Reinforcement Learning for Matrix Computations: PageRank as an Example

Vivek S. Borkar and Adwaitvedant S. Mathkar

Department of Electrical Engineering,
Indian Institute of Technology,
Powai, Mumbai 400076, India
{borkar.vs,mathkar.adwaitvedant}@gmail.com

Abstract. Reinforcement learning has gained wide popularity as a technique for simulation-driven approximate dynamic programming. A less known aspect is that the very reasons that make it effective in dynamic programming can also be leveraged for using it for distributed schemes for certain matrix computations involving non-negative matrices. In this spirit, we propose a reinforcement learning algorithm for PageRank computation that is fashioned after analogous schemes for approximate dynamic programming. The algorithm has the advantage of ease of distributed implementation and more importantly, of being model-free, i.e., not dependent on any specific assumptions about the transition probabilities in the random web-surfer model. We analyze its convergence and finite time behavior and present some supporting numerical experiments.

Keywords: Reinforcement Learning, PageRank, Stochastic Approximation, Sample Complexity.

1 Introduction

Reinforcement learning has its roots in models of animal behavior [1] and mathematical psychology [2], [3]. The recent resurgence of interest in the field, however, is propelled by applications to artificial intelligence and control engineering. By now there are several textbook accounts of this development [4] (Chapter 16), [5], [6], [7], [8], [9]. To put things in context, recall that methodologically, reinforcement learning sits somewhere in between supervised learning, which works with a reasonably accurate information regarding the performance gradient or something analogous (e.g., parameter tuning of neural networks), and unsupervised learning, which works without such explicit information (e.g., clustering). To be specific, supervised learning is usually based upon an optimization formulation such as minimizing an error measure, which calls for a higher quantum of information per iterate. Reinforcement learning on the other hand has to manage with signals somehow correlated with performance, but which fall short of the kind of information required for a typical supervised learning scheme. It then makes simple incremental corrections based on these 'suggestive though inexact'

R. Natarajan (Ed.): ICDCIT 2014, LNCS 8337, pp. 14–24, 2014.

signals, usually with low per iterate computation. The latter aspect has also made it a popular framework for models of bounded rationality in economics [10].

Our interest is in its recent avatar as a scheme for simulation-based methodology for approximate dynamic programming for Markov decision processes which has found applications, among other things, in robotics [11]. These can be viewed as stochastic approximation counterparts of the classical iterative methods for solving dynamic programming equations, such as value and policy iteration. Stochastic approximation, introduced by Robbins and Monro [12] as an iterative scheme for finding the roots of a nonlinear function given its noisy measurements, is the basis of most adaptive schemes in control and signal processing. What it does in the present context is to replace a conditional average appearing on the right hand side of the classical iterative schemes (or their variants) by an actual evaluation at a simulated transition according to the conditional distribution in question. It then makes an incremental move towards the resulting random quantity. That is, it takes a convex combination of the current value and the random right hand side, with a slowly decreasing weight on the latter. The averaging properties of stochastic approximation then ensure that asymptotically you see the same limiting behavior as the original scheme.

But there are other situations wherein one encounters iterations involving conditional averages. In fact, by pulling out row sums of a non-negative matrix into a diagonal matrix pre-multiplier, we can write it as a product of a diagonal matrix and a stochastic matrix. This allows us to cast iterations involving non-negative matrices as iterations involving averaging with respect to stochastic matrices, making them amenable to the above methodology. This opens up the possibility of using reinforcement learning schemes for distributed matrix computations of certain kind. Important instances are plain vanilla averaging and estimation of the Perron-Frobenius eigenvectors [13]. Reinforcement learning literature is replete with means of curtailing the curse of dimensionality, a hazard only too common in dynamic programming applications. This machinery then becomes available for such matrix computations. An important special case is the case of linear function approximation, wherein one approximates the desired vector by a weighted combination of a moderate number of basis vectors, and then updates these weights instead of the entire vector [14].

