TECHNICAL RESEARCH REPORT

Simulation-Based Algorithms for Average Cost Markov Decision Processes

by Ying He, Michael C. Fu, Steven I. Marcus

T.R. 99-56

Technical Report

Simulation-Based Algorithms for Average Cost Markov Decision Processes

Ying He yhe@isr.umd.edu
Michael C. Fu mfu@isr.umd.edu
Steven I. Marcus marcus@isr.umd.edu
Institute for Systems Research
University of Maryland
College Park, MD 20742
http://www.isr.umd.edu/IPDPM/

July, 1999

Abstract

In this paper, we give a summary of recent development of simulation-based algorithms for average cost MDP problems, which are different from those for discounted cost problems or shortest path problems. We introduce both simulation-based policy iteration algorithms and simulation-based value iteration algorithms for average cost problem, and give the pros and cons of each algorithm.

Keywords: Simulation-Based Policy Iteration, Simulation-Based Value Iteration, Markov Decision Processes, Average Cost, Unichain.

1 Introduction

Optimization problems with average cost criterion are common in economic, computer and communication systems. Some examples are inventory control problems and computer and communication networks, where decisions are made based on throughput rate or average time a job or packet remains in the system [1]. One approach for solving these optimization problems is to formulate them in the Markov Decision Process (MDP) framework, by defining appropriate states, actions, transition probabilities, time horizon, and cost criterion [1]. Due to the notorious "curse of dimensionality" and "curse of modeling" (the decision maker does not have access to the exact mathematical model of the MDP) problems, there has been a lot of interest in simulation-based algorithms for MDP [2]. Two basic algorithms for solving MDPs are policy iteration and value iteration. Policy iteration includes a sequence of policy evaluation and policy improvement at each iteration. The policy evaluation step involves solving linear equations with the same number of equations as the number of states. Value iteration calculates the optimal cost-to-go successively by turning the Bellman optimality equation into an update rule. Simulation-based algorithms, accordingly, can be roughly divided into simulation-based policy iteration (SBPI) and simulation-based value iteration (SBVI) algorithms.

For problems with average cost criterion, simulation-based algorithms are different from those for discounted cost problems or shortest path problems, where most of the research on simulation-based algorithms has been focused. The differences originate from the corresponding exact policy iteration and value iteration. First, with average cost problems, the evaluated costs are the average cost and differential costs, instead of the total costs used in discounted cost problems or shortest path problems. Second, results and analyses of simulation-based algorithms on average cost problems depend on the chain structure of the transition matrices of Markov chains generated by stationary policies, whereas those on discounted cost problems or shortest path problems do not.

On the basis of the chain structure, MDPs for average cost problems can be classified as recurrent, unichain, or multichain [1]. An MDP is *recurrent* if the transition matrix corresponding to every deterministic stationary policy consists of a single recurrent class. Under this assumption, Cao [3] proposed two single-path algorithms and provided convergence conditions; Tsitsiklis and Van Roy [4] extended the temporal-difference learning algorithm from the discounted cost case to the average cost case; Konda [5] also gave several actor-critic algorithms, simulation schemes derived from policy iteration, with average cost criterion and provided convergence proofs. However, the recurrent assumption is often not satisfied by problems of practical interest.

An MDP is *unichain* if the transition matrix corresponding to every deterministic stationary policy consists of one single recurrent class plus a possibly empty set of transient states, whereas it is *multichain* if the transition matrix corresponding to at least one stationary policy consists of two or more recurrent classes. Under the assumption that the MDP is unichain and that there is at least one common state that is recurrent under all policies, Bertsekas [6] converted the average cost problem into a stochastic shortest path problem,

and provided a multi-run scheme and corresponding error bound for SBPI. In this algorithm, the reference state is the same from iteration to iteration, and the differential costs are obtained by subtracting from expected total cost the product of the average cost and expected number of transitions. The simulation-based version of value iteration, Q-learning, has also been studied under the unichain plus common recurrent state assumption, e.g., Abounadi, Bertsekas and Borkar proved the convergence of Q-learning using the ODE method [7].

