_r, $r = 1, 2, \ldots$ denote the subspace

$$S_r = \{\Phi^r \theta \mid \theta \in \mathcal{R}^d\}.$$

Let $\check{V}_r^{\mu}$ represent the best approximation to the value function V^{μ} within the subspace S_r. This is obtained by simply projecting V^{μ} to S_r. Note that TD (or for that matter any other algorithm that operates only within the subspace S_r) does not in general converge to $\check{V}_r^{\mu}$. In fact the parameters θ_n^r, $n \geq 0$ obtained from TD, see Step 1 of the feature adaptation scheme (above), converge to θ_*^r almost surely where

$$\theta_*^r = (\Phi^{r\top} D^{\mu}(\gamma P^{\mu} - I)\Phi^r)^{-1} \Phi^{r\top} D^{\mu} c^{\mu}.$$

This would correspond to a value function estimate of $V_*^{\mu,r} = \Phi^r \theta_*^r$ which is not the same as $\check{V}_r^{\mu}$ even though both $V_*^{\mu,r}$ and $\check{V}_r^{\mu}$ are vectors in the subspace S_r. Since we run TD for a fixed number M of instants during each visit of the algorithm to Step 1, the estimate of the value function as given by TD after the rth cycle (of M iterates) is $V_M^{\mu,r} = \Phi^r \theta_M^r$. By choosing M sufficiently large, one can bring down the difference between $V_M^{\mu,r}$ and $V_*^{\mu,r}$.

The key idea behind our adaptation scheme is the following: Since one replaces the ‘worst basis vector’ by the current best estimate of the value function and the ‘next-to-worst basis vector’ by uniformly generated random numbers, one ensures that (a) the best linear combination of the basis functions in the current subspace is retained while (b) in addition, the scheme does a random exploration in order to find a potentially better subspace than the current. As our experiments demonstrate, it is indeed seen to be the case that our adaptive scheme results in significant performance improvement.

Remark 1 *One possible variation is to pick new basis vectors from a prespecified overcomplete basis. These vectors could be sparsely represented. Nevertheless, the running estimate of the value function need not have a sparse representation. Thus the algorithm stores all but one sparse vectors, thus saving on storage.*

Remark 2 *Another possible tweak is to perform a Gram-Schmidt step on each new independent basis vector picked relative to the rest and replace it by the unit normal vector thus obtained. While this is aesthetically more pleasing and will avoid certain 'low probability' pathologies, it has a computational overhead. We did not use this tweak in our experiments as we obtained good results regardless.*

Remark 3 *A good error estimate for fixed point of a projected linear contraction relative to fixed point of the contraction itself, that holds with 'high probability' under certain dimensionality requirements on the range of the projection, is given in [2].*

In the next section, we give a partial analysis of the convergence behaviour of our feature adaptation scheme.

23.4 CONVERGENCE ANALYSIS

We first introduce some notation. Let U denote the unit sphere in $\mathcal{R}^{|S|}$. Since we normalize the basis vectors, they can be identified with points in U. Let $V^{\perp}$ denote the subspace of $\mathcal{R}^{|S|}$ orthogonal to V^{μ} (recall that V^{μ} is the true value function). Let $B \subset U$ denote the ϵ-neighborhood of $V^{\perp} \cap U$ in U for a small $\epsilon > 0$. Let D denote the ϵ-neighborhood of V^{μ}. Let Π_r denote the projection to $Range(\Phi^r)$.

Then $\check{V}_r^{\mu} = \Pi_r(V^{\mu})$. Let $\Psi : \mathcal{R}^{|S|} \times \mathcal{R}^{|S|} \to \mathcal{R}^{|S|}$ denote the map

$$\Psi(x, y) := \frac{x - \langle x, y/\|y\| \rangle y/\|y\|}{\|x - \langle x, y/\|y\| \rangle y/\|y\|\|},$$

i.e., the component of x orthogonal to y, normalized to have unit norm. Let

$$\beta_r = \frac{\| V^{\mu} - V_*^{\mu,r} \|}{\| V^{\mu} - \check{V}_r^{\mu} \|}.$$

We have the following lemma.