In the present article, we illustrate this methodology in the context of Google's PageRank, an eigenvector-based ranking scheme. It is primarily based on the stationary distribution π of the 'random web-surfer' Markov chain, equivalently, the normalized left Perron-Frobenius eigenvector of its transition probability matrix. This chain is defined on a directed graph wherein each node i is a web page. Let $\mathcal{N}(i) :=$ the set of nodes to which i points. Let $d(i) := |\mathcal{N}(i)|$ and $N :=$ the total number of nodes. The chain moves from i to $j \in \mathcal{N}(i)$ with a probability $(1-c)\frac{1}{d(i)} + \frac{c}{N}$, and to any other node in the graph with probability $\frac{c}{N}$ where $c > 0$ is the 'Google constant'. The latter renders it irreducible, ensuring a unique stationary distribution. An excellent account of the numerical techniques for computing π, essentially based on the 'power method' and its variants, appears

in [15], along with a brief historical account. See also [16]. While a bulk of the work in this direction has been on efficient computations for the power method, there have also been alternative approaches, such as Markov Chain Monte Carlo [17], [18], optimization based methods [19], and schemes based on stochastic approximation and/or gossip [20], [21], [22].

Such 'spectral ranking' techniques, made popular by the success of PageRank, are in fact quite old. See [23] for a historical survey. Evaluative exercises of this kind occur in other applications as well, such as reputation systems or popularity measures on social networks. In such applications (for that matter, in search), it is unclear whether the assumption that each $j \in \mathcal{N}(i)$ is equally important to i is reasonable. Motivated by this, we propose a model-free scheme based on ideas from reinforcement learning. This idea has also been discussed in [13]. The present scheme, however, differs in an essential way from [13] in that whereas [13] views PageRank as a special instance of the general problem of eigenvector estimation, we exploit the special structure of the random web-surfer model to simplify the problem to a simple linear scheme. This is very much in tune with some of the works cited above (notably [20], [22]), but with a non-standard sample and update rule. The outcome is an algorithm that can run on accumulated traces of node-to-node interactions without requiring us to explicitly estimate the probabilities associated with these.

The next section describes our algorithm and its convergence analysis. Section 3 describes finite time analysis and a variant of the basic scheme. Section 4 presents some numerical experiments. Section 5 concludes with some general observations.

2 The Algorithm

Let P be an $N \times N$ stochastic matrix. Define $\hat{P} := cP + \frac{1-c}{N}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$. Let π denote the unique stationary probability distribution of $\hat{P}$. That is, for $\mathbf{1} := [1, 1, \cdots, 1]^T$,

$$\begin{aligned} \pi &= \pi\hat{P} \\ &= \pi cP + \pi\frac{1-c}{N}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \\ &= c\pi P + \frac{1-c}{N}\mathbf{1}^T \\ \Rightarrow \pi(I - cP) &= \frac{1-c}{N}\mathbf{1}^T \\ \Rightarrow \pi &= \frac{1-c}{N}\mathbf{1}^T(I - cP)^{-1}. \end{aligned}$$

Here π is a row vector and every other vector is a column vector. Since we are only interested in ranking we can neglect the factor $\frac{1-c}{N}$. Thus by abuse of terminology,

$$\pi^T = \mathbf{1} + cP^T\pi^T.$$

To estimate π, we run the following N dimensional stochastic iteration. Sample (X_n, Y_n) as follows: Sample X_n uniformly and independently from $\{1, 2, ..., N\}$. Sample Y_n with $P(Y_n = j|X_n = i) = p(i, j)$, independent of all other random variables realized before n. Update z_n as follows:

$$z_{n+1}(i) = z_n(i) + a(n)(I\{X_{n+1} = i\}(1 - z(n)) + cz_n(X_{n+1})I\{Y_{n+1} = i\}), \quad (1)$$

where the step-sizes $a(n) > 0$ satisfy $\sum_{n=0}^{\infty} a(n) = \infty$ and $\sum_{n=0}^{\infty} a(n)^2 < \infty$. Hence $z_n(i)$ is updated only if X_{n+1}, Y_{n+1} or both are i. We can write (1) as follows:

$$\begin{aligned} &z_{n+1}(i) \\ &= z_n(i) + a(n)(I\{X_{n+1} = i\}(1 - z(n)) + cz_n(X_{n+1})p(X_{n+1}, i) + M_{n+1}(i)), \end{aligned}$$

where $M_{n+1} := cz_n(X_{n+1})I\{Y_{n+1} = i\} - cz_n(X_{n+1})p(X_{n+1}, i)$ is a martingale difference sequence w.r.t. $\sigma(X_m, Y_m, m \leq n; X_{n+1})$. By Theorem 2, p. 81, [24], the ODE corresponding to the iteration is

$$\dot{z}(i) = \frac{1}{N}(1 + \sum_{j=1}^{N} cz(j)p(j, i) - z(i)).$$

In vector form,

$$\dot{z} = \frac{1}{N}(\mathbf{1} + cP^T z - z).$$

Since the constant $1/N$ doesn't affect the asymptotic behavior, we consider

$$\dot{z} = (\mathbf{1} + cP^T z - z) =: h(z). \quad (2)$$

Define $h_\infty(z) := \lim_{a\uparrow\infty} \frac{h(az)}{a} = cP^T z - z$. It is easy to see that $\frac{h(az)}{a} \to h_\infty(z)$ uniformly on R^N.

Theorem 1. Under the above assumptions, $z(t) \to z^*$ a.s., where z^* is the unique solution to $h(z^*) = 0$.

Proof: Define $V_p(z(t)) := \|z(t) - z^*\|_p$, $p \in [1, \infty)$. As in the proof of Theorem 2, p. 126, [24], for $1 < p < \infty$,

$$\dot{V}_p(z(t)) \leq \|cP^T(z(t) - z^*)\|_p - \|z(t) - z^*\|_p.$$

Integrating,

$$V_p(z(t)) - V_p(z(s)) \leq \int_s^t \left(\|cP^T(z(r) - z^*)\|_p - \|z(r) - z^*\|_p \right) dr.$$

Letting $p \downarrow 1$,

$$\begin{aligned} V_1(z(t)) - V_1(z(s)) &\leq \int_s^t \|cP^T(z(r) - z^*)\|_1 - \|z(r) - z^*\|_1 dr, \\ &\leq -\int_s^t (1-c)\|z(r) - z^*\|_1 dr, \\ &\leq 0, \end{aligned}$$

with equality iff $z(t) = z^*$. We similarly get that $V_1(z(t)) := \|z(t)\|_1$ is a Lyapunov function for the scaled o.d.e $\dot{z}(t) = h_\infty(z(t))$ which has the origin as its globally asymptotically stable asymptotic equilibrium. By Theorem 9, p. 75, [24], $\sup_n \|z(n)\| < \infty$ a.s. In turn (2) has z^* as its globally stable asymptotic equilibrium with $V(z(t)) = \|z(t) - z^*\|_1$ as its Lyapunov function. The claim follows from Theorem 7 and Corollary 8, p. 74, [24]. □

3 Remarks

1. We first look at sample complexity of the stochastic iteration. We mainly use section 4.2 of [24] to derive sample complexity estimates.

   Let $1 \leq m < M < N$. Let z^* denote the stationary distribution. Without loss of generality (by relabeling if necessary), let $z_1^* \geq z_2^* \geq .... \geq z_N^*$, i.e., the components of z^* are numbered in accordance with their ranking. We shall consider as our objective the event that the top m ranks of z^* fall within the top M ranks of the output of our algorithm when stopped, for a prescribed pair $m < M$. This is a natural criterion for ranking problems, which are an instance of 'ordinal optimization' [25]. To avoid pathologies, we assume that $z_m^* > z_M^*$. We shall derive an estimate for the number of iterates needed to achieve our aim with 'high' probability. Let $C := \{z \in \Delta^N : \text{ if } z_{l_1} \geq z_{l_2} \geq .... \geq z_{l_N} \text{ then } z_i \geq z_{l_M}, 1 \leq i \leq m\}$, where $\frac{N}{1-c}\Delta^N$ is the N-dimensional probability simplex. Thus C consists of all distributions such that the top m indices of z^* are in the top M indices of the given distribution.

