Gossip With Transmission Constraints

V. S. Borkar and R. Makhijani
Department of Electrical Engineering
IIT Bombay
Powai, Mumbai 400076
India
Email: {borkar.vs,rahulmakhijani19}@gmail.com

R. Sundaresan
Department of Electrical Communication Engineering
Indian Institute of Science
Bangalore 560012
India
Email: rajeshs@ece.iisc.ernet.in

***Abstract*—We consider an asynchronous stochastic approximation version of the classical gossip algorithm wherein the inter-processor communication is subject to transmission delays. We highlight some fundamental difficulties associated with it and suggest an alternative scheme based on reinforcement learning.**

I. Plain vanilla gossip

Gossip algorithm [10] is a popular paradigm for averaging schemes across a group of communicating processors, e.g., in sensor networks. These are successive averaging schemes of the form

$$x(n+1) = Px(n),\ n \geq 0, \tag{1}$$

which compute successive averages $x(n+1) \in \mathcal{R}^d$ of the previous iterate $x(n) \in \mathcal{R}^d$ with respect to a stochastic matrix $P = [[p(i,j)]] \in \mathcal{R}^{d\times d}$, beginning with a prescribed initial data vector $x(0) = [x_1(0), \cdots, x_d(0)]^T \in \mathcal{R}^d$. If P is irreducible, this leads to the convergence

$$x(n) \to \sum_{i=1}^{d} \eta_i x_i(0),\ \text{as } n \to \infty$$

where $\eta = [\eta_1, \cdots, \eta_d]^T$ is the unique stationary distribution for P. An 'incremental' version

$$x(n+1) = (1-a)x(n) + aPx(n),\ n \geq 0,$$

has been proposed and analyzed as a model of opinion formation in societies [7], where $a \in (0,1]$ is a parameter that modulates the emphasis put on others' opinions, as opposed to one's own evaluation.

II. Enter stochastic approximation

In engineering applications, one often considers a *stochastic approximation* [4] version wherein at each time n, the processor (or 'agent') i polls a neighbor j according to probability $p(i,j)$ and 'pulls' the latter's data $x_j(n)$ for averaging. The recursion then is

$$x_i(n+1) = (1-a)x_i(n) + ax_{\xi_i(n)}(n),\ 1 \leq i \leq d,\ n \geq 0, \tag{2}$$

where $\xi_i(n)$ is generated with probability $p(i,\cdot)$ independently of all other random variables realized till time n. By adding and subtracting the one step conditional expectation of the last term, (2) can be written as

$$x(n+1) = (1-a)x(n) + a\Big(Px(n) + M(n+1)\Big),\ n \geq 0,$$

where $\{M(n)\}$ is an appropriately defined martingale difference sequence. This then becomes an instance of the 'constant step-size' version of the classical Robbins-Monro scheme for stochastic approximation:

$$x(n+1) = x(n) + a\Big(h(x(n)) + M(n+1)\Big),\ n \geq 0, \tag{3}$$

for a Lipschitz $h : \mathcal{R}^d \mapsto \mathcal{R}^d$. Under reasonable conditions (see [4, Ch.9]), the iterate in (3) tracks the asymptotic behavior of its limiting ordinary differential equation (in a sense that is made precise in [4, Ch.2])

$$\dot{x}(t) = h(x(t)),\ t \geq 0, \tag{4}$$

which for us is the linear system

$$\dot{x}(t) = (P-I)x(t),\ t \geq 0. \tag{5}$$

Here $I \in \mathcal{R}^{d\times d}$ is the identity matrix. Since P is stochastic, $\mathbf{1} := [1, 1, \cdots, 1]^T$ is the right Perron-Frobenius eigenvector of P. It is easy to see that $x(t)$ converges to $(\eta^T x(0))\mathbf{1}$, which depends on the initial condition $x(0)$. The convergence rate of this (hence of (3)) as well as the convergence rate of the original discrete scheme (1) are dictated by the eigenvalue of P with the second highest absolute value. We shall refer to this as the 'second eigenvalue' henceforth. This has prompted a lot of analysis and algorithms for minimizing the second eigenvalue, *ipso facto* maximizing the rate of convergence [6], [10].

The stochastic approximation version already introduces 'noise', as we are replacing an averaging operation by a sample picked according to the averaging probability weights. An additional complication arises when the implementation is *asynchronous* wherein,

- not all components of $x(n)$ are updated concurrently, and,
- the values computed at different processors are received at other processors with random inter-processor communication delays.