Lemma 1 *(a)* $\beta_r \in \left[1, \dfrac{1}{1-\gamma}\right] \forall r \geq 0.$

(b) For β_r *close to one (for any* $r \geq 0$*),* $V_*^{\mu,r} \approx \check{V}_r^{\mu}$*.*

Proof: (a) It is easy to see that the vectors $V^{\mu} - V_*^{\mu,r}$, $V^{\mu} - \check{V}_r^{\mu}$ and $\check{V}_r^{\mu} - V_*^{\mu,r}$ form a right angle triangle with $V^{\mu} - V_*^{\mu,r}$ as the hypotenuse. Further, $V^{\mu} - \check{V}_r^{\mu}$ and $\check{V}_r^{\mu} - V_*^{\mu,r}$ form a right angle with $\check{V}_r^{\mu} - V_*^{\mu,r}$ being a vector within the subspace S_r and $V^{\mu} - \check{V}_r^{\mu}$ being orthogonal to S_r. Thus,

$$\| V^{\mu} - V_*^{\mu,r} \| \geq \| V^{\mu} - \check{V}_r^{\mu} \|,$$

implying that $\beta_r \geq 1 \ \forall r \geq 0$. Further, from Lemma 6 of [26], we also obtain that

$$\| V^{\mu} - V_*^{\mu,r} \| \leq \frac{1}{(1-\gamma)} \| V^{\mu} - \check{V}_r^{\mu} \| .$$

Thus we also have $\beta_r \leq \dfrac{1}{1-\gamma} \ \forall r \geq 0$.

(b) By the Pythagorean theorem, we have

$$\| V^{\mu} - V_*^{\mu,r} \|^2 = \| V^{\mu} - \check{V}_r^{\mu} \|^2 + \| \check{V}_r^{\mu} - V_*^{\mu,r} \|^2 . \tag{23.8}$$

From (23.8), we obtain

$$\beta_r^2 = 1 + \frac{\| \check{V}_r^{\mu} - V_*^{\mu,r} \|^2}{\| V^{\mu} - \check{V}_r^{\mu} \|^2},$$

or that

$$\| \check{V}_r^{\mu} - V_*^{\mu,r} \| = \sqrt{\beta_r^2 - 1} \, \| V^{\mu} - \check{V}_r^{\mu} \|$$

$$\leq 2K\sqrt{\beta_r^2 - 1},$$

where K is an upper bound on $\| V^{\mu} \|$. Thus if β_r is close to one, we have that $V_*^{\mu,r} \approx \check{V}_r^{\mu}$. □

Now let $\nu > 0$ be such that $\| \check{V}_r^{\mu} - V_*^{\mu,r} \| < \nu$. Let $\alpha > 0$ be such that if $V \notin D$ and $z \notin B$ is a unit vector, then for $\Pi' :=$ projection onto $Range(\Phi^r)$ and $\Pi'' :=$ projection onto span$(Range(\Phi^r) \cup \{z\})$, we have

$$\|V - \Pi''(V)\| \leq \|V - \Pi'(V)\| - \alpha.$$

Next, we will consider the actual error $\| V_M^{\mu,r} - V^{\mu} \|$ and show that it decreases on the average as r is increased when M is large enough. We have the following result.

Theorem 23.1 *$\exists \alpha_0 > 0$ and $M_0 > 0$ such that for $\alpha > \alpha_0$, and $M \geq M_0$,*

$$E[\| V_M^{\mu,r+1} - V^{\mu} \|] < E[\| V_M^{\mu,r} - V^{\mu} \|].$$

Proof: Note that because of the finite (M) step termination of the TD update in between two successive updates of the feature matrix, the quantity $V_M^{\mu,r}$ is random for any given $r \geq 0$. The dynamics of $\{V_M^{\mu,r}, r \geq 0\}$ can be described according to

$$V_M^{\mu,r+1} = V_*^{\mu,r+1} + \eta((V_M^{\mu,r}, \xi_{r+1}), \zeta_{r+1}),$$

where:

1. the i.i.d. variables $\{\xi_r\}$ denote the new basis vectors being inducted,

2. ζ_{r+1} captures the random inputs of the algorithm with the fixed basis, between the rth and $(r+1)$th basis change, and,

3. $\eta_{r+1} := \eta((V_M^{\mu,r}, \xi_{r+1}), \zeta_{r+1}) = V_M^{\mu,r+1} - V_*^{\mu,r+1}$ is the error due to inexact convergence (because of the finite run of TD).

