Distributed Consensus and Optimization under Communication Delays

Konstantinos I. Tsianos
Department of Electrical and
Computer Engineering
McGill University
Montreal, Quebec H3A 2A7
Email: konstantinos.tsianos@gmail.com

Michael G. Rabbat
Department of Electrical and
Computer Engineering
McGill University
Montreal, Quebec H3A 2A7
Email: michael.rabbat@mcgill.ca

***Abstract*—We study the effects of communication delays in distributed consensus and optimization algorithms. We propose two ways to model delays. First, assuming each edge of a communication network has a fixed delay, we characterize the consensus value exactly as a function of the delays and edge weights and obtain convergence rate bounds using results from non-reversible Markov chains. Second, we propose a novel way to model random delays per edge. Our model allows the reception of multiple delayed messages from the same sender in the same time slot, a situation that can happen in practice. Both models admit a description of the consensus updates in the presence of delays via linear equations. Finally, we briefly discuss how to apply our delay models to analyze distributed optimization algorithms in the presence of delayed information.**

I. Introduction

In this paper we study the effects of communication delays in distributed consensus and optimization algorithms. We propose a fixed and a random delay model in discrete time for a communication network exchanging messages to reach consensus. Both models describe the consensus dynamics via linear update equations. Assuming a network with fixed (but not equal) edge delays, we characterize the consensus value exactly and bound the convergence rate using non-reversible Markov chain theory. Our random delay model is more general and allows the reception of multiple messages from the same sender in the same time slot, a scenario that happens in practice and is not captured by continuous time models. In the final part of the paper, we briefly discuss how our delay models can be used to understand the effect of delayed information on distributed optimization algorithms.

The problem of distributed computation over a network has recently received significant attention. For example, gossip and consensus algorithms [1], [2] appear as a key component in problems such as distributed linear parameter estimation, source localization, distributed compression and field estimation and many others. More recently, consensus algorithms have found a new application in the area of distributed optimization. For example, in large scale machine learning, a dataset is distributed across a network of processors and we typically try to optimize a cost function over all data to learn a model [3]–[5]. In these problems, the communication overhead can greatly affect the performance of a distributed algorithm. The network can incur communication delays such that nodes consistently receive outdated information from their neighbours. Just as a simple example, consider a network with 1 Gigabit per second ethernet. For a small machine learning problem we may need to send messages of size 1Mbyte per iteration which translates to a transmission delay of 8 milliseconds per message. For a modern processor using some fast local optimization routine (e.g. stochastic gradient descent [6]), 8 milliseconds is enough time to perform multiple iterations, and the time it takes to exchange information over the network is not negligible.

There exist various studies of the effects of communication delays on consensus algorithms in the literature. For applications in partial differential equations, distributed control and multi-agent coordination we can find a lot of continuous time models. See for example [7], [8] and [9], [10] which analyze a simplified variation where all edges experience the same constant delay. Our motivation comes from applications in distributed optimization where both computation and communication happen in rounds and take a significant amount of time. For this reason we focus on discrete time models which are more suitable. An early treatment of delays in discrete time consensus can be found in [11], where it is proven that convergence is not guaranteed if delays are unbounded. An analysis of conditions for convergence in the presence of delays is given in [12]. Closer to our work are [13] and [14] which model delays in discrete time for consensus problems by augmenting the state space with delay nodes. However, none of these models allows for receiving multiple messages from one sender in the same time slot, a situation that can occur when assuming bounded delays in discrete time, as we explain below. Moreover, the value at which the consensus algorithm asymptotically converges to is not characterized. Finally, the convergence rate bound in [14] is loose in many situations of practical interest.

The main contribution of this paper is the description of two models for analyzing discrete-time distributed consensus algorithms in the presence of communication delays. The first model covers the case where each directed edge experiences a randomly chosen but fixed delay throughout execution of the consensus algorithm. The second model allows for random

bounded delays per edge and is to the best of our knowledge novel. Our random delay model is very general allowing the reception of multiple messages from the same sender in one iteration. For both models we explicitly show how to describe the consensus dynamics as a set of linear update equations. We can thus exploit known results for non-reversible Markov chains to bound the convergence rate in the presence of delays. In the case of fixed delays we characterize the consensus value exactly and show that for max-degree weights the stationary distribution only depends on the total amount of delay and not how the delay is distributed across the edges. We also show that it is possible to solve a network optimization problem to mitigate the effect of delays. Finally, we briefly discuss how to use our models to understand the effects of delayed communication in the context of distributed optimization.