This situation is often brought about by transmission constraints such as those imposed by the wireless medium that disallow the co-occurrence of transmission of messages across certain edges. This leads to several nontrivial complications not present in the deterministic versions (1) or (5). Even though (1) and (5) converge to a unique limit for any initial condition $x(0)$, the same may not be true of the stochastic

case (3). If we use a decreasing step-size schedule $\{a(n)\}$ with $\sum_n a(n) = \infty$, $\sum_n a(n)^2 < \infty$, we observe a.s. convergence of the projected scheme to a random multiple of the eigenvector $\mathbf{1}$, not necessarily the desired one. This convergence is in fact suggested by the results of [8], established under somewhat more restrictive conditions. If we use a constant stepsize, stability of the iterations (2) can also become an issue, since constant stepsize algorithms are not provably a priori bounded a.s. In the special case here, they were found to converge, presumably because the martingale noise $\{M(n)\}$ also scales down to zero with $\min_c \|x(n) - c\mathbf{1}\|$. Constant stepsize schemes have higher fluctuations in general; so we introduce a parallel averaging scheme at each node as follows in order to reduce variance:

$$z(n+1) = z(n) + \frac{1}{n+1}\left(x(n+1) - z(n)\right), \ n \geq 0. \quad (6)$$

This leads to graceful convergence. Nevertheless, the convergence can be at a point other than the desired one. We also consider the case of 'noisy measurements' wherein $x_{\xi_i(n)}(n)$ above gets replaced by $x_{\xi_i(n)}(n) + W(n+1)$ for some i.i.d. zero mean noise $\{W(n)\}$. In this case, the constant stepsize scheme does not even converge. See lack of convergence in the noisy case in Figure 2 and compare with convergence to consensus value (maximum error from consensus value approaches zero) in Figure 1. We now describe the set up used for these simulations.

Simulation description: The simulation was done on a single randomly generated Erdös-Rényi graph of 100 nodes. The probability of an edge between a pair of nodes was 0.2. The P matrix is symmetric with each entry in the upper triangle having independent and uniform distribution over $[0, 1]$ whenever a link exists between the corresponding nodes. The nodes were initialized with $x(0)$ generated with independent and uniform distribution over $[0, 1]$. The plotted errors are the supremum norms of errors $x(n) - (\eta^T x(0))\mathbf{1}$ and $z(n) - (\eta^T x(0))\mathbf{1}$. Figure 1 is a plot of the iterates where the data is received noiselessly. The updates were asynchronous, and the update rates differed across nodes. (Average inter-update time for node j was $10 + j$ time steps.) This difference was introduced in order to simulate the effect of activation set transmission constraints. The total number of time steps simulated were 120,000. Figure 2 considers the case when the data is corrupted by additive white Gaussian noise of variance 0.25. This yields a per node average signal power to noise power ratio of 1.

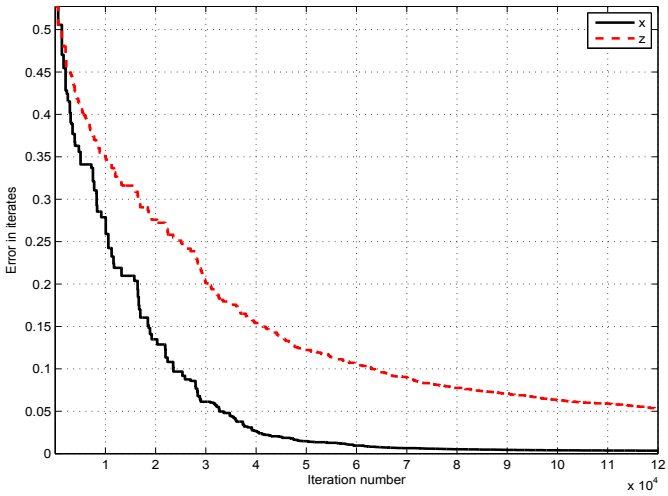

Fig. 1. Residual error in iterates over time. Data is noiselessly received.

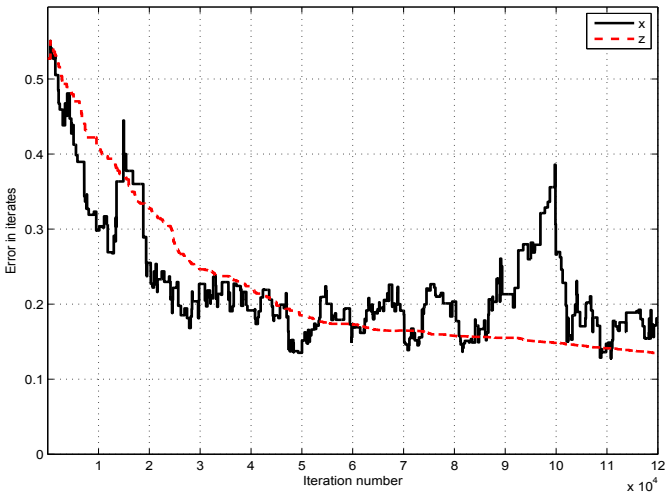

Fig. 2. Residual error in iterates over time. Data is received with noise.