Let $\chi_r := V_*^{\mu,r} - \check{V}_r^{\mu}$. Further, as before, let K be an upper bound on $\| V^\mu \|$. Note that by Corollary 14, Chapter 4 of [10], $\exists M_0$ such that for $M \geq M_0$, η_{r+1} can be bounded by a given $\kappa > 0$ with probability $1 - \delta$, $\delta > 0$ prescribed. Let

$$\begin{aligned} A_r &= \{\Psi(\xi_{r+1}, V_M^{\mu,r}) \in B\}, \\ C_r &= \{\|\eta_{r+1}\| \geq \kappa\}. \end{aligned}$$

Then for $\epsilon_1 > P(A_r)$, we have

$$\begin{aligned} E[\|V_M^{\mu,r+1} - V^\mu\|] &\leq E[\|V_*^{\mu,r+1} - V^\mu\|] + E[\|\eta_{r+1}\|] \\ &\leq E[\|\check{V}_{r+1}^{\mu} - V^\mu\|] + E[\|\chi_{r+1}\|] + E[\|\eta_{r+1}\|] \\ &\leq E[\|\check{V}_{r+1}^{\mu} - V^\mu\| I_{A_r}] + E[\|\check{V}_{r+1}^{\mu} - V^\mu\| I_{A_r^c}] \\ &\quad + E[\|\chi_{r+1}\|] + E[\|\eta_{r+1}\|] \\ &\leq E[\|\check{V}_r^{\mu} - V^\mu\|] - \alpha + 2\epsilon_1 K + E[\|\chi_{r+1}\|] + E[\|\eta_{r+1}\|] \\ &\leq E[\|V_M^{\mu,r} - V^\mu\|] - \alpha + 2\epsilon_1 K + E[\|\chi_r\|] + E[\|\eta_r\|] \\ &\qquad\qquad + E[\|\chi_{r+1}\|] + E[\|\eta_{r+1}\|] \\ &\leq E[\|V_M^{\mu,r} - V^\mu\|] - \alpha + 2((\epsilon_1 + \delta)K + (1-\delta)\kappa + \nu). \end{aligned}$$

Let $\alpha_0 := 2((\epsilon_1 + \delta)K + (1-\delta)\kappa + \nu)$. Then for $\alpha > \alpha_0$,

$$E[\|V_M^{\mu,r+1} - V^\mu\|] < E[\|V_M^{\mu,r} - V^\mu\|].$$

Further, $\alpha > \alpha_0$ will hold if $\delta, \nu, \kappa, \epsilon_1$ are sufficiently small and $V_M^{\mu,r} \notin D$. □

From Theorem 23.1, the average norm difference between the true value function and the approximate value function to which TD(0) converges for a given set of bases decreases as the set of bases is updated after each update of the basis adaptation scheme.

23.5 APPLICATION TO TRAFFIC SIGNAL CONTROL

We consider the problem of traffic signal control as an application setting. The aim in general for this problem is to find an optimal policy for switching signals at traffic junctions in a road network so as to maximize flows and minimize delays. The signals that can be switched to green simultaneously form a sign configuration. Q-learning with linear function approximation has been applied for this control problem in [21]. We, however, consider the problem of adaptively tuning the feature matrix when actions are chosen according to a given policy. The policy that we use is obtained using the Q-learning algorithm from [21], i.e., after convergence of the Q-learning algorithm. We apply the feature adaptation scheme described in Section 23.3.

Consider a road network with $m \geq 1$ junctions and N signalled lanes. We let the state be the vector of queue lengths on individual lanes and of the elapsed times since the signal turned red on each of those lanes that have a traffic signal (at the various junctions). We assume that there is a central controller that receives the state information from the various lanes and based on the given policy, conveys information to the individual junctions regarding which traffic lights to switch green during a cycle. We assume no delays in the transfer of information from (to) the controller to (from) the various lanes. (Accounting for bounded delays would add another asymptotically vanishing error term – see Chapter 7 of [10].) The state at instant n is the tuple

$$s_n = (q_1(n), \ldots, q_N(n), t_1(n), \ldots, t_N(n)),$$

where $q_1(n), \ldots, q_N(n)$ are the queue lengths on each of the N lanes at instant n. Similarly, $t_1(n), \ldots, t_N(n)$ are the elapsed times on each of the above lanes. The actions a_n correspond to the sign configurations, i.e., feasible combination of traffic lights to switch at each of the m junctions in the road network. Thus, $a_n = (a_1(n), \ldots, a_m(n))$, where $a_i(n)$ is the sign configuration at junction i in the time slot n. We only allow those sign configurations to be in the action set that are feasible and ignore all other (infeasible) sign configurations. This helps keep the computational complexity manageable.