   Let Φ_T be the time-T flow-map associated with the differential equation, where $T > 0$. Thus,

   $$\Phi_T(z) = e^{\frac{cP^T - I}{N}T}(z - (cP^T - I)^{-1}\mathbf{1}) - (cP^T - I)^{-1}\mathbf{1},$$

   with $\Phi_T(z^*) = z^*$. Define

   $$C^* := \{z \in C : \|z - z^*\|_1 \leq \min_{z' \in \partial C} \|z' - z^*\|_1\},$$

and for $\epsilon > 0$,

$$C^\epsilon := \{x : \inf_{y \in C} \|x - y\|_1 < \epsilon\}.$$

Then

$$\begin{aligned}
&\min_{z \in \Delta^N - C} \left[\|z - z^*\|_1 - \|\Phi_T(z) - z^*\|_1 \right] \\
&= \min_{z \in \Delta^N - C} \left[\|z - z^*\|_1 - \|e^{\frac{cP^T - I}{N} T}(z - z^*)\|_1 \right] \\
&\geq \min_{z \in \Delta^N - C} \left[\|z - z^*\|_1 - \|e^{\frac{cP^T - I}{N} T}\|_1 \|(z - z^*)\|_1 \right] \\
&= (1 - \|e^{\frac{cP^T - I}{N} T}\|_1) \min_{z \in \Delta^N - C} \|z - z^*\|_1 \\
&= (1 - \|e^{\frac{cP^T - I}{N} T}\|_1)\kappa
\end{aligned}$$

where $\kappa := \min_{z \in \Delta^N - C} \|z - z^*\|_1$ and $\|A\|_1$ for a matrix A is its induced matrix norm. We argue that $\|e^{\frac{cP^T - I}{N} T}\|_1 < 1$. To see this, view $Q = cP - I$ as the rate matrix of a continuous time Markov chain killed at rate $1 - c$. Then $e^{\frac{cP - I}{N} T}$ is its transition probability matrix after time $\frac{T}{N}$, whose row sums will be uniformly bounded away from 1. The claim follows. Let $\gamma > 0$ and pick $T > 0$ such that

$$\gamma \geq \min_{z \in \Delta^N - C} [\|z - z^*\|_1 - \|\Phi_T(z) - z^*\|_1].$$

Since $\max_{z \in \Delta^N} \|z - z^*\|_1 = 2$,

$$\frac{\max_{z \in \Delta^N} \|z - z^*\|_1}{\gamma/2} \times (T + 1) \leq \tau := \frac{4}{(1 - \|e^{\frac{cP^T - I}{N} T}\|_1)\kappa} \times (T + 1).$$

Let $n_0 := \min\{n \geq 0 : \sum_{m=0}^n a(m) \geq \tau\}$. Also, let $\mathcal{N}_\eta(S) := \{z : \inf_{y \in S} \|z - y\|_2 < \eta\}$ denote the open η-neighborhood w.r.t. $\| \cdot \|_2$ norm of a generic set S. Set $\delta := \frac{\gamma}{2\sqrt{N}}$. Then $\|x - y\|_2 < \delta \Longrightarrow \|x - y\|_1 < \frac{\gamma}{2}$. Arguing as in Corollary 14, p. 43 of [24], we have

$$P(z_n \in N_\delta(C^{\frac{\gamma}{2}}) \ \forall n \geq n_0 + k) \geq 1 - 2Ne^{-\frac{K\delta^2}{N \sum_{m=k}^\infty a(m)^2}} = 1 - o(\sum_{m=k}^\infty a(m)^2),$$

where $K > 0$ is a suitable constant. (In *ibid.*, replace H^ϵ by C^* and Δ by γ.)

2. Note that at each time n, we can generate more than one, say m pairs $(X_n^i, Y_n^i), 1 \leq i \leq m$, which are independent, each distributed as (X_n, Y_n) above, and change the iteration to:

$$\begin{aligned}
z_{n+1}(i) = z_n(i) &+ a(n)(I\{i \in \{X_{n+1}^j, 1 \leq j \leq m\}\}(1 - z(n)) \\
&+ c \sum_{j=1}^m z_n(X_{n+1}^j) I\{Y_{n+1}^j = i\}).
\end{aligned}$$

That is, we update several components at once. This will speed up convergence at the expense of increased per iterate computation.