The rest of the paper is organized as follows. Section II defines the consensus problem and introduces some of the notation used throughout the paper. We describe and analyze our two delay models in Sections III, IV and V, while Section VI employs those models in the context of distributed optimization. The paper concludes in Section VII with a summary and a listing of possible future research directions.

II. Distributed Consensus

Let us consider a network $G = (V, E)$ of n nodes. Assuming each node $i \in V$ holds a value $x_i(0)$ at time 0, the consensus problem asks for an algorithm such that the nodes exchange messages over the links in E to eventually reach a consensus, i.e., $[x_1(k)\ x_2(k)\ \cdots\ x_n(k)]^T \to v\mathbf{1}$ as $k \to \infty$. Generally v is a function of the initial values. An extremely popular variant demands that all nodes converge to the average [15]. For the purposes of this paper it is enough that the nodes reach an agreement on a value that is just a weighted average. We will consider distributed protocols where at each iteration each node computes a convex combination of its own current value and the values received by its neighbours; i.e.,

$$x_i(k+1) = \sum_{j=1}^{n} p_{ij} x_j(k). \tag{1}$$

We assume that $p_{ij} > 0$ only when $(i, j) \in E$ so that the matrix $P = [p_{ij}]$ describes the connectivity structure of G. To form convex combinations of incoming messages, we require $\sum_{j=1}^{n} q_{ij} = 1$. Thus, P is a (row) stochastic matrix describing a Markov chain. If, in addition, the Markov chain P is irreducible and aperiodic, protocol (1) solves the consensus problem [9]. Without further assumptions, P describes a generally non-reversible Markov chain whose stationary distribution is not uniform. For the rest of the paper we assume that the delay-free transition matrix P is doubly stochastic; i.e., its columns also sum to one ($\mathbf{1}^T P = \mathbf{1}^T$). In that case it can be shown that $P^k \to \frac{1}{n}\mathbf{1}\mathbf{1}^T$ as $k \to \infty$. We now proceed to model and analyze the case where messages are delivered with time delay.

III. Time Delayed Communication Models

Equation (1) updates the network state in discrete time steps. We say that a message from node i to node j is delayed by b if it is received b time steps after it has been sent. For simplicity, let us first assume that each directed edge (i, j) experiences a fixed delay b_{ij} in the sense that each message leaving node i takes b_{ij} iterations to reach j. This suggests linear update consensus equations of the form

$$x_i(k+1) = \sum_{j=1}^{n} p_{ij} x_j(k - b_{ij}) \tag{2}$$

where $0 \le b_{ij} \le B$ for a network with bounded maximum delay B. This model describes a system with *fixed delays* and is analyzed in the following section. Each b_{ij} can be thought of as the average delay experienced on that link.

A more realistic model relaxes the fixed delay assumption. Assuming there is still a maximum delay bound B, we allow for random time-varying delays sampled in $\{0, \ldots, B\}$ every time a new message is sent. Under this *random delay* model analysis is more complicated since in one iteration a node can receive multiple messages from the same sender. This situation is analyzed in Section V. For both models we assume that there is no delay in self loop messages; i.e., each node always has access to its most recent local estimate. As we will see, for both models we devise a formulation to describe the consensus update equations with a new matrix Q that is constructed from the initial matrix P. The construction involves an augmentation of the communication graph G with delay nodes. The idea itself is not new and appears in other work such as [13] and [14]. However our formulation is significantly different both in terms of the number of delay nodes we introduce as well as the equations it leads to. In our model, the intuition is that for every unit of delay on an edge we force a message to pass through an extra intermediate node before reaching its destination.

To understand the effect of delays in distributed consensus processes we ask the following questions:

- How do we construct the matrix Q by adding delays to a network initially described by P?
- What is the stationary distribution π of Q?
- At what rate does Q converge to π?

