Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

ON STEP SIZES, STOCHASTIC SHORTEST PATHS, AND SURVIVAL PROBABILITIES IN REINFORCEMENT LEARNING

Abhijit Gosavi

Department of Engineering Management & Systems Engineering
Missouri University of Science and Technology
Rolla, MO 65401, U.S.A.

ABSTRACT

Reinforcement Learning (RL) is a simulation-based technique useful in solving Markov decision processes if their transition probabilities are not easily obtainable or if the problems have a very large number of states. We present an empirical study of (i) the effect of step-sizes (learning rules) in the convergence of RL algorithms, (ii) stochastic shortest paths in solving average reward problems via RL, and (iii) the notion of survival probabilities (downside risk) in RL. We also study the impact of step sizes when function approximation is combined with RL. Our experiments yield some interesting insights that will be useful in practice when RL algorithms are implemented within simulators.

1 INTRODUCTION

Reinforcement Learning (RL) is a simulation-based technique that is useful on large-scale and complex Markov decision processes (MDPs) (Sutton and Barto 1998). In this paper, we will address (i) the role of step sizes (learning rules) in discounted-reward problems and (ii) that of the grounding mechanism of the shortest stochastic path (SSP) in average-reward problems and (iii) the notion of introducing survival probability (downside risk) within RL. We will study the impact of these factors on the values of iterates and examine by how much values can diverge from the values obtained from dynamic programming. In the context of step sizes, we will perform a study using some standard rules to determine how they perform. For average reward problems, the SSP-grounding mechanism allows us to compare the values of the iterates to those obtained from a comparable value iteration algorithm; here, however, multiple step-sizes are needed and an empirical study needs to take that into account. We also develop and test an RL algorithm that models the survival-probability of a system. Typically, the survival probability is defined with respect to a known target revenue. The survival probability of a system is the probability that the revenue in unit time will exceed the target. It is directly related to the downside risk in operations research and the exceedance probability in the insurance industry (Grossi and Kunreuther 2005). We present a Bellman equation for survival probabilities, and then numerically show that the iterates in the associated RL algorithm converge to an optimal solution. It is our belief that our results will be of use to a practicing analyst interested in using RL. The rest of this article is organized as follows. Section 2 presents a discussion of our experiments with discounted reward, Section 3 presents our results with the SSP-grounding mechanism on average-reward problems, and Section 4 discusses our algorithm with the survival probability considerations. Section 5 concludes this paper.

2 DISCOUNTED REWARD

The impact of the *rate* of convergence of linear and polynomial step sizes on the values to which Q-values in RL converge has been studied in Even-Dar and Mansour (2003). They have established *theoretically* that linear rules (e.,g, $1/k$, where k denotes the iteration number) can take an exponential time to converge while polynomial rules (e.g., $1/k^{\psi}$ where ψ is some integer) can converge in polynomial time. In this paper, our goal is less ambitious; we wish to conduct experiments with some simple step sizes to test how they perform *empirically* and how far they stray from the optimal values in an empirical setting. It is well-known that, in practice, one has to fine tune the performance of an RL algorithm via trials of numerous step sizes, and we believe that it will be beneficial to a practical user of these algorithms to know how some of the well-known rules perform under known conditions. We chose the following rules: $1/k$, $a/(b+k)$ (note that $1/k$ is a special case of this), and $log(k)/k$. A disadvantage of $a/(b+k)$ is that one has to conduct numerous trials to determine suitable values of a and b, where as the other rules do not have such parameters.

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

We will restrict our attention to the asynchronous Q-Learning algorithm (Watkins 1989) for which convergence has been established under asynchronous conditions in numerous works (see e.g., Borkar and Meyn (2000)). In all the literature, the step-sizes are required to satisfy some basic conditions such as $\sum_{k=1}^{\infty} \alpha^k = \infty$ and $\sum_{k=1}^{\infty} \left(\alpha^k\right)^2 < \infty$ where α^k denotes the step size in the kth iteration. For some other less well-known conditions, see Borkar (1998); all the three rules we consider satisfy these conditions. Our tests will compare the performance of a Q-Learning algorithm with that of value iteration (Puterman 1994) which instead of computing the value function computes Q-values.