As with [21], we allow lanes on the main road to have a higher priority than those on the side roads. This is accomplished through the form of the cost function as explained below. Let I_p denote the set of lanes that are on the main road. Then the form of the cost function $c(s_n, a_n)$ is given by

$$\begin{aligned} c(s_n, a_n) = & \quad r_1 * (\textstyle\sum_{i \in I_p} r_2 * q_i(n) + \sum_{i \notin I_p} s_2 * q_i(n)) \\ + & \quad s_1 * (\textstyle\sum_{i \in I_p} r_2 * t_i(n) + \sum_{i \notin I_p} s_2 * t_i(n)). \end{aligned}$$

Here $r_i, s_i \geq 0$ are certain weights that satisfy $r_i + s_i = 1, i = 1, 2$. Further, we let $r_2 > s_2$. In our experiments, we let $r_1 = s_1 = 0.5$. Thus, we assign equal

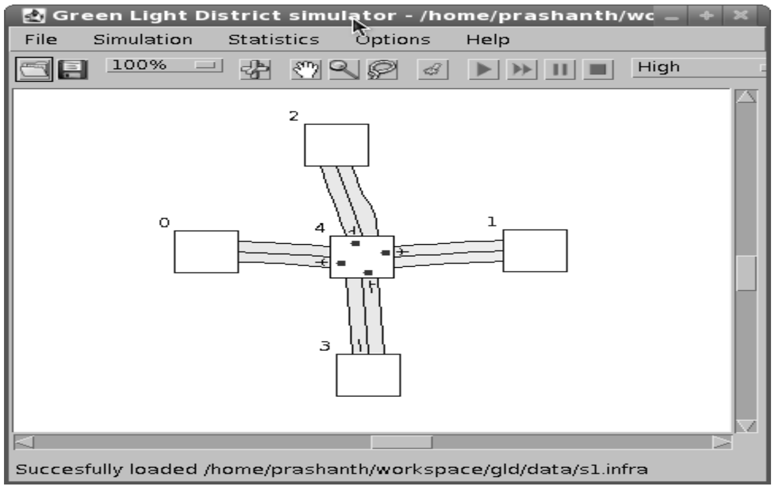
Green Light District simulator - /home/prashanth/wc
File
Simulation
Statistics
Options
Help
100%
High
2
0
4
1
3
Succesfully loaded /home/prashanth/workspace/gld/data/s1.infra

Figure 23.1 A Single-Junction Road Network

weightage to both queue lengths and elapsed times. Further, we let $r_2 = 0.6$ and $s_2 = 0.4$, respectively. Thus, queue lengths and elapsed times for lanes on the main road are weighted more than those on the side roads. We let the discount factor $\gamma = 0.9$ in the experiments.

We consider two different road traffic networks: (a) a network with a single traffic signal junction and (b) a corridor with two traffic signal junctions. The two networks are shown in Figures 23.1 and 23.2, respectively. We implemented these network settings and our algorithm on the green light district (GLD) open source software for road traffic simulations [28].

We study the performance of our feature adaptation scheme using estimates of $E[\| V_M^{\mu,r} \|]$. Recall that from Theorem 23.1, the estimates $E[\| V_M^{\mu,r} - V^\mu \|]$ diminish with r. The value function V^μ, however, is not available, so we use the fact that by the foregoing, $E[\| V_M^{\mu,r} \|]$ will tend to increase. For estimating $E[\|$

