Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2011) 000–6

Procedia Computer Science

Complex Adaptive Systems, Volume 1
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2011- Chicago, IL

Approximate Policy Iteration for Semi-Markov Control Revisited

Abhijit Gosavi [*]

219 Engineering Management, Missouri University of Science and Technology, Rolla, MO 65409, USA

Abstract

The semi-Markov decision process can be solved via reinforcement learning without generating its transition model. We briefly review the existing algorithms based on approximate policy iteration (API) for solving this problem for discounted and average reward under the infinite horizon. API techniques have attracted significant interest in the literature recently. We first present and analyze an extension of an existing API algorithm for discounted reward that can handle continuous reward rates. Then, we also consider its average reward counterpart, which requires an updating based on the stochastic shortest path (SSP). We study the convergence properties of the algorithm that does not require the SSP update.

Keywords: approximate policy iteration, reinforcement learning, average reward, Semi-Markov

1. Introduction

Markov decision problems (MDP) are problems of sequential decision-making. In sequential decision making, typically, one considers a system with multiple states where the system can transition randomly from one state to another. The decision-maker's goal is to select the best action in each state in order to optimize some function of the rewards earned in each state. The rewards could be earned over an infinite time horizon or a finite time horizon. We will be interested in the infinite time horizon in this paper. The rewards tend to be functions of the action chosen, as well as the state in which the action was chosen and the state to which the system transitioned as a result of the action.

A subset of these sequential decision-making problems called MDPs are characterized by Markov chains in which the state transitions obey the "Markov property." In this paper, we will study a more general version of the MDP called the semi-Markov decision problem (SMDP), which is characterized by what is known as the semi-Markov property.

[*]Tel: 573-341-4624; Email: gosavia@mst.edu

In an MDP, the time spent by the system in any given transition is irrelevant, whereas in an SMDP, the time spent is a part of the model. We will elaborate on this later.

MDPs and SMDPs can be solved via dynamic programming (DP) methods [1]. DP suffers from the curse of modeling and dimensionality, due to difficulties in estimating the transition model of the problem and in being able to store the model. Reinforcement learning, abbreviated as RL [2, 3, 4, 5], is a simulation-based version of DP that can break these curses. Many algorithms in RL are based on equations from DP. The framework of DP has provided us with two algorithms: value iteration and policy iteration. The famous Q-Learning algorithm [6] is based on value iteration. In this paper, we will study algorithms based on policy iteration. In RL, such algorithms are loosely termed as approximate policy iteration (API) techniques. Policy iteration has two steps: policy evaluation and policy improvement. In policy evaluation, a fixed policy is evaluated, and in policy improvement a new policy is selected on the basis of calculations performed in the policy evaluation step. The policy evaluation step in RL may require many iterations, while in DP, it is based on solving a linear equation called the Poisson equation, also known as the Bellman equation for a fixed policy.

The field of API has grown significantly in the last few years, although much work in the early years of RL was based on value iteration. The modified Q-Learning algorithm of [7], also known as SARSA, is based on the idea of performing policy iteration. In the ideal version of API [2], one performs a large number of iterations in the policy evaluation step (i.e., $k_{\max} >> 1$). We will use multiple iterations but will be interested in the SMDP. An important point of difference between SARSA and the ideal version of API is that SARSA uses Q-factors, while API is generally described in terms of the value (or cost-to-go) function of DP, i.e., J. The algorithm on which we focus our attention here, i.e., Q-P-Learning [8, 4], is based on Q-factors and was designed for solving SMDPs.

In this context, it is important to point out that the so-called actor-critic algorithm [9, 10, 11] is also related to API. However, the updating equation it uses is a slight variant of the Poisson equation. It performs only one update of the policy evaluation step (i.e., $k_{\max} = 1$) before attempting to improve the policy. Other algorithms that are related to API are least-squares policy iteration [12] and asynchronous distributed policy iteration [13].

We analyze the Q-P-Learning algorithm for average and discounted reward SMDPs. For the discounted problem, we assume that some of the reward is earned as a lump sum at the start of the state transition, while some may be earned over the transition. For the average reward SMDP, Q-P-Learning [8] assumes an update based on the stochastic shortest path problem (SSP). The version we analyze here does not require the SSP.

The rest of this article is organized as follows. Section 2 presents some of the notation and mathematical background. Section 3 discusses the discounted reward case while Section 4 discusses the average reward case. Section 5 presents some concluding remarks.

2. Background

We begin with some notation that we will need, then present the underlying equations, and finally present the structure of the API algorithm we are interested in.