We will now present some notation. Let $r(i,a,j)$ denote the reward earned in going from state i to state j under action a. Let $p(i,a,j)$ denote the probability associated with the same transition. We will use μ to denote a policy for which $\mu(i)$ will denote the (deterministic) action to be chosen in state i; e.g., $(2,1)$ will denote a policy with action 2 in state 1 and action 1 in state 2. Let λ denote the discount factor. Also, $\mathbf{P}_\mu$ and $\mathbf{R}_\mu$ will denote the transition probability and transition reward matrices, respectively, associated with policy μ. Finally, $Q(i,a)$ will denote the Q-value for state i and action a.

2.1 Parameters for mdp1

The first test instance, which we call $mdp1$, is a 2-state MDP with the following parameters: $\lambda = 0.8$, and

$$\mathbf{P}_{(1,1)} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{P}_{(2,2)} = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix};$$

$$\mathbf{R}_{(1,1)} = \begin{bmatrix} 6.0 & -5 \\ 7.0 & 12 \end{bmatrix}; \mathbf{R}_{(2,2)} = \begin{bmatrix} 10.0 & 17 \\ -14 & 13 \end{bmatrix}.$$

2.2 Parameters for other test instances

We use 3 other test instances, which are defined as follows. All the parameters for the remaining test instances are identical to those of $mdp1$ with the following exceptions: $mdp2$ — $r(1,1,2) = 5$ and $r(2,2,1) = 14$; $mdp3$ — $r(1,2,1) = 12$; $mdp4$ — $r(1,1,1) = 16$.

2.3 Numerical results

We now present numerical results obtained in five settings: Q-Learning with the three different step-size rules, Q-learning with a neuron that uses the log-rule for the neuron's learning rule, and value iteration performed with Q-values; see Table 1. The value of $a = 150$ and $b = 300$ in our experiments. Also, $\varepsilon = 0.01$ in the value iteration algorithm (Puterman 1994); the main transformation in that algorithm is: For all (i,a) do until ε-convergence: $Q(i,a) \leftarrow \sum_j p(i,a,j)\,[r(i,a,j) + \lambda \max_b Q(j,b)]$. The Q-Learning algorithms were run for 10,000 iterations, with a exploration probability set at 0.5 throughout. The computer programs were written in MATLAB and can be found at `<www.eng.buffalo.edu/~agosavi/codes/wsccodes.html>`.

Table 1: This table compares the Q-values obtained via Q-Learning (Q-L) under the various step-size rules, via a neuron coupled with Q-Learning (N-QL), and via value iteration using Q-values (Q-VI). Q-L-ab will denote Q-Learning with rule $a/(b+k)$, Q-L-k will denote Q-Learning with rule $1/k$ and Q-Learning with the log rule will be denoted by Q-L-log.

	Method	$Q(1,1)$	$Q(1,2)$	$Q(2,1)$	$Q(2,2)$
$mdp1$	Q-VI	44.84	53.02	51.87	49.28
$mdp1$	Q-L-ab	44.40	52.97	51.84	46.63
$mdp1$	Q-L-k	11.46	18.74	19.62	16.52
$mdp1$	Q-L-log	39.24	47.79	45.26	42.24
$mdp1$	N-QL	43.90	51.90	51.54	49.26
$mdp2$	Q-VI	51.67	55.76	57.34	61.45
$mdp2$	Q-L-ab	51.55	55.53	57.11	60.94
$mdp2$	Q-L-k	17.12	20.38	21.08	23.53
$mdp2$	Q-L-log	45.61	50.16	50.07	53.78
$mdp2$	N-QL	50.99	54.70	57.27	62.01
$mdp3$	Q-VI	50.36	60.83	56.66	53.59
$mdp3$	Q-L-ab	49.89	60.82	56.66	51.18
$mdp3$	Q-L-k	12.54	21.72	20.17	16.89
$mdp3$	Q-L-log	43.96	54.83	49.09	45.60
$mdp3$	N-QL	49.20	59.43	56.19	53.38
$mdp4$	Q-VI	48.97	40.91	49.36	47.02
$mdp4$	Q-L-ab	47.72	40.29	48.93	43.93
$mdp4$	Q-L-k	16.16	9.16	18.96	16.04
$mdp4$	Q-L-log	42.73	34.97	42.38	39.72
$mdp4$	N-QL	48.64	40.72	49.71	47.76