III. A REINFORCEMENT LEARNING TWIST

The foregoing discussion prompts us to consider a different scheme motivated by the reinforcement learning algorithms for average cost problem [1]. This is based on the Poisson equation

$$V = PV + x(0) - \beta, \quad (7)$$

This is to be solved for the pair $V(\cdot) \in \mathcal{R}^d, \beta \in \mathcal{R}$. Under an irreducibility hypothesis on P, it has a solution $(V(\cdot), \beta)$. Here V is specified uniquely modulo an additive scalar constant, whereas β is characterized uniquely as the optimal cost: $\beta = \eta^T x(0)$, where once again η is the stationary distribution for P. Furthermore, these can be computed by the iterative scheme, dubbed *relative value iteration*, given by:

$$V(n+1) = PV(n) + x(0) - V_{i_0}(n)\mathbf{1}, \ n \geq 0,$$

where i_0 is a fixed state. It can be shown that $V(n) \to V^*, V_{i_0}(n) \to \beta$, where V^* is the unique solution of (7) satisfying $V^*_{i_0} = \beta$. See [9] for these and related facts for the more complicated 'controlled' P.

A stochastic approximation version of the above can be given along the lines of Abounadi et al. [1], as

$$y_i(n+1) = (1-a)y_i(n) + a\left(y_{\xi_i(n)}(n) - x_i(0) - y_{i_0}(n)\right). \quad (8)$$

It is proved in [1, Th.3.5] that $y(n) \to V^*, y_{i_0}(n) \to \beta$, a.s.

As pointed out in [1], we can replace $y_{i_0}(n)$ above by $f(y(n))$ for any $f : \mathcal{R} \mapsto \mathcal{R}$, satisfying $f(\mathbf{1}) = 1$ and $f(\mathbf{x} + c\mathbf{1}) = f(\mathbf{x}) + c$ for $c \in \mathcal{R}$.

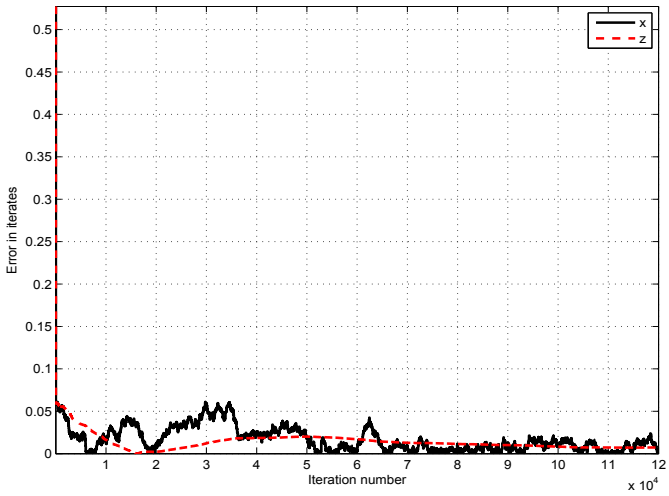

Fig. 3. Residual error in reinforcement-learning iterates over time. Data is received with noise.

Our numerical experiments showed significant improvement in error when (8) was used in place of (2) and (6). What's more, the additional measurement noise $\{W(n)\}$ introduced above does not affect the conclusions. Figure 3 depicts the error in iterates (8). The setting is the same as the setting in Figure 2 (with noise).

One important issue with (8) is that the i_0th component of the iteration has to be broadcast to all nodes. Alternatively, we can replace it by a suitable weighted average of the $x_i(n)$'s that is computed in a distributed manner on a faster time scale, as in [5].

IV. Additional possibilities

Some further possibilities are as follows.