2.1. Notation

Let $\mathcal{S}$ denote the finite set of states, $\mathcal{A}(i)$ the finite set of actions permitted in state i, and $\mu(i)$ the action chosen in state i when a deterministic, stationary policy μ is pursued, where $\cup_{i\in\mathcal{S}}\mathcal{A}(i) = \mathcal{A}$. Further let $r(.,.,.) : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \Re$ denote the one-step immediate reward and $p(.,.,.) : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$ denote the associated transition probability. In SMDPs, the discount factor is given as $\exp(-\bar{\gamma}\tau)$ where $\bar{\gamma}$ is the rate of discounting and τ is the duration of time period over which one discounts (see [1]). The rate of discounting is oftentimes equal to the rate of inflation or the interest, depending on the application. Also the time spent in each state is defined as $t(.,.,.) : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \Re$ and denotes the time of one transition. In the infinite horizon MDP, the performance metric, i.e., the average or discounted reward, does not depend on the amount of time spent in the state transitions, where as in the SMDPs, it is a function of this time. Hence, in the SMDP, one must take the time into account in the updating transformations of the algorithm.

2.2. Poisson equations

We now present the underlying Poisson equation (or the Bellman equation for a fixed policy) that we will seek to solve in the policy evaluation step.

2.2.1. Discounted reward

Here, it is necessary to distinguish between reward earned immediately after the transition starts, also called lump sum reward, and the reward that is earned continuously over the transition. We thus need to qualify our notation for immediate reward in this context. Let $r_C(i,a,j)$ denote the continuous rate of reward (e.g., in dollars per hour) from state i to j under action a, and $r_L(i,a,j)$ denotes the immediate reward (e.g., in dollars) earned from i *immediately* after action a is taken and the system goes to j (lump sum reward). The Poisson equation is:

$$J_\mu(i) = \sum_{j\in S} p(i,\mu(i),j) r_L(i,\mu(i),j) + R(i,\mu(i)) + \sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t) f_{i,\mu(i),j}(t) J_\mu(j)dt\right], \quad (1)$$

where $f_{i,a,j}(t)$ denotes the pdf of the transition time from i to j under a,

$$R(i,a) = \sum_{j\in S}\left[\int_0^\infty r_C(i,a,j)\frac{1-\exp(-\bar{\gamma}t)}{\bar{\gamma}} f_{i,a,j}(t)dt\right], \text{ and } \lim_{t\to\infty} f_{i,a,j}(t) = p(i,a,j).$$

In Equation (1), the unknown is the value function J_μ. Now, we define a Q-factor as follows:

$$Q_\mu(i,a) \equiv \sum_{j\in S} p(i,a,j) r_L(i,a,j) + R(i,a) + \sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t) f_{i,a,j}(t) J_\mu(j)dt\right].$$

From the above and (1), one has that for all i, $J_\mu(i) = Q_\mu(i,\mu(i))$. Then, using this in the above, we obtain the Q-version of the Poisson equation:

$$Q_\mu(i,a) = \sum_{j\in S} p(i,a,j) r_L(i,a,j) + R(i,a) + \sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t) f_{i,a,j}(t) Q_\mu(j,\mu(j))dt\right]. \quad (2)$$

2.2.2. Average reward

In average reward, the continuously earned reward and the immediate reward earned at the start or any time during the transition can be lumped together into the term $r(i,a,j)$, since there is no discounting. The Poisson equation in terms of Q-factors can be derived as shown in [8] to be:

$$Q_\mu(i,a) = \sum_{j\in S} p(i,a,j)\left[r(i,a,j) - \rho_\mu t(i,a,j) + Q_\mu(j,\mu(j))\right], \text{ where } \rho_\mu \text{ denotes the SMDP's average reward.} \quad (3)$$

2.3. Algorithm format

We now present the format of the API algorithm based on Q-factors (also called Q-P-Learning in [8]).

Step 1 Initialize $P(i,a)$ for all (i,a) pairs to random values, e.g., 0. Set E, the number of policy evaluations, to 0. Set $E_{\max}$ and $k_{\max}$ to large values.

Step 2 Set $Q(i,a) = 0$ for all (i,a) pairs. Set k, the number of iterations within an evaluation, to 0.

Step 3 (**Policy evaluation**): Start fresh simulation.

Step 3a Let the current state be i. Simulate action $a \in \mathcal{A}(i)$ with a probability of $1/|\mathcal{A}(i)|$.

Step 3b Let the next state in the simulator be j. Update $Q(i,a)$ using the relevant updating equation.

Step 3c Increment k by 1. If $k < k_{\max}$, set current state i to value of j and go to Step 3a. Otherwise, go to Step 4.