The results show that while all the RL algorithms converge to the optimal policy, the $1/k$-rule produces values that remain far away from the optimal Q-values generated by the value iteration algorithm. Perhaps this behavior can be improved by reducing exploration, but that will introduce additional parameters for tuning. What is interesting is that theoretically all the rules are guaranteed to take us to optimal Q-values. The best performance was produced by the $a/(b+k)$ rule; it must be noted, however, that the log-rule which does not have any tuning parameter performs much better than the $1/k$-rule in terms of approximating the value function. The poor performance of $1/k$ can be explained by the fact that it decays very quickly. Also, encouraging is the performance of a neuron-coupled Q-Learning algorithm that uses a log-rule for the neuron's internal learning and an $a/(b+k)$-rule for the algorithm. The results indicate that (i) $1/k$ (used in Gosavi (2004a)) is perhaps not an ideal choice for most cases, (ii) the log rule appears to be promising, and (ii) there is a need to find parameter-less rules (which do not

have parameters such as a and b) that can be used without elaborate experimentation. It needs to be pointed out that in large-scale problems, one does not have the luxury of knowing what the optimal value function is and it is very critical that one has a step-size rule that takes one close to optimality. In large-scale problems, it is quite possible that the rule which causes significant deviation from the optimal value function actually leads one to a sub-optimal policy.

3 AVERAGE REWARD

We now turn our attention to average-reward MDPs. We will perform computational studies with a Q-Learning algorithm that uses two time scales for updating and hence needs two different step sizes simultaneously (Gosavi 2004b). Other algorithms with proven convergence properties include a version of Q-Learning based on relative value iteration (see e.g., Borkar and Meyn (2000)). Here, we wish to study the impact of the stochastic shortest path on average reward problems in RL (Bertsekas 1995).

We will compute the optimal value function using a value iteration algorithm for average reward. Let ρ^{μ} denote the average reward of the policy μ, and ρ^* denote the optimal average reward. Then if ρ^* is known, one can develop a value iteration algorithm for average reward problems. It must be noted that such a value iteration algorithm is being studied here only for the sake of testing how far the Q-Learning algorithm strays from the optimal values (clearly, in practice ρ^* is unknown, and one must use other algorithms; see e.g., Puterman (1994)). The value iteration algorithm will have the following main transformation: For all (i,a) do until ε-convergence: $Q(i,a) \leftarrow \sum_j p(i,a,j)\,[r(i,a,j) - \rho^* + \max_b Q(j,b)]$. The Q-learning algorithm with its SSP-grounding mechanism is described in the Appendix. It has two step sizes: $\alpha(k)$ for the Q-value and $\beta(k)$ for the value of ρ, where $\lim_{k\to\infty} \beta(k)/\alpha(k) = 0$. We use the test instances used in the last section with the understanding that there is now no discount factor. The results are tabulated in Table 2. We ran the Q-learning algorithm for 10,000 iterations and used $\varepsilon = 0.01$; also $mdp1$ — $\rho^* = 10.56$, $mpd2$ — $\rho^* = 11.53$, $mdp3$ — $\rho^* = 12.00$ and $mdp4$ — $\rho^* = 9.83$. These values for ρ^* were determined by an exhaustive evaluation of the average reward of each deterministic policy. The exploration probability was fixed at 0.5 for both actions. The results show that the value function, which is defined as $v(i) = \max_a Q(i,a)$, is reasonably approximated by the Q-Learning algorithm, although some Q-values are not so well approximated.

3.1 Is Bellman optimality worth achieving?

The numerical results of this section and the previous section raise an important question. Is Bellman optimality, which means achieving the value function that would result from solving the Bellman equation, really worth achieving, or would it be okay for an algorithm to generate the optimal solution? Note that in Section 2.3, the $1/k$-rule and the log-rule generate optimal policies, although the value function they generate strays considerably from that generated by dynamic programming (Bellman equation). The same is true of the results for average reward. This is an issue that requires further analysis. An important question that needs to be addressed is: how much deviation in the value function can be tolerated? In other words, by how much can the value function deviate without resulting in a sub-optimal policy? The answer to this question might pave the way to solving the MDP without strict adherence to Bellman principles. Gosavi (2004a) has shown that that for any given state, if the absolute value of the error in the value function is less than half of the absolute value of the difference between the Q-value of the optimal function and the Q-value of the sub-optimal action (assuming we have 2 actions in each state), then that error can be tolerated. But an in-depth study of this issue may prove to be of importance in the future — especially in the context of function approximation, where we have clear deviation from Bellman optimality.