In [8] we proposed a simulation-based policy iteration algorithm, in which the common recurrent state assumption is relaxed. In the proposed algorithm, the average cost is evaluated first and then realization factors [3] (the difference between the differential costs) from states to a reference state are evaluated, instead of the differential costs directly. In this way, the problem is also converted into a stochastic shortest path problem. Using the realization factors gives the flexibility of choosing the reference state not necessarily the same from iteration to iteration, which leads to remove the common recurrent state assumption. In addition, the average cost is embedded into the stage cost, where the new stage cost is the original stage cost minus the average cost, and temporal-difference learning scheme for stochastic shortest path problem in [2] is applied. Thus, the proposed algorithm should be more computationally efficient than the algorithm in [6]. To improve the performance further, transient states are selected as the initial states for sample paths, and the inverse of the visiting time is chosen as the stepsize.

Here we would like to give a brief summary of recent development of simulation-based algorithms for average cost MDP problems. To be clear, we introduce simulation-based policy iteration algorithms and simulation-based value iteration algorithms in separate sections.

2 Background

The basic problem we consider in this paper is the problem of optimizing a stochastic discrete-time dynamic system with average cost criterion. The dynamic system equation is

$$x_{t+1} = f_t(x_t, u_t, w_t), \quad t = 0, 1, \ldots, T-1, \tag{1}$$

where t indexes a time epoch; x_t is the state of the system; u_t is the action to be chosen at time t; w_t is a random disturbance which is characterized by a conditional probability distribution $P(\cdot \mid x_t, u_t)$; and T is the decision horizon. We denote the set of possible system states by S and the set of allowable actions in state $i \in S$ by $U(i)$. We assume S, $U(i)$, and $P(\cdot \mid x_t, u_t)$ do not vary with t. We further assume that the sets S and $U(i)$ are finite sets, where S consists of n states denoted by $0, 1, \ldots, n-1$.

If, at some time t, the system is in state $x_t = i$ and action $u_t = u$ is applied, we incur an expected cost $g(x_t, u_t) = g(i, u)$, and the system moves to state $x_{t+1} = j$ with probability $p_{ij}(u) = P(x_{t+1} = j \mid x_t = i, u_t = u)$. $p_{ij}(u)$ may be given a priori or may be calculated from the system equation and the known probability distribution of the random disturbance. $g(i, u)$ is assumed bounded. This framework containing states, actions, costs, probabilities

and the decision horizon is also known as a Markov Decision Process (MDP).

The objective is to minimize over all policies $\pi = \{\mu_0, \mu_1, \ldots\}$ with $\mu_t : S \to U, \mu_t(i) \in U(i)$ for i and t, the average cost per stage

$$J_\pi(i) = \lim_{T\to\infty} \frac{1}{T} E\left\{ \sum_{t=0}^{T-1} g(x_t, \mu_t(x_t)) \mid x_0 = i \right\}. \tag{2}$$

A stationary policy is an admissible policy of the form $\pi = \{\mu, \mu, \ldots\}$; we denote it by μ_∞.

In an average cost MDP, results and analyses depend on the chain structure corresponding to stationary policies. For simplicity, we consider only unichain MDPs in this paper. To our knowledge, there has been no research on simulation-based algorithms under the multichain assumption.

Under the unichain assumption the following hold for average cost MDPs [1] [6]:

- The average cost per stage associated with an initial state i and a stationary policy μ_∞, $J_\mu(i)$, and the optimal average cost from state i, $J^*(i)$, are independent of the initial state i. We denote these by η_μ and η^*, respectively.