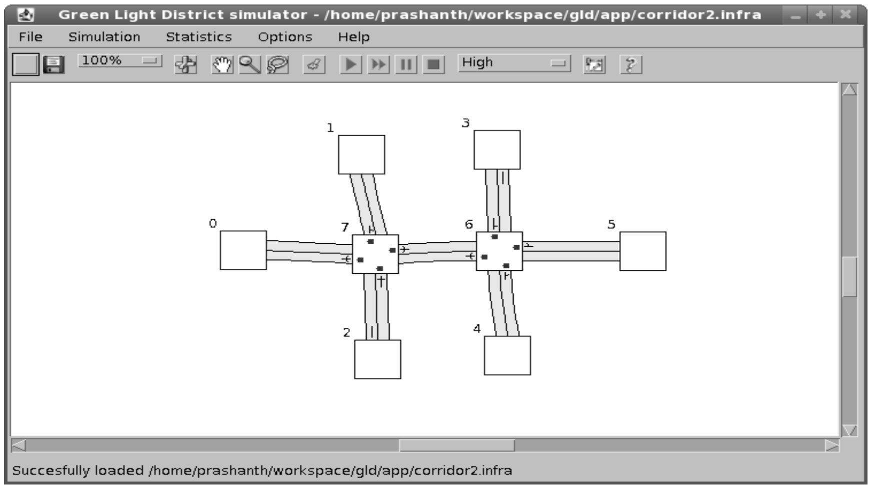
Green Light District simulator - /home/prashanth/workspace/gld/app/corridor2.infra
File
Simulation
Statistics
Options
Help
100%
High
0
1
2
3
4
5
6
7
Succesfully loaded /home/prashanth/workspace/gld/app/corridor2.infra

Figure 23.2 A Corridor Network with Two Junctions

$V_M^{\mu,r}$ $\|]$, we use the sample averages of the estimates of $\|$ $V_M^{\mu,r}$ $\|$. As mentioned previously, we let $V_n^r = \Phi^r \theta_n^r$ denote the nth estimate of the value function when the feature matrix is Φ^r. We obtain the aforementioned sample averages by running the recursion

$$Z_{n+1}^r = (1-a)Z_n^r + a \parallel V_n^r \parallel,$$

where for any given $r \in \{0, 1, \ldots, R-1\}$, the above recursion is run for M iterations i.e., with $n \in \{0, 1, \ldots, M-1\}$. Next, the value of r is updated and the above procedure repeated. Here a is a small step-size that we select to be 0.001. By abuse of notation, we denote Z_n^r as Z_m in these figures where m denotes the number of cycles or time instants (in absolute terms) i.e., $m \in \{0, 1, \ldots, RM-1\}$, when Z_m is updated.

We call each group of M cycles when the feature matrix Φ^r is held fixed for some r, an episode. We conducted our experiments for the cases of single-junction