Table 2: This table compares the Q-values obtained via Q-Learning (Q-L) for average reward (see Appendix) and via value iteration using Q-values (Q-VI). For $mdp2$ $\alpha(k) = 500/(1000+k)$ and $\beta(k) = 150/(300+k)$, while for the remaining instances we used $\alpha(k) = 150/(300+k)$ and $\beta(k) = 50/(49+k)$.

	Method	$Q(1,1)$	$Q(1,2)$	$Q(2,1)$	$Q(2,2)$
$mdp1$	Q-L	-3.46	0.1710	-1.89	-3.02
$mdp1$	Q-VI	-7.99	0.2789	-1.12	-3.80
$mdp2$	Q-L	-2.85	0.57	4.48	6.10
$mdp2$	Q-VI	-1.85	0.37	7.18	7.31
$mdp3$	Q-L	-4.99	0.1061	-4.81	-5.19
$mdp3$	Q-VI	-9.80	0.99	-3.996	-7.39
$mdp4$	Q-L	-1.14	-8.19	-0.298	-3.94
$mdp4$	Q-VI	-0.1904	-8.28	0.24	-2.08

4 SURVIVAL PROBABILITY

The notion of risk has been studied in the context of RL via utility functions (Borkar 2002), variance penalties (Sato and Kobayashi 2001), and probability of entering forbidden states (Geibel and Wysotzki 2005). See Heger (1994) for an earlier work. Variance penalties in the context of MDPs were studied in Filar, Kallenberg, and Lee (1989). In this paper, we consider the penalties associated with downside risk which is defined with respect to a target. Given a target for the one-step reward, we define the downside risk (DR) to be the probability of the reward falling below the target; this risk should be minimized. Hence $1 - DR$ will

denote the probability of survival, which is maximized. If one considers costs instead of rewards, the probability of exceeding the target will be the associated downside risk; this is also called the exceedance probability in catastrophe modeling (Grossi and Kunreuther 2005). We next present intuitively-conjectured Bellman and Poisson equations, and then an RL algorithm.

4.1 Bellman Equation

Let τ denote the target one-step reward. Then for a given deterministic, stationary policy μ, the downside risk is defined as:

$$DR^{\mu} = \sum_{i \in \mathscr{S}} \Pi^{\mu}(i) \sum_{j \in \mathscr{S}} p(i,\mu(i),j) I(r(i,\mu(i),j) < \tau) \quad (1)$$

where $I(.)$ denotes the indicator function (which equals 1 if the condition inside the brackets is true and 0 otherwise) and $\Pi^{\mu}(i)$ denotes the limiting probability (invariant probability) for state i under policy μ. Our objective function in the downside-risk-penalized problem will be: $\phi = \rho - \theta DR$ where ρ denotes the average reward and θ is a positive scalar chosen by the risk manager. The greater the value of θ, the higher the risk-sensitivity. Such parameters were popularized by the pioneering work by Markowitz (1952) in finance, and have been used in MDPs by Filar, Kallenberg, and Lee (1989) and Gosavi (2006) and in RL by Sato and Kobayashi (2001). We now propose Bellman and Poisson equations for this objective function without proof. A theoretical analysis will be the subject of future work.
(i) (Poisson equation) If a scalar $\phi \in \Re$ and an $|S|$-dimensional finite vector $\vec{h}$ satisfy for all $i \in \mathscr{S}$:

$$\phi + h(i) =$$

$$\sum_{j \in \mathscr{S}} p(i,\mu(i),j) \left[r(i,\mu(i),j) - \theta I(r(i,\mu(i)j) < \tau) + h(j) \right]$$

then ϕ is the variance-penalized score associated with the policy μ.
(ii) (Bellman equation) Assume that a scalar ϕ^* and an $|\mathscr{S}|$-dimensional finite vector $J(i)$ satisfy for all $i \in \mathscr{S}$

$$\phi^* + J(i) = \max_{a \in \mathscr{A}(i)}$$

$$\left[\sum_{j \in \mathscr{S}} p(i,a,j) [r(i,a,j) - \theta I(r(i,a,j) < \tau) + J(j)] \right]. \quad (2)$$