- The optimal average cost η^* together with some vector $h^* = \{h^*(0), \ldots, h^*(n-1)\}$ satisfies the optimality equation

$$\eta^* + h^*(i) = \min_{u\in U(i)}[g(i,u) + \textstyle\sum_{j=0}^{n-1} p_{ij}(u) h^*(j)], \quad i = 0, \ldots, n-1. \tag{3}$$

  Furthermore, if $\mu(i)$ attains the minimum in the above equation for all states, the stationary policy μ_∞ is optimal. In addition, we may set $h^*(0) = 0$ to uniquely determine the vector h^*, which is also called the optimal differential cost.

- Given a stationary policy μ_∞ with corresponding average cost per stage η_μ, there is a unique vector $h_\mu = \{h_\mu(0), \ldots, h_\mu(n-1)\}$ such that $h_\mu(0) = 0$ and

$$\eta_\mu + h_\mu(i) = g(i, \mu(i)) + \textstyle\sum_{j=0}^{n-1} p_{ij}(\mu(i)) h_\mu(j), \quad i = 0, \ldots, n-1. \tag{4}$$

  h_μ is also called the differential cost associated with a stationary policy μ_∞.

One method to solve the optimality equation is policy iteration. Policy iteration consists of two steps: policy evaluation and policy improvement. At each iteration step k, a stationary policy $\mu_\infty^k = \{\mu^k, \mu^k, \ldots\}$ is given.

1. **Policy evaluation**: obtain the corresponding average and differential cost η^k and $h^k(i)$ satisfying (4).

2. **Policy improvement**: find a stationary policy μ^{k+1}, where for all i, $\mu^{k+1}(i)$ is such that

$$g(i, \mu^{k+1}(i)) + \sum_{j=0}^{n-1} p_{ij}(\mu^{k+1}(i)) h^k(j) = \min_{u\in U(i)} [g(i,u) + \sum_{j=0}^{n-1} p_{ij}(u) h^k(j)]. \tag{5}$$

If $\eta^{k+1} = \eta^k$ and $h^{k+1}(i) = h^k(i)$ for all i, the algorithm terminates; otherwise, the process is repeated with μ^{k+1} replacing μ^k.

Under the unichain assumption, the policy iteration algorithm terminates in a finite number of iterations with an optimal stationary policy. See [1] for multichain problems.

The other method of solving the optimality equation is value iteration. One version of value iteration for the average cost problem is simply to select arbitrarily a terminal cost function, say J_0, and to generate successively the corresponding optimal k-stage costs $J_k(i)$. However, this version of value iteration has some drawbacks [6]. An improved version, known as *relative value iteration*, takes form:

$$h^{k+1}(i) = \min_{u \in U(i)} [g(i,u) + \sum_{j=0}^{n-1} p_{ij}(u) h^k(j)] - \min_{u \in U(s)} [g(s,u) + \sum_{j=0}^{n-1} p_{sj}(u) h^k(j)]. \tag{6}$$

One variant [6] of this version can be implemented under the unichain and common recurrent state assumption.

Recently, Bertsekas [9] proposed a new value iteration method, under the unichain and common recurrent state assumption, by connecting the average cost problem to a stochastic shortest path problem. It has the form

$$\begin{array}{l} h^{k+1}(i) = \min_{u \in U(i)} [g(i,u) + \sum_{j=0}^{n-1} p_{ij} h^k(j)] - \eta^k, \quad i = 0, \ldots, n-1. \\ \eta^{k+1} = \eta^k + \alpha^k h^{k+1}(n-1) \end{array} \tag{7}$$

where γ^k is a positive stepsize. We call this version of value iteration *SSP (Stochastic Shortest Path) value iteration*. Computational tests indicate that SSP value iteration substantially outperforms the standard method for different problems [9].