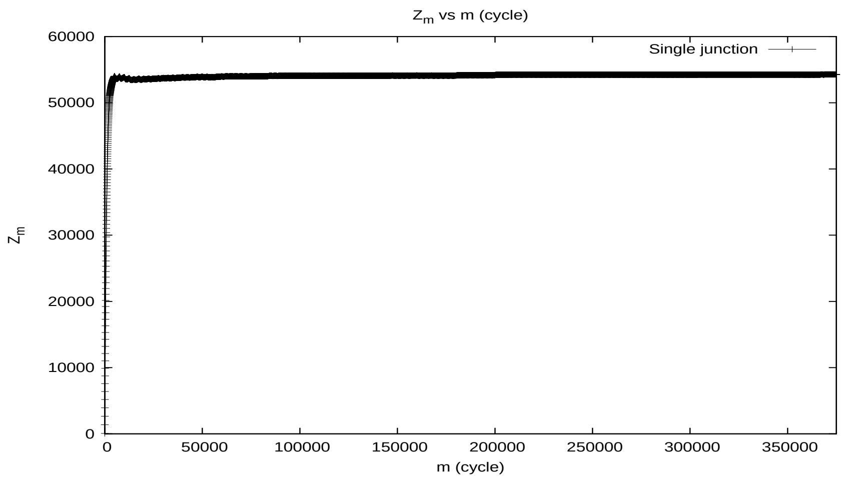

Figure 23.3 Plot of Z_m vs. m in the Case of Single-Junction Road Network

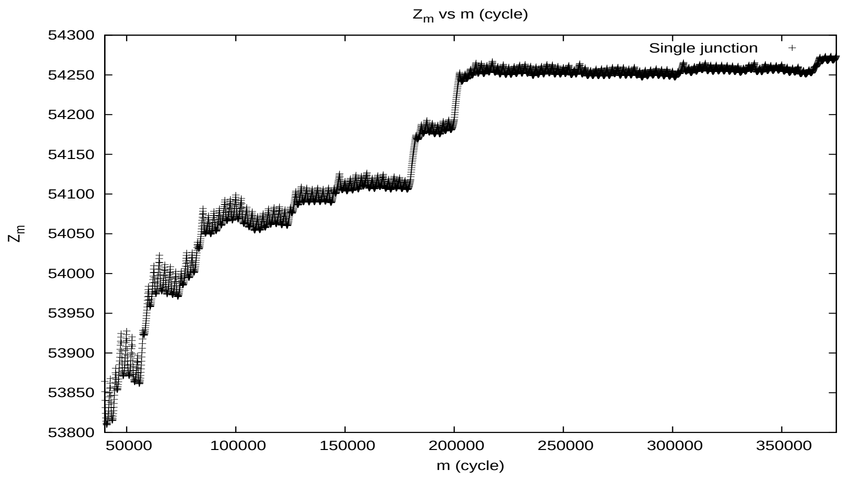

Figure 23.4 Plot of Z_m vs. m after 40,000 Cycles in the Case of Single-Junction Road Network

and two-junction-corridor networks for a total of 150 episodes in each where each episode comprised of 2,500 cycles. Thus $M = 2500$ and $R = 150$ in our experiments.

Table 23.1 Performance Improvement with Feature Adaptation for the Single-Junction Road Network

# Cycle (m)	Z_m	$Z_m - Z_{M-1}$, ($m \geq M-1$)
2499	51042.23	
74999	54003.00	2960.76
149999	54116.59	3074.36
224999	54260.28	3218.05
299999	54255.38	3213.15
374999	54274.72	3232.49

For the single-junction case, we show in Figure 23.3 the plot of Z_m as a function of m (the number of cycles). We observe that there is a significant improvement after the first episode which results in a steep jump in the Z_m value. In subsequent (r) iterations when the Φ^r matrix is updated, the performance improvement continues, though in smaller steps. The improvement in performance from feature adaptation can be seen more clearly in Figure 23.4, when the values of Z_m are plotted for $m \geq 40,000$. The value of Z_m as well as the difference $Z_m - Z_{M-1}$ i.e., improvement in 'Z_m performance' in relation to its value obtained after the completion of the first episode (i.e., with the originally selected feature matrix) is shown at the end of the 30th, 60th, 90th, 120th and 150th episodes respectively, in Table 23.1. As expected, the values of Z_m are seen to consistently increase here.

Next, for the case of the two-junction corridor road network, we show a similar plot of Z_m as a function of m in Figure 23.5. Further, in Figure 23.6, we show the same plot for cycles 40,000 onwards to show performance improvement resulting