Any policy that attains the max in the RHS of the above will be an optimal policy, i.e., it will generate the maximum value for the risk-penalized score. Analogous equations for variance penalties and semi-variance penalties can be constructed. Variance in the MDP for a policy μ is defined as:

$$\sum_{i \in \mathscr{S}} \Pi^{\mu}(i) \sum_{j \in \mathscr{S}} p(i,\mu(i),j)(r(i,\mu(i),j) - \rho_{\mu})^2$$

and semi-variance as:

$$\sum_{i \in \mathscr{S}} \Pi^{\mu}(i) \sum_{j \in \mathscr{S}} p(i,\mu(i),j)(\tau - r(i,\mu(i),j))_+^2$$

where $a_+ = \max(0,a)$. For semi-variance, the indicator function would be replaced by $(\tau - r(i,\mu(i),j))_+^2$ in the Poisson equation and by $(\tau - r(i,a,j))_+^2$ in the Bellman equation (Gosavi 2008), and for variance, we would replace the indicator function in the Bellman equation by $(r(i,a,j) - \rho^*)^2$, where ρ^* is the average reward of the optimal policy of the variance-penalized MDP (Gosavi and Meyn 2008). While variance and semi-variance are acceptable measures of risk, downside risk is even more appealing because it is a probability measure. We now present an RL algorithm for downside-risk penalties.

4.2 Q-Learning for survival

A Q-value version of the Bellman equation can be developed from Equation (2) above. From that, it is not difficult to derive a Q-Learning algorithm. Since the Bellman equation models the optimal value of the objective function, ϕ^*, (this is analogous to ρ^* in the risk-neutral Bellman equation for average reward), we need to use an algorithm that uses *relative* values. Our algorithm's main features are as follows. In the first step, choose some state-action pair to be a distinguished state-action pair; call it (i^*,a^*). The main update in the simulator is:

$$Q^{k+1}(i,a) \leftarrow (1-\alpha(k))Q^k(i,a) + \alpha(k) \times$$

$$\Big[r(i,a,j) - \theta I(r(i,a,j) < \tau) - Q^k(i^*,a^*) +$$

$$+ \max_{b \in \mathscr{A}(j)} Q^k(j,b) \Big],$$

where $Q^k(.,.)$ is the Q-value in the kth iteration. For the risk-neutral case, it can be shown that with probability 1, $\lim_{k \to \infty} Q^k(i^*,a^*) = \rho^*$ (see Borkar and Meyn (2000)); intuition suggests that with probability 1:

$$\lim_{k \to \infty} Q^k(i^*,a^*) = \phi^*. \quad (3)$$

Of course, as stated earlier, a theoretical proof is a subject of future work. We will now conduct simulation tests to determine how the algorithm performs.

4.3 Parameters for test instances

We use four test instances named *mdp5* through *mdp8*. For all the test instances, $\tau = 8$ and $\theta = 2$. We describe them next.
mdp5: Identical to $mdp1$ except for: $r(1,1,1) = 3; r(1,1,2) = 11; r(1,2,1) = 6; r(2,2,2) = 7$.
mdp6: Identical to $mdp1$ except for: $r(1,1,1) = 3; r(1,1,2) = 11; r(2,1,2) = 9; r(1,2,1) = 6; r(2,2,2) = 7$.
mdp7: Identical to $mdp1$ in terms of the transition probabilities, but with the following reward structures:

$$\mathbf{R}_{(1,1)} = \begin{bmatrix} 9.0 & -1 \\ 12.0 & 8 \end{bmatrix}; \mathbf{R}_{(2,2)} = \begin{bmatrix} 6.0 & 20 \\ -14 & 7 \end{bmatrix}.$$

mdp8: Identical to $mdp1$ in terms of the transition probabilities, but with the following reward structures:

$$\mathbf{R}_{(1,1)} = \begin{bmatrix} 3.0 & 7 \\ 9.0 & 1 \end{bmatrix}; \mathbf{R}_{(2,2)} = \begin{bmatrix} 6.0 & 9 \\ 14 & 7 \end{bmatrix}.$$