3 Simulation-based Policy Iteration (SBPI)

SBPI originates from policy iteration, and the general structure of SBPI is the same as for exact policy iteration. There are two differences, however [2]:

- Given the current stationary policy μ_∞, the corresponding average cost and differential costs are not computed exactly. Instead, an approximate average cost $\hat{\eta}_\mu$ and approximate differential costs $\hat{h}_\mu(i)$ are obtained via simulation. Here, two sources of error are produced. First, the simulation length is finite. Second, noise from simulation experiments becomes a source of error.

- Once approximate policy evaluation is completed and $\hat{\eta}_\mu$ and $\hat{h}_\mu(i)$ are available, we generate a new policy $\bar{\mu}_\infty$ which is greedy with respect to $\hat{\eta}_\mu$ and $\hat{h}_\mu(i)$, i.e. satisfying (5). The greedy policy can be calculated exactly, or it can be approximated which introduces a new source of error.

In most simulation-based policy iteration algorithms for average cost problems so far (see [3], [4], [8] and [2]) the policy evaluation and policy improvement steps have been considered separately, with concentration on the ways to generate approximate average cost and differential costs for a fixed policy via simulation.

Next I would like to describe one of the above algorithms, an algorithm proposed by our group [8], in detail. Note that our proposed algorithm is realized under the unichain without common recurrent state assumption.

First, we discuss how to approximate the average cost associated with a stationary policy μ_∞ via simulation, which by definition can be written as

$$\eta_\mu = \lim_{T\to\infty} \frac{1}{T} E\left\{\sum_{t=0}^{T-1} g(x_t, \mu(x_t))\right\}. \tag{8}$$

Given a stationary policy, if we run L independent and identically distributed (i.i.d.) sample paths starting from an arbitrary state, each with length T, we can obtain an approximation of the average cost via simulation as T and L become large. We may also estimate η_μ iteratively.

Then let us discuss how to approximate the differential cost via simulation. Given a stationary policy μ_∞, and assuming the corresponding Markov chain is an aperiodic chain (can be relaxed), we also have [1]

$$h_\mu(i) = \lim_{T\to\infty} E\left\{\sum_{t=0}^{T-1} (g(x_t, \mu_t(x_t)) - \eta_\mu) \mid x_0 = i\right\}. \tag{9}$$

We may refer to the average cost (8) as the stationary cost, since it represents cost per stage for a system in steady state. Thus (9) allows interpretation of the differential cost as the expected total difference between the total cost and the stationary cost. The differential cost is also known as the bias [1] or the potential [3].

Assume the Markov chain associated with the current stationary policy, starting from state i, encounters state j at time epoch T_{ij} the first time, that is, $T_{ij} = \min\{t : t \geq 0, x_t = j\}$ (note that $T_{ii} = 0$, so that T_{ii} is not the recurrent time of state i); then

$$\begin{aligned}
h_\mu(i) &= \lim_{T\to\infty} E\left\{\sum_{t=0}^{T_{ij}-1} (g(x_t, \mu_t(x_t)) - \eta_\mu) + \sum_{t=T_{ij}}^{T-1} (g(x_t, \mu_t(x_t)) - \eta_\mu) \mid x_0 = i\right\} \\
&= E\left\{\sum_{t=0}^{T_{ij}-1} (g(x_t, \mu_t(x_t)) - \eta_\mu) \mid x_0 = i\right\} + \lim_{T\to\infty} E\left\{\sum_{t=T_{ij}}^{T-1} (g(x_t, \mu_t(x_t)) - \eta_\mu) \mid x_{T_{ij}} = j\right\} \\
&= b_\mu(i,j) + h_\mu(j),
\end{aligned}$$

where

$$b_\mu(i,j) \equiv E\left\{\sum_{t=0}^{T_{ij}-1} (g(x_t, \mu_t(x_t)) - \eta_\mu) \mid x_0 = i\right\}$$

are called realization factors [3]. This gives us an idea how to approximate $h_\mu(i)$ via simulation.