4 Numerical Experiments

In this section we simulate the algorithm for different number of nodes. The results for the cases when the number of nodes are 50, 200 and 500 are plotted in Figure 1, Figure 2 and Figure 3 resectively. The dotted line indicates the distance between z^* and z_n w.r.t. n. The solid line indicates the percentage of top 5 indices of z^* that do not feature in the top 10 indices of z_n. Figure 4, Figure 5 and Figure 6 further show (for 200 nodes) that the number of iterations required to achieve this objective varies inversely with variance of z^*.

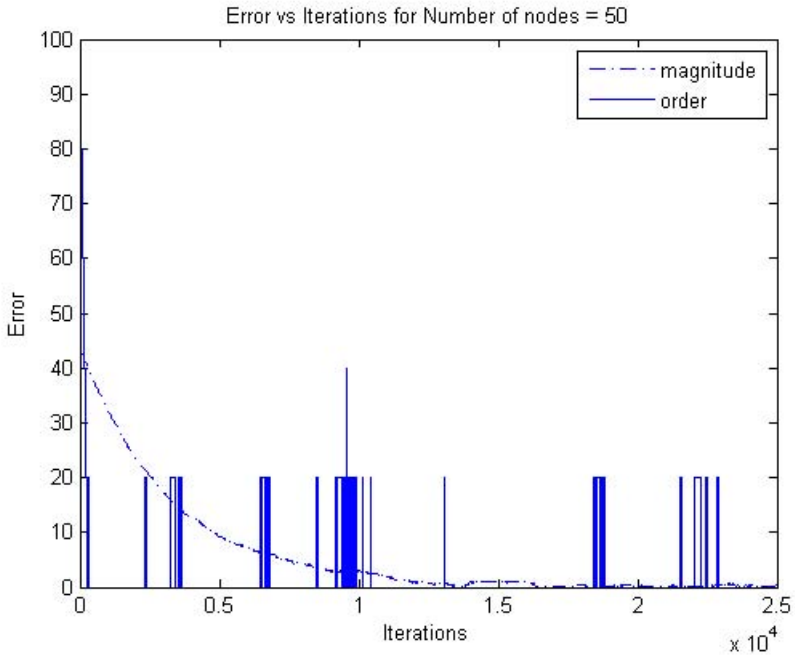

Fig. 1. Varaince of z^*=47.1641

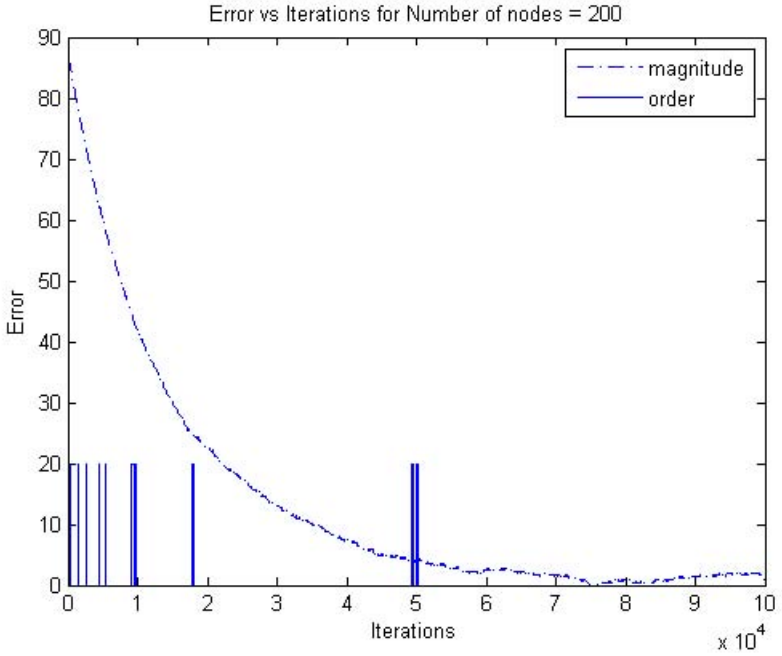

Fig. 2. Variance of z^*=277.3392

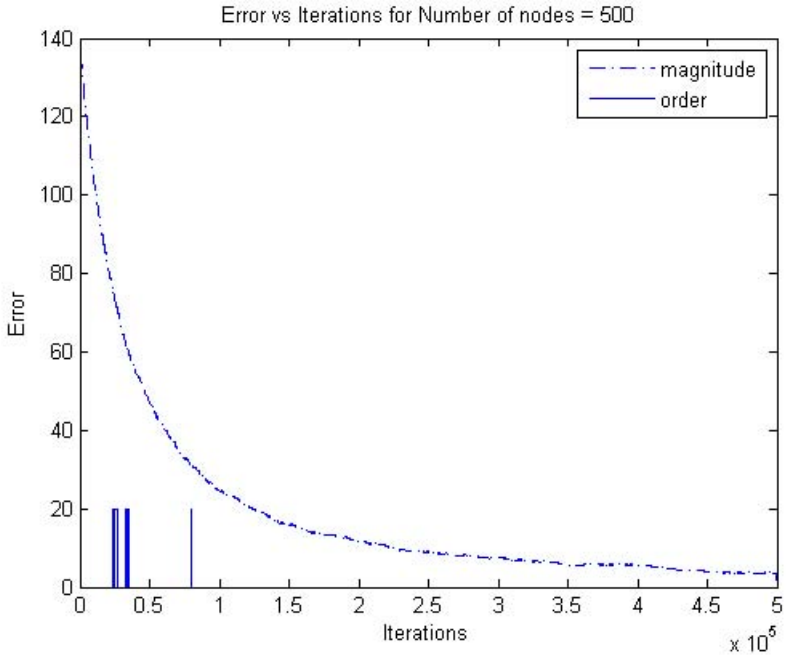

Fig. 3. Variance of $z^* = 743.4651$