4.4 Simulation experiments

We first analyzed via exhaustive evaluation the average reward and the downside risk for each policy in the 4 test instances. The average reward is $\rho^\mu = \sum_{i\in\mathscr{S}} \Pi^\mu(i) \sum_{j\in\mathscr{S}} p(i,\mu(i),j) r(i,\mu(i),j)$. The downside risk is defined in Equation (1). The limiting probabilities of each state can be determined by solving the classical invariant equations: $\sum_{j\in\mathscr{S}} \Pi^\mu(j) p(j,\mu(i),i) = \Pi^\mu(i)$ for all $i \in \mathscr{S}$ and $\sum_{i\in\mathscr{S}} \Pi^\mu(i) = 1$. The results are presented in Table 3. On all the examples, the algorithm converged to optimal solutions in 10,000 iterations. We fix the exploration probability at a value of 0.5 for each action. We also show in the table the value to which $Q(i^*,a^*)$ converges. We do not present all the $Q(.,.)$ values because we have not compared them to values from dynamic programming, and hence the values by themselves convey nothing. What is more interesting is the value to which $Q(i^*,a^*)$ converges. As is expected from Equation (3), it converges to a value very close to ϕ^*. Also, note that for $mdp5$, $mdp6$, and $mdp7$, the risk-neutral optimal policy (that maximizes ρ) does not coincide with the risk-sensitive optimal policy (that maximizes our risk-penalized score, ϕ).

4.5 Semi-Markov control

A natural and important extension of MDP theory is to Semi-Markov decision processes (SMDPs) (Puterman 1994), where the time spent in each transition is modeled as a random variable. Let $t(i,a,j)$ denote the time spent in going from i to j under action a. We first need the definitions of the risk measures considered above. Downside risk will be defined as:

$$DR^\mu = \sum_{i\in\mathscr{S}} \Pi^\mu(i) \sum_{j\in\mathscr{S}} p(i,\mu(i),j) I(\frac{r(i,\mu(i),j)}{t(i,\mu(i),j)} < \tau).$$

The corresponding Bellman equation would be: $J(i) =$

$$\max_{a\in\mathscr{A}(i)} \Big[\sum_{j\in\mathscr{S}} p(i,a,j) \{ r(i,a,j) - \theta I(r(i,a,j) < \tau t(i,a,j))$$

$$- \phi^* t(i,a,j) + J(j) \} \Big]$$

and the Q-learning algorithm can be derived to be:

$$Q^{k+1}(i,a) \leftarrow (1-\alpha(k)) Q^k(i,a) + \alpha(k) \times$$

$$\left[r(i,a,j) - \theta I(\frac{r(i,a,j)}{t(i,a,j)} < \tau) - Q^k(i^*,a^*) t(i,a,j) \right]$$

$$+ \alpha(k) \left[\max_{b\in\mathscr{A}(j)} Q^k(j,b) \right],$$

Semi-variance in the SMDP can be defined as:

$$\sum_{i\in\mathscr{S}} \Pi^\mu(i) \sum_{j\in\mathscr{S}} p(i,\mu(i),j) (\tau t(i,\mu(i),j) - r(i,\mu(i),j))^2_+.$$

The SMDP Bellman equation for semi-variance can be obtained from that of downside risk via replacement of the indicator function by $(\tau t(i,a,j) - r(i,a,j))^2_+$. A Q-learning algorithm can also be derived for semi-variance.

For variance, we need to define some quantities first. We will use the renewal reward theorem (RRT) (Ross 1997) because underlying the SMDP, one has a renewal process. Consider a counting process $\{N(t), t \geq 0\}$, and let T_n denote the time between the $(n-1)$th event and the nth event in the process; $n \geq 1$. If $\{T_1, T_2, \ldots\}$ denotes a sequence of non-negative i.i.d random variables, then $\{N(t), t \geq 0\}$ is a renewal process. Let R_n denote the reward accrued in the nth renewal in the renewal process underlying the SMDP. Also, let $E[R_n] \equiv E[R]$ and $E[T_n] \equiv E[T]$. The average reward for the SMDP can be shown via the RRT to be : $\rho = E[R]/E[T]$ where (the action a in each state i is defined by the policy under consideration)

$$E[R] = \sum_{i\in\mathscr{S}} \Pi(i) \sum_{j\in\mathscr{S}} p(i,a,j) r(i,a,j) \text{ and}$$

$$E[T] = \sum_{i\in\mathscr{S}} \Pi(i) \sum_{j\in\mathscr{S}} p(i,a,j) t(i,a,j).$$

The natural definition for the asymptotic variance is defined in (4) below. From the RRT, we know that with probability