We know that the unichain Markov chain associated with a stationary policy contains a single recurrent class and a possibly empty set of transient states, so that each state in the recurrent class is reached in a finite number of steps from all initial states with a positive probability. If we choose one such state as reference state r, the differential costs of all other states can be expressed as

$$h_\mu(i) = b_\mu(i, r) + h_\mu(r). \tag{10}$$

If the reference state is simply state 0, the state having the differential cost zero, $h_\mu(i) = b_\mu(i, r)$ since we set $h_\mu(0) = 0$; if not, we can first approximate $b_\mu(0, r)$ via simulation and obtain $h_\mu(i)$ using $h_\mu(i) = b_\mu(i, r) - b_\mu(0, r)$, since $h_\mu(r) = -b_\mu(0, r)$.

So the task of approximating $h_\mu(i)$ reduces to approximating $b_\mu(i, r)$, where

$$b(i, r) \equiv E\left\{ \sum_{t=0}^{T_{ir}-1} (g(x_t, \mu_t(x_t)) - \hat{\eta}_\mu) \mid x_0 = i \right\}$$

for each state i. For brevity, we refer to $b_\mu(i, r)$ as $b_\mu(i)$.

Now the problem is converted into a stochastic shortest path problem, where the new stage cost is $g(x_t, \mu_t(x_t)) - \hat{\eta}_\mu$ and the new termination state is the reference state. Below, we apply the temporal-difference learning scheme for the stochastic shortest path problem [2] to our problem.

Consider a trajectory $(i_0, i_1, i_2, \ldots, i_N)$ and let m be an integer between 0 and N. We note that this trajectory contains the subtrajectory $(i_m, i_{m+1}, \ldots, i_N)$. At the end of a simulation run that generates the state trajectory $(i_0, i_1, i_2, \ldots, i_N)$, for each $m = 0, 1, \ldots, N-1$, use the formula

$$\hat{b}_\mu(i_m) := \hat{b}_\mu(i_m) + \gamma_{l_m}((g(i_m, i_{m+1}) - \eta_\mu) + \cdots + (g(i_{N-1}, i_N) - \eta_\mu) - \hat{b}_\mu(i_m)), \tag{11}$$

where γ_{l_m} is the stepsize.

Define *temporal differences* (TD) [2]:

$$d_m = (g(i_m, i_{m+1}) - \eta_\mu) + \hat{b}_\mu(i_{m+1}) - \hat{b}_\mu(i_m), \quad m = 0, 1, \ldots, N. \tag{12}$$

Then, following the state transition (i_m, i_{m+1}), the cost update formula can be written as follows:

$$\begin{cases} \hat{b}_\mu(i_1) := \hat{b}_\mu(i_1) + \gamma_{l_1} d_m \\ \hat{b}_\mu(i_2) := \hat{b}_\mu(i_2) + \gamma_{l_2} d_m \\ \cdots \\ \hat{b}_\mu(i_m) := \hat{b}_\mu(i_m) + \gamma_{l_m} d_m \end{cases} \tag{13}$$

Note that we approximate realization factors instead of differential costs using TD learning.

Now let us review other SBPI algorithms under the stronger assumptions. Under the recurrence assumption, Tsitsiklis and Van Roy [4] extend TD learning for the discounted

cost case to the average cost case. A linear function approximation of the following form is used to represent the differential costs:

$$h(i,r) = \sum_{k=1}^{K} r(k)\phi_k(i).$$

If a sequence of states $(i_0, i_1, i_2, \ldots, i_m, \ldots,)$ via simulation is observed, the average cost estimate is updated according to

$$\eta_{m+1} = (1-\alpha_m)\eta_m + \alpha_m g(i_m, i_{m+1}), \tag{14}$$

where α_m is a sequence of scalar step sizes. Note that the average cost is estimated iteratively here. Concurrently, the parameter for the differential cost evolves according to a more complex iteration:

$$r_{m+1} = r_m + \gamma_m d_m \sum_{k=0}^{m} \lambda^{m-k}\phi(i_k), \tag{15}$$

where the temporal difference d_m corresponding to the transition from i_m to i_{m+1} by

$$d_m = (g(i_m, i_{m+1}) - \eta_m) + h(i_{m+1}, r_m) - h(i_m, r_m). \tag{16}$$

Further, convergence is established for the case where approximations are generated by linear combinations of basis functions over a finite state space. If an average cost problem is proved to be recurrent, this TD learning method is a good choice.