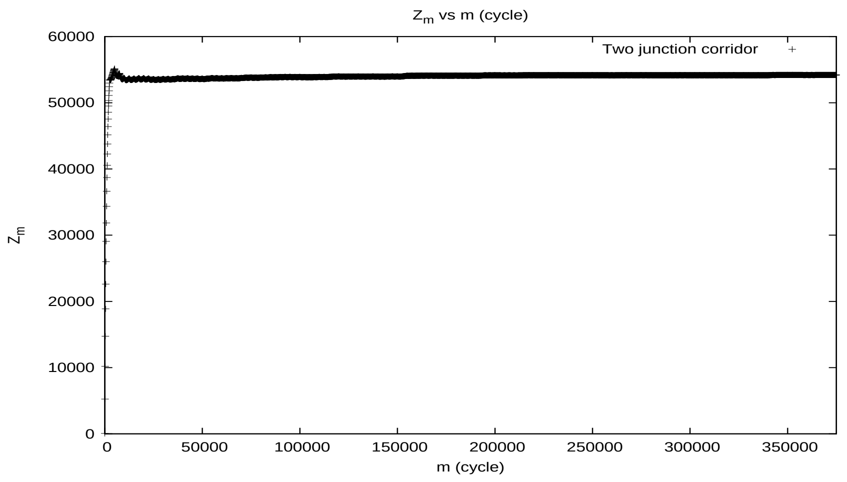

Figure 23.5 Plot of Z_m vs. m in the Case of Two-Junction Corridor Network

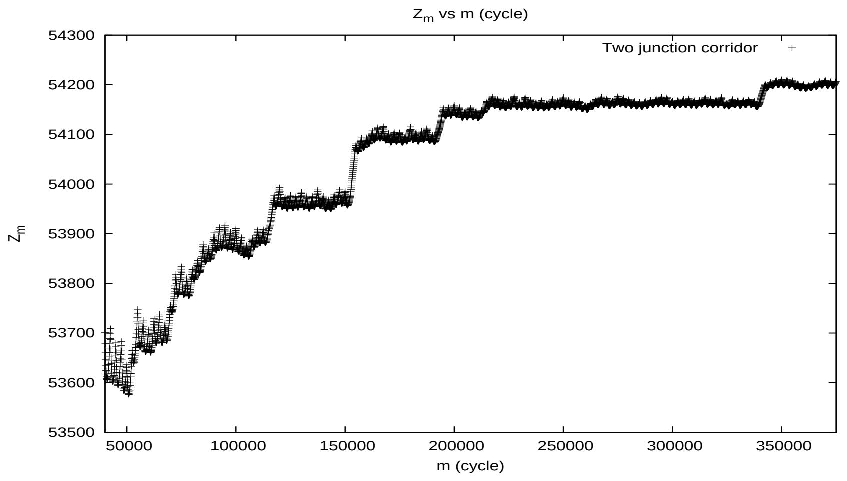

Figure 23.6 Plot of Z_m vs. m after 40,000 Cycles in the Case of Two-Junction Corridor Network

from feature adaptation. Similar observations as for the single-junction case hold in the case of the two-junction corridor as well.

Table 23.2 Performance Improvement with Feature Adaptation for the Two-Junction Corridor Road Network

# Cycle (m)	Z_m	$Z_m - Z_{M-1}$ ($m \geq M-1$)
2499	53480.65	
74999	53834.04	353.39
149999	53985.54	504.89
224999	54166.69	686.04
299999	54167.99	687.34
374999	54207.78	727.13

Finally, as before, we show in Table 23.2, the values of Z_m as well as of the difference $Z_m - Z_{M-1}$ at the end of the 30th, 60th, 90th, 120th and 150th episodes respectively. The values of Z_m are again seen to consistently increase here as well.

23.6 CONCLUSIONS

We presented in this paper a novel online feature adaptation algorithm. We observed significant performance improvements on two different settings for a problem of traffic signal control. We considered the problem of prediction here and applied our feature adaptation scheme in conjunction with the temporal difference learning algorithm. As future work, one may consider the application of our algorithm together with other schemes such as least squares temporal difference (LSTD) learning [12], [11] and least squares policy evaluation (LSPE) [19], [6]. Moreover, one may apply a similar scheme for the problem of control, for instance, in conjunction with the actor-critic algorithms in [16] and [9].

Acknowledgements

V. S. Borkar was supported in his research through a J. C. Bose Fellowship from the Department of Science and Technology, Government of India. The work of S. Bhatnagar and Prashanth L. A. was supported by the Automation Systems Technology (ASTec) Center, a program of the Deparment of Information Technology, Government of India.

REFERENCES

1. Baras, J. S. and Borkar, V. S. (2000) “A learning algorithm for Markov decision processes with adaptive state aggregation”, *Proceedings of the 39th IEEE Conference on Decision and Control, Dec. 12 – 15, 2000*, vol.4, Sydney, Australia: 3351 - 3356.
2. Barman, K. and Borkar, V. S. (2008) “A note on linear function approximation using random projections”, *Systems and Control Letters*, 57(9):784-786.
3. Bertsekas, D. P. (2005) *Dynamic Programming and Optimal Control, Vol.I (3rd ed.)*, Athena Scientific, Belmont, MA.
4. Bertsekas, D. P. (2007) *Dynamic Programming and Optimal Control, Vol.II (3rd ed.)*, Athena Scientific, Belmont, MA.
5. Bertsekas, D. P. (2011) “Approximate Dynamic Programming”, (Online) Chapter 6 of *Dynamic Programming and Optimal Control Vol.II, (3rd ed.)*.
URL: http://web.mit.edu/dimitrib/www/dpchapter.html
6. Bertsekas, D. P.; Borkar, V. S. and Nedic, A. (2004) “Improved temporal difference methods with linear function approximation”, *Handbook of Learning and Approximate Dynamic Programming by A.Barto, W.Powell, J.Si (Eds.)*, pp.231-255, IEEE Press.
7. Bertsekas, D. P. and Tsitsiklis, J. N. (1996) *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA.
8. Bertsekas, D. P. and Yu, H. (2009) “Projected equation methods for approximate solution of large linear systems”, *Journal of Computational and Applied Mathematics*, 227: 27-50.
9. Bhatnagar, S.; Sutton, R. S.; Ghavamzadeh, M. and Lee, M. (2009) “Natural actor-critic algorithms”, *Automatica*, 45: 2471–2482.
10. Borkar, V. S. (2008) *Stochastic Approximation: A Dynamical Systems Viewpoint*, (Jointly published by) Cambridge University Press, Cambridge, U. K. and Hindustan Book Agency, New Delhi, India.
11. Boyan, J. A. (1999) “Least-squares temporal difference learning”, *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 49–56. Morgan Kaufmann, San Francisco, CA.