1 (w.p.1), $\lim_{t\to\infty}\frac{N(t)}{t}=\frac{1}{E[T]}$ using which we can work out the following:

$$\begin{aligned}\sigma^2 &\equiv \lim_{t\to\infty}\frac{\sum_{n=1}^{N(t)}[R_n-\rho T_n]^2}{t} \quad (4)\\ &= \lim_{t\to\infty}\sum_{n=1}^{N(t)}\left[\frac{R_n^2-2\rho T_nR_n+\rho^2T_n^2}{N(t)}\right]\frac{N(t)}{t}\\ &= \frac{E[R^2]}{E[T]}-2\rho\frac{E[T\cdot R]}{E[T]}+\rho^2\frac{E[T^2]}{E[T]} \quad (w.p.1)\\ &= \frac{E[R^2]}{E[T]}-2\rho\frac{E[T]E[R]}{E[T]}+\rho^2\frac{E[T^2]}{E[T]}\\ &\quad \text{(since } T \text{ and } R \text{ independent)}\\ &= \frac{E[R^2]}{E[T]}-2\rho^2E[T]+\rho^2\frac{E[T^2]}{E[T]},\end{aligned}$$

where

$$E[R^2]=\sum_{i\in\mathscr{S}}\Pi(i)\sum_{j\in\mathscr{S}}p(i,a,j)r^2(i,a,j) \text{ and}$$

$$E[T^2]=\sum_{i\in\mathscr{S}}\Pi(i)\sum_{j\in\mathscr{S}}p(i,a,j)t^2(i,a,j).$$

Using $E[R],E[R^2],E[T]$, and $E[T^2]$ one can define the variance of the SMDP. Then if ρ^* denotes the average reward of the policy that optimizes a variance-penalized SMDP, then the Bellman equation for the variance-penalized SMDP can be obtained by replacing the indicator function in the corresponding equation for downside risk by $(r(i,a,j)-\rho^*t(i,a,j))^2$.

5 CONCLUSIONS

This paper presented an empirical study of (i) the use of different step-sizes in discounted RL, (ii) the use of shortest stochastic paths in average reward RL, and (iii) the notion of survival probability or downside risk in RL. The empirical study with the step size (Section 2) indicates that the $1/k$-rule does not appear to be a reliable or robust choice even on very small problems, and that the $(a/b+k)$-rule performs very well on small problems, but the values of a and b need to be determined. The log-rule performs reasonably well, and its advantage is that it does not have any tuning parameters. The empirical study with the stochastic paths (Section 3) indicates that using SSP grounding, one obtains reasonable approximations of the actual value function. Our empirical results do point to the need for studying how much deviation can be tolerated from Bellman optimality. Finally, in Section 4, we present a new Q-Learning algorithm that allows the optimization of a survival-probability-penalized objective function. Numerical results on small test problems indicate that the algorithm performs well. A theoretical study of this algorithm is a topic for future research.

Table 3: This table lists the ρ, DR and ϕ values of all the policies along with the value of $Q^\infty(i^*,a^*)$; we used $i^*=1$ and $a^*=1$. The values in bold are those for the optimal policy.

	μ	ρ^μ	DR^μ	ϕ^μ	$Q(i^*,a^*)$
$mdp5$	(1,1)	**7.3714**	**0.5714**	**6.2286**	6.3340
	(1,2)	3.84	0.8800	2.0800	-
	(2,1)	7.68	0.8	6.08	-
	(2,2)	5.6667	0.9333	3.8	-
$mdp6$	(1,1)	**7.7143**	**0.4**	**6.9143**	6.9545
	(1,2)	3.84	0.88	2.08	-
	(2,1)	7.84	0.72	6.4	-
	(2,2)	5.6667	0.9333	3.8	-
$mdp7$	(1,1)	**7.5429**	**0.1714**	**7.2**	7.2097
	(1,2)	4.08	0.72	2.64	-
	(2,1)	7.84	0.72	6.4	-
	(2,2)	5.866	0.9333	4.000	-
$mdp8$	(1,1)	4.2	0.8286	2.5429	-
	(1,2)	6.72	0.88	4.96	-
	(2,1)	5.88	0.84	4.2	-
	(2,2)	**7**	**0.8667**	**5.266**	5.3043

A APPENDIX

We present some details of the Q-Learning algorithm used for average reward (Gosavi 2004b).