In [3], Cao presented two single-path algorithms, in which the differential costs or realization factors are calculated as accumulated costs of a sample path divided by the visiting time. There is no learning involved.

In the above algorithms mentioned, the policy improvement step follows that of exact policy iteration, i.e., using min/max computation. The necessary assumption for that is that the probabilities of the system are readily available in explicit form, which is not the case in many circumstance. Besides, in the above algorithms, each policy improvement step will not be implemented until the corresponding evaluated cost function converges. This implies the algorithms would be slow if it is desired to guarantee the accuracy of evaluated cost function. A proposed algorithm, called optimistic policy iteration [2], in which the policy improvement step is implemented before the convergence of the evaluated cost function, has been successful in some experimental tests but has no theoretical support.

Recently, Konda and Borkar [5] proposed several actor-critic type simulation-based policy iteration algorithms and tried to solve the afore-mentioned difficulties. The original idea of actor-critic emerged from the machine learning community. In actor-critic methods there is a separate memory structure which is independent of the value function that explicitly represents the policy. The policy structure is known as the *actor* or *action network*, since it is used to select actions, and the estimated value function is known as the *critic* or *critic network*, because it criticizes the actions made by the actor [10].

In [5], the need to have the policy evaluation recursion converge before implementing policy improvement is circumvented by two-time scale stochastic approximation. The outer

loop operates on a slower scale and thus see the inner loop as essentially equilibrated, while the inner loop sees the outer one as quasi-static.

For average cost problem, one version has the following form:

$$\begin{array}{rl} h^{k+1}(i) = & (1-\gamma(k))h^k(i) + \gamma(k)[g(i,\mu^k(i)) + h^k(\xi^k(i,\mu^k(i))) - \eta] \\ \pi^{k+1}(i) = & P(\ \pi^k(i) + \alpha(i,k)\ \delta(\pi^k(i), h^k(\cdot), g(i,\cdot))\ ). \end{array} \tag{17}$$

We conjecture that the expression for $h^{k+1}(i)$ originates from an incremental version of (4) which is:

$$h^{k+1}(i) := (1-\gamma(k))h^k(i) + \gamma(k)[g(i,\mu^k(i)) + \sum_{j=0}^{n-1} p_{ij}(\mu^k(i))h^k(j) - \eta] \tag{18}$$

Note that one difference is that the summation involving $p_{ij}(\mu^k(i))$ is replaced by a simulated transition according to that probability.

In the expression for $\pi^{k+1}(i)$, $\pi^{k+1}(i)$ is a stationary randomized policy updated with $\delta(\cdot)$ (critic) which contains information of the differential cost and randomized policy for the previous stage. P is a projection to guarantee $\pi^{k+1}(i)$ to be a randomized policy too.

The advantages of these actor-critic algorithms are: 1) Just as we mentioned, it is not necessary to know system probabilities. 2) These algorithms are more efficient since policy improvement need not wait for the policy evaluation step to converge. 3) convergence is guaranteed.

One disadvantage of these algorithms is that randomized policies are used, but one usually desires to implement nonrandomized policies.