A. Conditional importance sampling

In the asynchronous case, there is an additional error term due to delays, over and above the errors due to discretization and noise. This can be shown to be bounded by a term proportional to a and any bound on the mean delays $\{E[\tau_{ji}(m)]\}$, where $\tau_{ji}(n)$ is the delay with which node i received the value at node j at time n. This is not surprising, because on the algorithm's time scale, in τ steps, each component would have changed by an amount that is $O(\tau)$. This suggests favoring low values of $E[\tau_{ji}(n)]$. Since this will be inversely proportional to the frequency with which i polls j, this suggests sampling with a different polling matrix

$$Q := [[q(i,j)]] \text{ with } q(i,j) > 0 \Longleftrightarrow p(i,j) > 0,$$

and compensating for it by inserting the appropriate likelihood ratio correction as in [2]. Thus we replace (2), (6) by

$$\begin{aligned} x_i(n+1) \;=\; & (1 - a(n+1,i))x_i(n) \\ & + a(n+1,i)\left(\frac{p(i,\xi_i(n))}{q(i,\xi_i(n))}\right)\cdot x_{\xi_i(n)}(n), \\ & 1 \le i \le d,\ n \ge 0, \end{aligned} \tag{9}$$

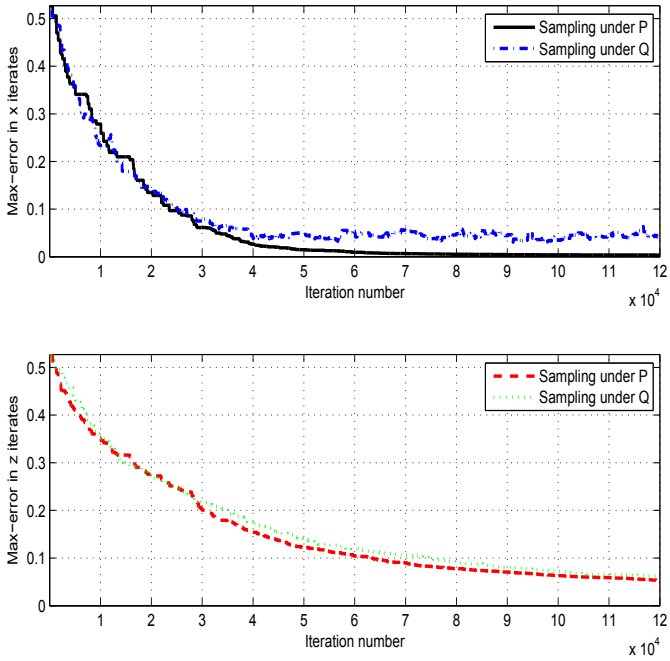

Fig. 4. Comparison of residual error under normal P sampling and under importance sampling (Q) over time. Data is noiselessly received.

$$\begin{aligned} z_i(n+1) \;=\; & z_i(n) + \frac{1}{n+1}\left(x_i(n+1) - z_i(n)\right), \\ & n \ge 0, \end{aligned} \tag{10}$$

where the stepsizes are judiciously chosen to compensate for any differences in the relative rates of updates across vertices. (We omit the details of this compensation, but refer the reader to [4, Sec.9.3.6]). In view of (9), the mean error due to delay in i receiving j's value is weighted by $q(i,j)\times\left(\frac{p(i,j)}{q(i,j)}\right) = p(i,j)$. The foregoing suggests choosing Q to minimize

$$\sum_i \sum_{j\in\mathcal{N}(i)} \frac{p(i,j)}{q(i,j)}, \tag{11}$$

subject to the constraints $\sum_j q(i,j) = 1$ for every i and $q(i,j) \ge 0$ for every (i,j). It is easy to see that the optimal Q is given by

$$q(i,j) = \frac{\sqrt{p(i,j)}}{\sum_k \sqrt{p(i,k)}}.$$

We tried this scheme as well, but the improvement for moderate sized problems was negligible, suggesting that the $O(a)$ bound on delay errors is pessimistic. See Figures 4 and 5 for plots without and with noise, respectively, in the received data. The simulation settings are the same as in Figures 1 and 2, respectively.

B. A friend's friend is also a friend: multihop

A further possibility to modulate the above scheme would be to use multihop. Consider, for example, the possibility of two hops. Let $j \in \mathcal{V}$ be a neighbor of $i \in \mathcal{V}$ and $k_1, k_2, \cdots, k_m \in \mathcal{V}$ neighbors of j that are not neighbors of