Step 4 (**Policy improvement**): Set $P(i,a) \leftarrow Q(i,a)$ for all (i,a) pairs and increment E by 1. If E equals $E_{\max}$, STOP; otherwise, return to Step 2.

3. Discounted Reward SMDPs

We now present an algorithm using API and Q-factors for solving discounted reward SMDPs. As mentioned above, the algorithm is a variant of that presented in [4]; it contains the continuous reward rate in addition to the lump sum reward. Consider the Poisson equation for discounted reward SMDPs, i.e., Equation (2). Using a step-size-based update that employs a sample rather than the expectation, the equation suggests the following algorithm:

$$Q(i,a) \leftarrow (1-\alpha)Q(i,a) + \alpha\left[r_L(i,a,j) + r_C(i,a,j) \times \frac{1-\exp(-\bar{\gamma}t(i,a,j))}{\bar{\gamma}} + \exp(-\bar{\gamma}t(i,a,j))Q\left(j, \arg\max_{b\in\mathcal{A}(j)} P(j,b)\right)\right], \tag{4}$$

where α is the usual step size of RL [2]. The above equation should be used as the relevant updating equation in Step 3b of section 2.3. $\vec{Q}$ will denote a vector of Q-factors whose mth element is $Q(i,a)$, where $m = (i,a)$. We now propose to prove the analyze the convergence properties of this algorithm via the following result.

Theorem 1. *The algorithm proposed in Equation (4) converges to the optimal solution of the discounted reward SMDP with probability 1.*

Proof We will present the main arguments in the proof here. Most RL algorithms can be analyzed for convergence by showing that they belong to the following model:

$$Q(i,a) \leftarrow (1-\alpha)Q(i,a) + \alpha\left[F(\vec{Q})(i,a) + w(i,a)\right] \tag{5}$$

where $F(.)$ denotes a transformation on $\vec{Q}$, which is the vector of the Q-factors, and $w(i,a)$ is a noise term. Here $x(i,a)$ denotes the (i,a)-th component of vector $\vec{x}$, e.g., $F(\vec{Q})$ and $\vec{w}$. We define the transformation F and the vector $\vec{w}$ as follows:

$$F(\vec{Q})(i,a) = \sum_{j\in S} p(i,a,j)r_L(i,a,j) + R(i,a) + \sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t)f_{i,a,j}(t)Q(j,\mu(j))dt\right] \tag{6}$$

where $\mu(j) = \arg\max_{b\in\mathcal{A}(j)} P(j,b)$, and

$$w(i,a) = r_L(i,a,j) + r_C(i,a,j) \times \frac{1-\exp(-\bar{\gamma}t(i,a,j))}{\bar{\gamma}} + \exp(-\bar{\gamma}t(i,a,j))Q\left(j, \arg\max_{b\in\mathcal{A}(j)} P(j,b)\right) - F(\vec{Q})(i,a).$$

Then, it can be shown, as is standard in the literature, that the vector $\vec{w}$ is a martingale (conditional zero mean) and hence its effect vanishes in the limit. We will now show that F is a contractive transformation. From the definition of F in Equation (6), we can write that for two different values of the Q-factor for (i,a), the following is true:

$$F(Q^1)(i,a) - F(Q^2)(i,a) = \sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t)f_{i,a,j}(t)\left[Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right]dt\right], \text{ which implies that}$$

$$\begin{aligned}
\left|F(Q^1)(i,a) - F(Q^2)(i,a)\right| &= \left|\sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t)f_{i,a,j}(t)\left[Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right]dt\right]\right| \\
&\leq \left|Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right| \sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t)f_{i,a,j}(t)dt\right] \\
&\leq \max_{j,\mu(j)}\left|Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right| \sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t)f_{i,a,j}(t)dt\right].
\end{aligned}$$

Now, since there exists a $\lambda \in \Re$ where $0 < \lambda < 1$ such that $\sum_{j\in S}\left[\int_0^\infty \exp(-\bar{\gamma}t)f_{i,a,j}(t)dt\right] \leq \lambda \sum_{j\in S}\left[\int_0^\infty f_{i,a,j}(t)dt\right] = \lambda$, we can write the above as:

$$\begin{aligned}
\left|F(Q^1)(i,a) - F(Q^2)(i,a)\right| &\leq \max_{j,\mu(j)}\left|Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right|\lambda \\
&= \lambda\|\vec{Q}^1 - \vec{Q}^2\|_\infty.
\end{aligned}$$

Since the above is true for all values of (i, a), it is also true for the value of (i, a) that maximizes the left hand side of the above, and hence, we have that:

$$||F(\vec{Q}^1) - F(\vec{Q}^2)||_\infty = \lambda||\vec{Q}^1 - \vec{Q}^2||_\infty$$