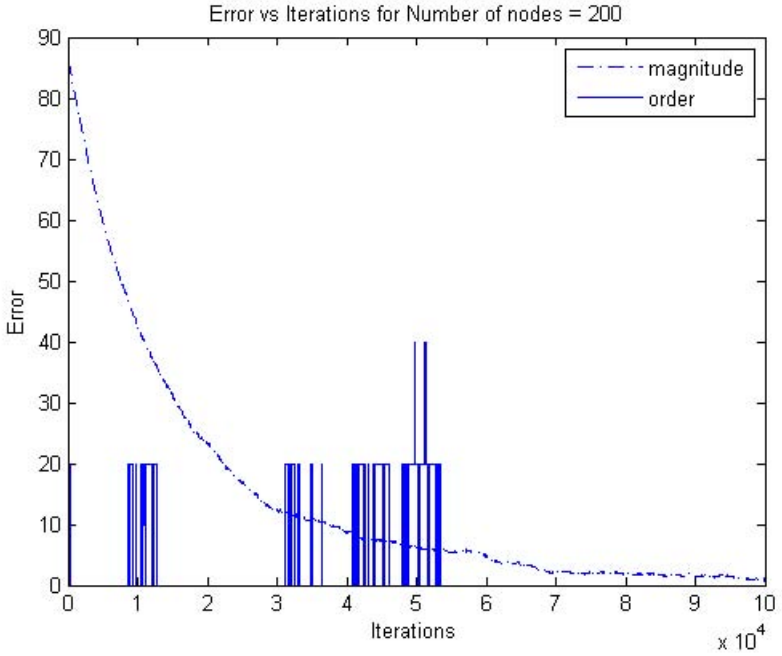

Fig. 4. Varaince of z^*=259.6187

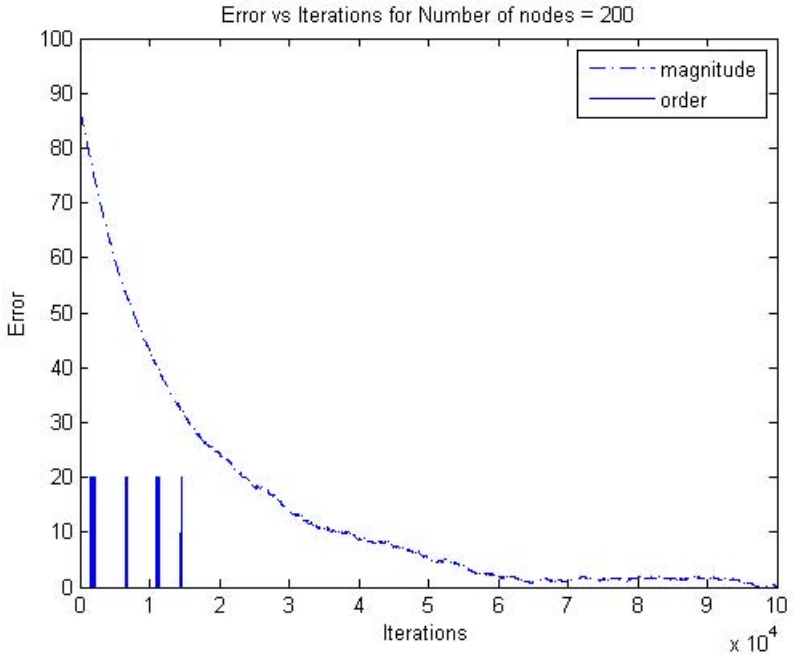

Fig. 5. Variance of z^*=335.6385

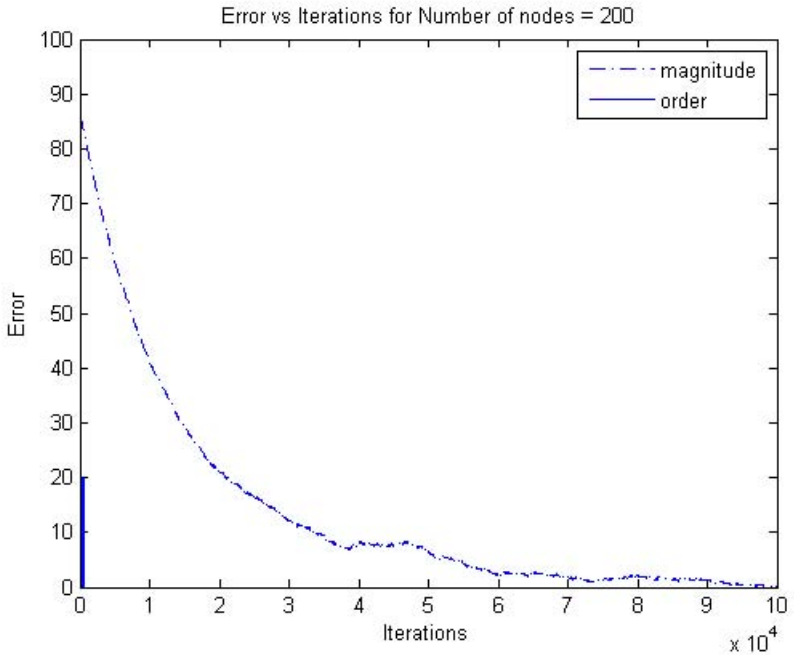

Fig. 6. Variance of $z^* = 365.0774$

5 Conclusions

In conclusion, we highlight some of the important features of the above scheme, which are facilitated by the reinforcement learning framework.

1. As already mentioned, the scheme does not depend on an a priori model for transition probabilities, but is completely data-driven in this aspect.
2. We use 'split sampling' introduced in [14] for reinforcement learning, sampling pairs (X_n, Y_n) with the desired conditional law for Y_n given X_n, but with uniform sampling for $\{X_n\}$. This is a departure from classical reinforcement learning, where one runs a single Markov chain $\{X_n\}$ according to $\hat{P}$ and $Y_n = X_{n+1}$.
3. Since we are iterating over probability vectors as they evolve under a transition matrix, the scheme requires *left-multiplication* by row vectors thereof. This is different from usual reinforcement learning schemes, which involve averaging with respect to the transition probabilities, i.e., *right-multiplication* by a column vector. We have worked around this difficulty by modifying the update rule. In classical reinforcement learning algorithms based on a simulated Markov chain $\{X_n\}$, one updates the X_nth component at time n, i.e., the ith component gets updated only when $X_n = i$. In the above scheme, the ith component gets updated both when $X_{n+1} = i$ and when $Y_{n+1} = i$, albeit in different ways. This is another novel feature of the present scheme.