Step 1. Set all $Q(l,u)\leftarrow 0$. Set k, the number of state changes, to 0. Set ρ^k, the estimate of the average reward per transition in the kth transition, to 0. Also set W to 0. Run the algorithm for $k_{\max}$ iterations, where $k_{\max}$ is sufficiently large. Start system simulation at any arbitrary state. Select some state to be a special state s^*.

Step 2. Let the current state be i. Select action a with a probability of $1/|\mathscr{A}(i)|$ where $\mathscr{A}(i)$ denotes the set of actions allowed in state i. A greedy action in state i is the action associated with the highest Q-value for i.

Step 3. Simulate action a. Let the next state be j. Let $r(i,a,j)$ be the immediate reward earned in going to j from i under a. Then update $Q(i,a)$ as follows:

$$Q(i,a)\leftarrow(1-\alpha(k))Q(i,a)+\alpha(k)\left[r(i,a,j)-\rho^k+M_b^j\right],$$

where $M_b^j=0$ if $j=s^*$ (this is called SSP grounding) and $M_b^j=\max_{b\in\mathscr{A}(j)}Q(j,b)$ otherwise.

Step 4. If a greedy action was selected in Step 2, then increment W as follows: $W\leftarrow W+r(i,a,j)$ and then update

ρ^k as follows:

$$\rho^{k+1} = (1-\beta(k))\rho^k + \beta(k)\frac{W}{k}$$

Step 5. If $k < k_{max}$, set $i \leftarrow j$, $k \leftarrow k+1$, and then go to Step 2. Otherwise, go to Step 6.
Step 6. For each state i, select $\mu(i) \in \text{argmax}_{a \in \mathscr{A}(i)} Q(i,a)$. The policy (solution) generated by the algorithm is μ. Stop.

REFERENCES

Bertsekas, D. 1995. *Dynamic programming and optimal control*. Belmont: Athena.

Borkar, V. 2002. *Q*-learning for risk-sensitive control. *Mathematics of Operations Research* 27(2):294–311.

Borkar, V. S. 1998. Asynchronous stochastic approximation. *SIAM Journal of Control and Optimization* 36 No 3:840–851.

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

Even-Dar, E., and Y. Mansour. 2003. Learning rates for Q-learning. *Journal of Machine Learning Research* 5:1–25.

Filar, J., L. Kallenberg, and H. Lee. 1989. Variance-penalized Markov decision processes. *Mathematics of Operations Research* 14(1):147–161.

Geibel, P., and F. Wysotzki. 2005. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research* 24:81–108.

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

Gosavi, A. 2004b. Reinforcement learning for long-run average cost. *European Journal of Operational Research* 155:654–674.

Gosavi, A. 2006. A risk-sensitive approach to total productive maintenance. *Automatica* 42:1321–1330.

Gosavi, A. 2008. Markov decision processes subject to semivariance risk. Working paper at University at Buffalo, SUNY, ISE Department.

Gosavi, A., and S. Meyn. 2008. The variance-penalized Bellman equation. Working paper at SUNY Buffalo and University of Illinois at Urbana-Champaign.

Grossi, P., and H. Kunreuther. 2005. *Catastrophe modeling: A new appraoch to managing risk*. Springer.

Heger, M. 1994. Consideration of risk in reinforcement learning. *Proceedings of the 11th International Conference on Machine Learning*:105–111.

Markowitz, H. 1952. Portfolio selection. *Journal of Finance* 7(1):77–91.

Puterman, M. L. 1994. *Markov decision processes*. New York: Wiley Interscience.

Ross, S. M. 1997. *Introduction to Probability Models*. Academic Press, San Diego, CA, USA.

Sato, M., and S. Kobayashi. 2001. Average-reward reinforcement learning for variance-penalized Markov decision problems. In *Proceedings of the 18th International Conference on Machine Learning*, 473–480. Morgan Kauffman.

Sutton, R., and A. G. Barto. 1998. *Reinforcement learning: An introduction*. Cambridge, MA, USA: The MIT Press.

Watkins, C. 1989, May. *Learning from delayed rewards*. Ph. D. thesis, Kings College, Cambridge, England.

AUTHOR BIOGRAPHY

ABHIJIT GOSAVI is an Assistant Professor in the Department of Engineering Management at the Missouri University of Science and Technology. His research interests lie in simulation-based optimization, supply chain management, and lean manufacturing. He is a member of IEEE, ASEE, POMS, IIE and INFORMS. His email address for these proceedings is <gosavia@mst.edu>.