Thus F is contractive, and so it must have a unique fixed point. The existence of the unique fixed point implies from ordinary differential equation (ODE) convergence theory that a so-called asymptotically stable equilibrium exists for an associated ODE (see [14]), which implies from Theorem 2.1 (i) of [15] that the Q-factors remain bounded. Then Theorem 3.1(b) in [16] ensures that the asynchronous update in Equation (5) converges with probability 1. In other words, $Q(i, a)$ on both sides of (5) become equal; since the noise terms vanish in the limit, we have that:

$$Q(i, a) = (1 - \alpha)Q(i, a) + \alpha \left[F(\vec{Q})(i, a)\right], \text{ i.e., } Q(i, a) = F(\vec{Q})(i, a),$$

which is the desired Poisson equation (2), and we have convergence to the desired solution. □

4. Average reward SMDPs

For the average reward case, one must estimate the value of the average reward *before* the policy evaluation phase. This is because the associated Poisson equation (3) has the term ρ_μ in it. Note that the policy being evaluated is contained in the P-factors, i.e., $\mu(i) = \arg\max_{b\in\mathcal{A}(i)} P(i, b)$. Hence, we modify Step 2 as follows:

[Step 2] Set $Q(i, a) = 0$ for all (i, a) pairs. Set k, the number of iterations within an evaluation, to 0. Simulate the system for a time interval of $T_{\max}$ using a **fixed** policy μ where $\mu(i) = \arg\max_{b\in\mathcal{A}(i)} P(i, b)$. In case of ties in actions from the argmax operation, break tie randomly. After the simulation, divide the total of immediate rewards earned during the simulation by $T_{\max}$ to obtain an estimate of ρ_μ. Use multiple replications if necessary.

Also, the updating equation in Step 3b depends on whether or not we use the SSP update. The description in [8] does not show the SSP update in the algorithm description but uses it for showing convergence. The SSP update requires that all states be recurrent. We first present the updating equation that uses the SSP update.

Step 3b with SSP update: This requires that a state be selected in Step 2 as the distinguished state, which will be called i^*.

$$Q(i, a) \leftarrow (1 - \alpha)Q(i, a) + \alpha \left[r(i, a, j) - \rho_\mu t(i, a, j)) + Q\left(j, \arg\max_{b\in\mathcal{A}(j)} P(j, b)\right)\right], \tag{7}$$

where in the right hand side

$$Q\left(j, \arg\max_{b\in\mathcal{A}(j)} P(j, b)\right) \leftarrow 0 \text{ if } j = i^*. \tag{8}$$

Note that in the above, $Q(i^*, a)$ is *not* fixed to 0 for any a. Only when its value has to be used in the updating equation (7), a value of 0 is used for it. Setting a value to 0 when it is used in the updating equation is called SSP grounding.

Step 3b without SSP update:

$$Q(i, a) \leftarrow (1 - \alpha)Q(i, a) + \alpha \left[r(i, a, j) - \rho_\mu t(i, a, j)) + Q\left(j, \arg\max_{b\in\mathcal{A}(j)} P(j, b)\right)\right]. \tag{9}$$

Note that the updating equation is identical to that in the SSP case except that we do not use (8), i.e., SSP grounding. A mathematical advantage of SSP grounding is that the transformation underlying the SSP-grounded update is contractive. Without SSP grounding the update is non-expansive at best. In practice, it has been found [4] that the non-SSP algorithm does produce optimal solutions. Hence, we now analyze its convergence properties here.

Theorem 2. *The algorithm proposed in Equation (9) converges to the optimal solution of the average reward SMDP with probability 1 as long as the Q-factors remain bounded.*

Note that we *assume* the the iterates (Q-factors) remain bounded due to the lack of existing results to show boundedness (non-expansive mapping with non-unique fixed point). Numerical evidence exists to support this [4].