4 Simulation-Based Value Iteration

If there is no explicit model of the system and the cost structure, Q-learning, another version of simulation-based value iteration, proves to be an effective method. It updates directly estimates of the Q-factors associated with an optimal policy, thereby avoiding the multiple policy evaluation steps of the policy iteration. Using Q-factor, the optimality equation is

$$\eta^* + Q^*(i,u) = g(i,u) + \textstyle\sum_{j=0}^{n-1} p_{ij}(u) \min_{u'} Q(j,u'), \quad i = 0,\ldots,n-1. \tag{19}$$

The first simulation-based value iteration algorithm for average cost problems was R-learning proposed by Schwartz [11], which is similar to Q-learning. In this algorithm, R-factor, values of state-action pairs like Q-factor and average cost are updated concurrently using the immediate reward along with an adjustment factor. The idea is to obtain a good estimate for the average cost while searching for the optimal policy using a value-iteration type update. There is no convergence analysis for this algorithm.

Singh [12] presented two Q-learning methods for average cost problems: one is based on R-learning by Schwartz; the other updates the estimate of the average cost in a fashion similar

to Jalali and Ferguson's deterministic asynchronous algorithm for average cost problem [7]. There are experimental results but no convergence analysis.

In fact, the update formula is the same for all of the three algorithms:

$$Q^{k+1}(i,u) = (1-\gamma(k))Q^k(i,u) + \gamma(k)(g(i,u,j) + \min_{u'} Q^k(j,u') - \eta^k). \tag{20}$$

Note that this formula comes from the incremental version of the optimal equation using Q-factor (19) with the item having probability replaced by simulation transition.

The difference lies in how to estimate the average cost: in R-learning, the average cost is updated only when the greedy action is executed, and in Singh's first algorithm the average cost is updated with every action. Singh argued that his algorithm is more efficient since R-learning seems to waste information whenever a non-greedy action is taken, which is quite often, especially in the beginning when the agent is exploring heavily [12]. In Singh's second algorithm, the average cost is estimated as the sample average of the costs received for greedy actions. Note that there are few attention on the assumption, different criteria, and convergence in these algorithms.

Based on above algorithms, Mahadevan [13] gave a nice summary of average cost problems and pointed out the need to consider the average cost criterion, especially the difference between gain-optimal and bias-optimal policies. He also gave many experimental results and suggested directions for future research.

Later, a simulation-based value iteration algorithm based on relative value iteration is given under the unichain plus common recurrent state assumption [2]:

$$Q^{k+1}(i,u) = (1-\gamma(k))Q^k(i,u) + \gamma(k)(g(i,u,j) + \min_{u' \in U(j)} Q^k(j,u') - \min_{u' \in U(s)} Q^k(s,u')). \tag{21}$$

However, there is no convergence analysis either.

Recently, Abounadi, Bertsekas and Borkar [7] proposed and gave for the first time a complete convergence analysis of two Q-learning algorithms for average cost problem. The first one is based on the relative value iteration and its synchronous version has the form:

$$Q^{k+1}(i,u) = (1-\gamma(k))Q^k(i,u) + \gamma(k)(g(i,u,\xi_{iu}^k) + \min_{u'} Q^k(\xi_{iu}^k,u') - f(Q^k)) \tag{22}$$

The differences between the above two formulas, both of which are based on relative value iteration, are that "j" in (21) is replaced by "ξ_{iu}^k" in (22) to emphasize the idea of replacing the conditional average with respect to the transition probabilities $p(i,u,\cdot)$ by an actual evaluation at a random variable ξ_{iu} with law $p(i,u,\cdot)$ and then "see" the conditional average by means of the averaging effect of the stochastic approximation algorithm and that $\min_{u' \in U(s)} Q^k(s,u')$ is replaced by a more general function f. In addition, there exists an asynchronous version.