12. Bradtke, S. J. and Barto, A. G. (1996) “Linear least-squares algorithms for temporal difference learning”, *Machine Learning*, 22:33-57.

13. Di Castro, D. and Mannor, S. (2010) “Adaptive bases for reinforcement learning”, *Machine Learning and Knowledge Discovery in Databases, Proceedings of ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Part I*, José Luis Balcźar, Francesco Bonchi, Aristides Gionis and Michèle Sebag (eds.), Lecture Notes in Computer Science Volume 6321: 312-327.

14. Huang, D.; Chen, W.; Mehta, P.; Meyn, S. and Surana, A. (2011) “Feature Selection for neuro-dynamic programming”, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control (Ed. F.L.Lewis and D.Liu)*, IEEE Press Computational Intelligence Series, Chapter 24, this volume.

15. Keller, P. W.; Mannor, S. and Precup, D. (2006) “Automatic basis function construction for approximate dynamic programming and reinforcement learning”, *Proceedings of the 23rd International Conference on Machine Learning, June 25 – 29, 2006*, Pittsburgh, PA.

16. Konda, V. R. and Tsitsiklis, J. N. (2003) “On actor–critic algorithms”, *SIAM Journal on Control and Optimization*, 42(4):1143-1166.

17. Mahadevan, S. and Liu, B. (2010) “Basis construction from power series expansions of value functions”, *Proceedings of Advances in Neural Information Processing Systems*, Vancouver, B.C., Canada.

18. Menache, I.; Mannor, S. and Shimkin, N. (2005) “Basis function adaptation in temporal difference reinforcement learning”, *Annals of Operations Research*, 134:215-238.

19. Nedic, A. and Bertsekas, D. P. (2003) “Least-squares policy evaluation algorithms with linear function approximation”, *Journal of Discrete Event Systems*, 13:79-110.

20. Parr, R., Painter-Wakefield, C., Li, L. and Littman, M. (2007) “Analyzing feature generation for value-function approximation”, *Proceedings of the 24th International Conference on Machine Learning, June 20 – 24, 2007*, Corvallis, OR.

21. Prashanth, L. A. and Bhatnagar, S. (2011) Reinforcement learning with function approximation for traffic signal control, *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412-421.

22. Puterman, M. L. (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley, New York.

23. Sun, Y. Gomez, F. Ring, M. and Schmidhuber, J. (2011) “Incremental basis construction from temporal difference error”, *Proceedings of the Twenty Eighth International Conference on Machine Learning*, Bellevue, WA, USA.

24. Sutton, R. S. (1988) “Learning to predict by the method of temporal differences”, *Machine Learning*, 3:9–44.

25. Sutton, R. S. and Barto, A. (1998) *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

26. Tsitsiklis, J. N. and Van Roy, B. (1997) “An analysis of temporal difference learning with function approximation”, *IEEE Transactions on Automatic Control*, 42(5):674-690.

27. Tsitsikis, J. and Van Roy, B. (1999) “Average cost temporal-difference learning”, *Automatica*, 35:1799-1808.

28. Wiering, M. Vreeken, J. van Veenen, J. and Koopman, A. (2004) “Simulation and optimization of traffic in a city”, *IEEE Intelligent Vehicles Symposium*, pp: 453-458.

29. Yu, H. and Bertsekas, D. P. (2009) “Basis function adaptation methods for cost approximation in MDP”, *Proceedings of the IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning, March 30 – April 2, 2009*, Nashville, TN, USA.