Proof Following the line of analysis in the previous result, we first define the transformation F as:

$$F(\vec{Q})(i,a) = \sum_{j\in\mathcal{S}} p(i,a,j)\left[r(i,a,j) - \rho_\mu t(i,a,j) + Q(j,\mu(j))\right] \tag{10}$$

where $\mu(j) = \arg\max_{b\in\mathcal{A}(j)} P(j,b)$. Note that $F(\vec{Q}) = \vec{Q}$ is the Q-version of the Poisson equation which has infinitely many solutions [1]. Hence the set of fixed points of F is non-empty. Using the definition of F, we can write:

$$F(Q^1)(i,a) - F(Q^2)(i,a) = \sum_{j\in\mathcal{S}} p(i,a,j)\left[Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right], \text{ which implies that}$$

$$\begin{aligned}
\left|F(Q^1)(i,a) - F(Q^2)(i,a)\right| &= \left|\sum_{j\in\mathcal{S}} p(i,a,j)\left[Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right]\right| \\
&\leq \left|Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right| \sum_{j\in\mathcal{S}} p(i,a,j) \\
&\leq \max_{j,\mu(j)} \left|Q^1(j,\mu(j)) - Q^2(j,\mu(j))\right| \sum_{j\in\mathcal{S}} p(i,a,j) = \|\vec{Q}^1 - \vec{Q}^2\|_\infty \cdot 1.
\end{aligned}$$

Since the above is true for any value of (i,a), it must hold for the value that maximizes the left hand side, and hence: $\|F(\vec{Q}^1) - F(\vec{Q}^2)\|_\infty \leq \|\vec{Q}^1 - \vec{Q}^2\|_\infty$. Thus, F is non-expansive. Since its set of fixed points is non-empty and since the Q-factors remain bounded, Theorem 3.1(b) in [16] ensures convergence to some solution of $F(\vec{Q}) = \vec{Q}$. Any solution is sufficient for policy evaluation. □

5. Conclusions

The field of API has attracted a great deal of research interest recently. In comparison to schemes based on value iteration, API can take significant amounts of time on the computer if $k_{\max}$ is set to a large number (ideal version). On the other hand, small values of $k_{\max}$ can lead to chattering (oscillatory instability), i.e., going to a worse policy upon policy improvement. But, the literature focusses on API for MDPs. This paper attempted to analyze the SMDP. We used the ideal version because of its robustness. The existing algorithm for the discounted case [4] cannot handle continuous reward rates while its average reward counterpart requires the SSP update [8]. Computational experience, however, suggests that the average reward algorithm that bypasses the SSP update may be stable [4]. We presented an analysis of the discounted reward algorithm that accounts for continuous reward rates and an average reward algorithm that bypasses the SSP update. Future research should could potentially pursue the following directions:

1. Engineering SMDP applications for the discounted reward algorithm
2. A proof for boundedness of iterates in the non-SSP average reward algorithm for SMDPs

References

References

[1] D. P. Bertsekas, Dynamic Programming and Optimal Control, Athena Scientific, Belmont, Massachusetts, 1995.
[2] D. Bertsekas, J. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, MA, 1996.
[3] R. Sutton, A. G. Barto, Reinforcement Learning, The MIT Press, Cambridge, Massachusetts, 1998.
[4] A. Gosavi, Simulation-Based Optimization:Parametric Optimization Techniques and Reinforcement Learning, Kluwer Academic Publishers, Boston, MA, 2003.
[5] W. Powell, Approximate Dynamic Programming: Solving the curses of dimensionality, Wiley-Interscience, NJ, USA, 2007.
[6] C. Watkins, Learning from delayed rewards, Ph.D. thesis, Kings College, Cambridge, England (May 1989).

[7] G. Rummery, M. Niranjan, On-line Q-learning using connectionist systems, technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University (1994).

[8] A. Gosavi, A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis, Machine Learning 55(1) (2004) 5–29.

[9] A. Barto, R. Sutton, C. Anderson, Neuronlike elements that can solve difficult learning control problems, IEEE Transactions on Systems, Man, and Cybernetics 13 (1983) 835–846.

[10] P. J. Werbös, Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research, IEEE Transactions on Systems, Man., and Cybernetics 17 (1987) 7–20.

[11] V. Konda, V. S. Borkar, Actor-critic type learning algorithms for Markov decision processes, SIAM Journal on Control and Optimization 38(1) (1999) 94–123.

[12] M. Lagoudakis, R. Parr, Least-squares policy iteration, Journal of Machine Learning Research 4 (2003) 1107–1149.

[13] D. Bertsekas, H. Yu, Distributed asynchronous policy iteration in dynamic programming, in: Conference Proceedings of the 48th Allerton Conference, IEEE, 2010.

[14] V. S. Borkar, K. Soumyanath, An analog scheme for fixed point computation, Part i:theory, IEEE Transactions Circuits and Systems I. Fundamental Theory and Appl. 44 (1997) 351–354.

[15] V. S. Borkar, S. Meyn, The ODE method for convergence of stochastic approximation and reinforcement learning, SIAM Journal of Control and Optimization 38 (2) (2000) 447–469.

[16] V. S. Borkar, Asynchronous stochastic approximation, SIAM J. Control Optim. 36 No 3 (1998) 840–851.