The second algorithm is based on the SSP value iteration and has the form:

$$\begin{aligned} Q^{k+1}(i,u) = \;& (1-\gamma(k))Q^k(i,u) + \gamma(k)(g(i,u,\xi_{iu}^k) \\ & + \min_{u'} Q^k(\xi_{iu}^k,u')I\{\xi_{iu}^k \neq s\} - \eta^k), \\ \eta^{k+1} = \;& \eta^k + \alpha(k)\min_{u'} Q^k(s,u'). \end{aligned} \tag{23}$$

In [7] some important directions are also pointed out: 1) The algorithms need to be interlaced with some approximation architectures and analyzed, since the state space can be very large. 2) An analysis of rate of convergence and good speed-up procedures is needed. 3) Extension to the case where the state space is not finite is an open issue.

5 Discussion

So far, we described some simulation-based algorithms for average cost MDP problems. But what next?

There are many simulation-based algorithms for discounted cost problems or SSP problems, so one idea is to use these algorithms to solve average cost problems directly. The difficulty is when to use it. Under the recurrence assumption, Tsitsiklis and Van Roy [14] provide an analytical comparison between discounted and average cost TD learning with linearly parameterized approximations. It is shown that as the discounted factor approaches 1, the value function produced by discounted TD approaches the differential cost generated by average cost TD. This connection suggests that it is possible to use simulation-based policy iteration for discounted cost problems under the recurrence assumption. Besides, SSP value iteration under unichain and common recurrent state assumption suggests that some simulation-based value iteration algorithms for SSP problems may be used for average cost problems.

However, average cost problems have their own characteristics, so not all the algorithms can be extended. Now there is still little research on simulation-based algorithms for unichain (without common recurrent state) or multichain average cost problems (except [8]). And most simulation-based algorithms can only attain gain-optimality, not bias-optimality.

Another important direction is the implementation of the algorithms on more applications.

Acknowledgement:

This work was supported in part by the National Science Foundation under Grant DMI-9713720, in part by the Semiconductor Research Corporation under Grant 97-FJ-491, and in part by a fellowship from General Electric Corporate Research and Development through the Institute for Systems Research.

References

[1] Martin L. Puterman, *Markov Decision Processes*, John Wiley & Sons, Inc., New York, 1994.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, Massachusetts, 1996.

[3] Xi-Ren Cao, "Single sample path based optimization of Markov chains," preprint.

[4] John N. Tsitsiklis and Van Roy, "Average cost temporal-difference learning," to appear *Automatica*.

[5] Vijay Rao Konda and Vivek S. Borkar, "Learning algorithms for Markov decision processes," preprint.

[6] Dimitri P. Bertsekas, *Dynamic Programming and Optimal Control Vol 1 & 2*, Athena Scientific, Belmont, Massachusetts, 1995.

[7] Abounadi J., D. Bertsekas, and Vivek S. Borkar, "Learning algorithms for Markov decision processes with average cost," Tech. Rep., MIT, 1998, LIDS-P-2434.

[8] Ying He, Michael C. Fu, and Steven Marcus, "A simulation-based policy iteration algorithm for average cost unichain Markov decision processes," submitted to *7th INFORMS Computing Society Conference on OR Computing Tools for the New Millennium*.

[9] Dimitri P. Bertsekas, "A new value iteration method for the average cost dynamic programming problem," *SIAM J. Control Optim.*, vol. 36, no. 2, pp. 742–759, 1998.

[10] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning*, The MIT Press, Cambridge, Massachusetts, 1998.

[11] A. Schwartz, "A reinforcement learning method for maximizing undiscounted rewards," in *Proc. of the 10th Intl. Conf. on Machine Learing*, Morgan Kaufmann, San Mateo, 1993, pp. 298–305.

[12] Satinder P. Singh, "Reinforcement learning algorithms for average-payoff Markovian decision processes," in *Proc. of the 12th National Conf. on Artificial Intelligence*, 1994, pp. 700–705.

[13] S. Mahadevan, "Average reward reinforcement learning: Foundations, algorithms and empirical results," *Machine Learning*, vol. 22, pp. 1–38, 1996.

[14] John N. Tsitsiklis and Van Roy, "On average versus discounted reward temporal-difference learning," submitted to *Machine Learning*, 1999.