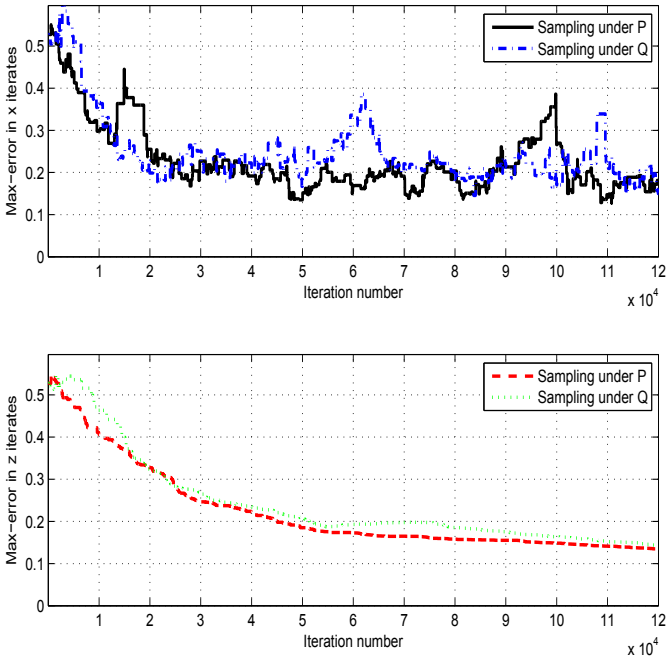

Fig. 5. Comparison of residual error under normal P sampling and under importance sampling (Q) over time. Data is received with noise.

i. Then each time i polls j, i may either pull the current value at j, or pull the value at some k_ℓ that has been already pulled and stored by j. Suppose the former is done with probability $p^0(i,j)$ and the latter with probability $p^\ell(i,j)$. Then we must have $\sum_{\ell=0}^{m} p^\ell(i,j) = p(i,j)$. Note that $p(i,j)$ now is the probability with which i polls j, but it is no longer the weight which it assigns to the values pulled from j. This is tantamount to replacing the original P by a modified Q with additional edges from i to the k_ℓs with weights $p^\ell(i,j)$ resp., and replacing the weight $p(i,j)$ of edge (i,j) by $p^0(i,j)$. There are, however, tradeoffs involved. For one, i is actually sampling j's value at a lower rate, thereby increasing the associated mean delay. Note also that the delay associated with i's 'virtual' sampling of k_l will be a combination of delays due to its sampling of j and j's sampling of k_r. The benefit however is that the network is better connected.

We also need to choose the new sampling probabilities so as to retain the stationary distribution as η, since our focus is on averaging with respect to η, not merely on obtaining a consensus. Some constraints suggest themselves, e.g., if $p(i,j,k)$ is the fraction of times i polls j in order to pull its stored value from k, then $p(j,k) \geq p(i,j,k)$ so as to avoid pulling the same value often. The tradeoffs and optimal choice of the parameters $p^\ell(i,j)$ are items for further study.

ACKNOWLEDGMENT

Research of V. S. Borkar was supported in part by a J. C. Bose Fellowship and a grant "Distributed Computation for Optimization over Large Networks and High Dimensional Data Analysis" from Department of Science and Technology, Government of India. R. Sundaresan was supported by the Indo-US Science and Technology Forum Fellowship and by the US National Science Foundation under grant CCF-1017430. This author thanks the Coordinated Sciences Laboratory, University of Illinois at Urbana-Champaign, for its hospitality during the course of this work.

REFERENCES

[1] Abounadi, J.; Bertsekas, D. P. and Borkar, V. S.; "Learning algorithms for Markov decision processes with average cost", *SIAM J. Control Optim.* 40(3), 681-698, 2001.

[2] Ahamed, T. P. I.; Borkar, V. S. and Juneja, S. K.; "Adaptive importance sampling technique for Markov chains using stochastic approximation", *Operations Research*, vol. 54(3), pp. 489-504, 2006.

[3] Bertsekas, D. P. and Tsitsiklis, J. N.; *Parallel and Distributed Computation: Numerical Methods*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[4] Borkar, V. S.; *Stochastic Approximation: A Dynamical Systems Viewpoint*, New Delhi: Hindustan Publ. Agency, and Cambridge, UK: Cambridge Uni. Press, 2008.

[5] Borkar, V. S. and Makhijani, R., "Who is the fairest of them all?", *Proc. 50th Allerton Conf. on Communication, Control and Computing*, Sep. 29 - Oct. 3, 2012, Monticello, Ill.

[6] Boyd, S.; Diaconis, P. and Xiao, L., "Fastest mixing Markov chains on a graph", *SIAM Review*, vol. 46(4), pp. 667-689, 2004.

[7] DeGroot, M.; "Reaching a consensus", *J. American Stat. Assoc.*, vol. 69, pp. 118-121, 1974.

[8] Huang, M.; "Stochastic approximation for consensus: a new approach via ergodic backward products", *IEEE Trans. on Automatic Control* 57(12), 2994-3008, 2012.

[9] Puterman, M. I.; *Markov Decision Processes*, Wiley-Interscience, New York, 2005.

[10] Shah, D.; "Gossip algorithms", *Foundations and Trends in Networking*, Vol. 3(1), pp. 1-125, 2008.