References

1. Thorndike, E.L.: Animal intelligence: an experimental study of the associative processes in animals. Psychological Review, Monograph Supplement 2(8) (1998)
2. Bush, R.R., Mosteller, F.: A mathematical model of simple learning. Psychological Review 58, 313–323
3. Estes, K.W.: Towards a statistical theory of learning. Psychological Review 57, 94–107
4. Bertsekas, D.P.: Dynamic Programming and Optimal Control, 4th edn., vol. 2. Athena Scientific, Belmont (2007)
5. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic Programming. Athena Scientific, Belmont (1996)
6. Gosavi, A.: Simulation-based Optimization, Parametric Optimization Techniques and Reinforcement Learning. Springer, New York (2003)
7. Powell, W.B.: Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd edn. Wiley, New York (2011)
8. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
9. Szepesvari, C.: Algorithms for Reinforcement Learning. Morgan and Claypool Publishers (2010)
10. Sargent, T.J.: Bounded Rationality in Macroeconomics. Oxford Uni. Press, Oxford (1994)
11. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
12. Robbins, H., Monro, J.: A stochastic approximation method. Annals of Math. Stat. 22, 400–407 (1951)
13. Borkar, V.S., Makhijani, R., Sundaresan, R.: How to gossip if you must (preprint, 2013), `http://arxiv.org/abs/1309.7841`
14. Borkar, V.: Reinforcement Learning - A Bridge between Numerical Methods and Markov Chain Monte Carlo. In: Sastry, N.S.N., Rajeev, B., Delampady, M., Rao, T.S.S.R.K. (eds.) Perspectives in Mathematical Sciences. World Scientific (2008)
15. Langville, A.N., Meyer, C.D.: Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton Uni. Press, Princeton (2006)
16. Langville, A.N., Meyer, C.D.: Deeper inside PageRank. Internet Mathematics 1(3), 335–380 (2004)
17. Avrachenkov, K., Litvak, N., Nemirovsky, D., Osipova, N.: Monte Carlo methods in PageRank computation: when one iteration is sufficient. SIAM J. Numer. Anal. 45(2), 890–904 (2007)
18. Avrachenkov, K., Litvak, N., Nemirovsky, D., Smirnova, E., Sokol, M.: Quick detection of top-k personalized PageRank lists. In: Frieze, A., Horn, P., Prałat, P. (eds.) WAW 2011. LNCS, vol. 6732, pp. 50–61. Springer, Heidelberg (2011)
19. Polyak, B.T., Timonina, A.V.: PageRank: new regularizations and simulation models. In: Proc. of 11th IFAC World Congress, Milano, August 28-September, pp. 11202–11207 (2011)
20. Ishii, H.: Distributed randomized algorithms for PageRank computation. IEEE Trans. Auto. Control 55(9), 1987–2002 (2010)
21. Nazin, A.V., Polyak, B.T.: 'The randomized algorithm for finding an eigenvector of the stochastic matrix with application to PageRank. Doklady Mathematics 79(3), 424–427 (2009)
22. Zhao, W., Chen, H-F. and Fang, H-T.: Convergence of distributed randomized PageRank algorithms. arXiv:1305.3178 [cs.SY] (2013)
23. Vigna, S.: Spectral ranking, `http://arxiv.org/abs/0912.0238`
24. Borkar, V.S.: Stochastic Approximation: A Dynamical Systems Viewpoint. Hindustan Publ. Agency, Cambridge Uni. Press, New Delhi, Cambridge (2008)
25. Ho, Y.-C.: An explanation of ordinal optimization: Soft computing for hard problems. Information Sciences 113(3-4), 169–192 (1999)