IV. Fixed Delay Model

As mentioned earlier, without delays we have a communication network G described by a doubly stochastic matrix P. By adding delays nodes we augment G so that it is described by a row stochastic matrix Q. We develop a procedure using simple matrix operations to build Q from P by inserting delays on edges one at a time. Suppose after some delay insertions we have a matrix Q and want to add a delay of b_{ij} on edge (i, j). We replace edge (i, j) by a delay chain $d_1, d_2, \ldots, d_{b_{ij}}$ and re-route all messages from i to j through that chain (see Figure 1). We define an $n \times n$ matrix M_1 responsible for setting to 0 the entry of Q corresponding to sending information from i to j without delay. Instead, i sends its message to the first

delay node d_1. The weight of the message is the same as the one that would be used to send from i to j directly without delay; i.e., $q_{ij} = p_{ij}$. This is achieved by an $n \times b_{ij}$ matrix M_2. A $b_{ij} \times n$ matrix M_3 delivers the message to j from the last delay node in the chain $d_{b_{ij}}$ and we use a $b_{ij} \times b_{ij}$ matrix M_4 to pass messages along the delay chain. We use e_i to denote the i-th column of the $n \times n$ identity matrix and g_i to denote the i-th element of the $b_{ij} \times b_{ij}$ identity matrix. With these definitions, we initialize $Q = P$. To add b_{ij} delay nodes on the edge $i \to j$ we apply the transformation

$$Q = \begin{bmatrix} I_n \\ \mathbf{0}_{b_{ij} \times n} \end{bmatrix} Q \begin{bmatrix} I_n & \mathbf{0}_{n \times b_{ij}} \end{bmatrix} + \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \tag{3}$$

$$M_1 = -e_i e_i^T P e_j e_j^T \tag{4}$$

$$M_2 = -M_1 e_j g_1^T \tag{5}$$

$$M_3 = g_{b_{ij}} e_j^T \tag{6}$$

$$M_4 = \sum_{k=1}^{n-1} g_k g_{k+1}^T. \tag{7}$$

Example: Consider a very simple chain graph G with 3 nodes. Without delays we define

$$P = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{2}{3} \end{bmatrix}. \tag{8}$$

To model a fixed delay of 2 whenever node 1 transmits to node 2, we consider the graph G as directed and augment it with two delay nodes d_1, d_2 so that information from 1 to 2 always goes through them (see Figure 1). The augmented graph is described by a stochastic matrix Q:

$$Q = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ d_1 \\ d_2 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & d_1 & d_2 \end{array} \\ \begin{bmatrix} \frac{2}{3} & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}. \tag{9}$$

Node 1 sends messages to the first delay node, scaled by the amount $q_{12} = p_{12}$. After that, all delay nodes just forward information until the destination node 2 is reached. Q represents a Markov chain with a stationary distribution that is not uniform. In the next two subsections we compute the stationary distribution exactly.

A. Stationary Distribution - Max-Degree Weights P

A popular choice for achieving (average) consensus is to use a doubly stochastic P representing a max-degree random walk [16]:

$$P = I - \frac{D - A}{d_{max} + 1} \tag{10}$$

where A is the adjacency matrix of the (undirected) graph G, $D = diag(deg(1), \ldots, deg(n))$ has the node degrees in the diagonal and $d_{max} = \max_{i \in V} deg(i)$. As we show below, this P has an appealing property when adding delays: the stationary distribution is only affected by the total number of delay nodes on the network and not by the amount of delay on any particular edge.

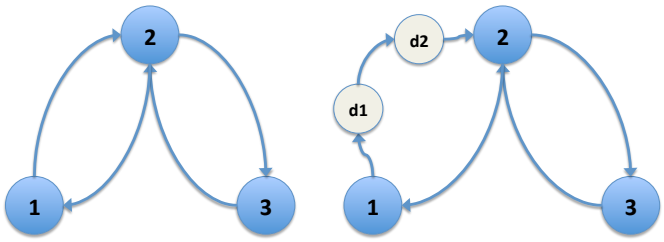

Fig. 1. (left) A network with 3 nodes. (right) The network when we add a delay of 2 on the edge $(1, 2)$.

For the max-degree transition matrix P, computing the stationary distribution is relatively simple. Take a network G with n nodes and $m = 2|E|$ edges and add $0 \leq b_r \leq B$ delay nodes on edge r where $r = 1, \ldots, m$ for a total of $\sum_{r=1}^{m} b_r = b$ delay nodes. We construct the delayed communication matrix Q as described earlier. The nodes in the graph with delays belong to either the set of the original computing nodes V, or the set of delay nodes – call it C. Matrix P is doubly stochastic, irreducible and aperiodic so a random walk on the original graph (only nodes in V) converges to the uniform distribution. Since the addition of delay nodes does not change the connectivity structure (degrees of nodes in V), a random walk on the augmented graph still converges to a stationary distribution where all nodes in V have equal probability. However, the probability of each node in V will be less than $\frac{1}{n}$ since part of the stationary mass has to go to the delay nodes. Moreover, the delay nodes only forward information so a random walk spends an equal amount of time on all nodes of the same delay chain. Finally, P assigns equal probability to transition out of a computing node to any of its neighbouring nodes. We thus expect that the nodes in C will also all have equal probability mass. In summary, we expect the stationary distribution of Q to be of the form $\pi = [\pi_V \mathbf{1}_n^T \quad \pi_C \mathbf{1}_b^T]^T$ where $\mathbf{1}_b$ is the vector of all ones of length b. To confirm this intuition, we compute the stationary distribution exactly by solving the balance equations:

$$Q^T \pi = \pi. \tag{11}$$

From this set of equations, we take an arbitrary one for a node $i \in V$ which has $0 \leq m_i \leq deg(i)$ outgoing edges to delay nodes. We have

$$\left(1 - \frac{m_i}{d_{max} + 1}\right) \pi_V + m_i \pi_C = \pi_V \tag{12}$$

which implies

$$\pi_V = (d_{max} + 1) \pi_C. \tag{13}$$

If we also demand that $\sum_{i=1}^{n+b} \pi_i = 1$, we obtain

$$\pi_V = \frac{d_{max}+1}{b+n(d_{max}+1)} \tag{14}$$

$$\pi_C = \frac{1}{b+n(d_{max}+1)}. \tag{15}$$

We can similarly verify that the rest of the stationarity equations are satisfied. Since the resulting π is a valid probability distribution and we know that the stationary distribution is unique, our intuition that at convergence we will only have two probability values was correct. However this is just a consequence of the fact that P assigns equal probability to all outgoing edges. Another important consequence is that for this particular P it does not matter how the delay is distributed on the edges. All that matters is the total number of delay nodes b. This situation is different if we consider P with general structure as we see next.

B. Stationary Distribution - General Case

In general, a delay free network is described by an aperiodic, irreducible, doubly stochastic matrix P which is not symmetric and does not necessarily assign equal weights to all outgoing edges from a node. Nevertheless, since P converges to the uniform distribution, we still expect the stationary probability mass to be equal for all computing nodes. We also expect the probability mass to be equal for delay nodes on the same edge but different between edges. We can still compute the stationary distribution by solving the balance equations. We assume again that the computing nodes have probability mass π_V. We have $m = 2|E|$ directed edges with $b_r, r = 1, \ldots, m$ delay nodes per edge. Denoting by $i(r)$ and $j(r)$ the two nodes of the original graph connected by edge r experiencing delay b_r, a delay chain C_r replaces the directed edge $(i(r), j(r))$ and $i(r)$ sends messages to the first delay node of C_r with weight $p_{i(r)j(r)}$. Letting all delay nodes on the same edge r have equal probability π_r, we are looking for a stationary distribution of the form

$$\pi = [\pi_V \mathbf{1}_n^T \;\; \pi_1 \mathbf{1}_{b_1}^T \;\; \cdots \;\; \pi_m \mathbf{1}_{b_m}^T]^T. \tag{16}$$

First we build the delay matrix Q as explained before. Looking closely, we observe that for each edge $(i(r), j(r)), r = 1, \ldots, m$ we have a row in Q^T whose only non-zero element is $p_{i(r)j(r)}$ at column $j(r)$. From $Q^T\pi = \pi$ for all such rows we obtain equations of the form

$$\pi_V p_{i(r)j(r)} = \pi_r. \tag{17}$$

Moreover, the elements of the stationary distribution π must sum to 1; i.e.,

$$\mathbf{1}^T \cdot \pi = 1 \quad \Rightarrow \quad n\pi_V + \sum_{r=1}^{m} b_r \pi_r = 1. \tag{18}$$

Substituting π_r from (17) to (18), we first compute π_V and then go back to (17) to get the π_r's:

$$\pi_V = \frac{1}{n + \sum_{r=1}^{m} b_r p_{i(r)j(r)}} \tag{19}$$

$$\pi_r = \frac{p_{i(r)j(r)}}{n + \sum_{r=1}^{m} b_r p_{i(r)j(r)}}. \tag{20}$$

One can easily verify that the rest of the equations in (18) are satisfied with the computed π. As we see, the stationary distribution depends both on the weight we use to send messages through every delay chain C_r and also on the length of the chain b_r.

C. Convergence Rate Bound under Fixed Delays

It is expected that the convergence rate of the delayed matrix Q will not be the same as the rate at which the original P converges. Characterizing the convergence rate of Q in relation to the convergence rate of P however is not obvious. Q is not symmetric and represents a non-reversible Markov chain. We can get a bound on the convergence rate using the result by Fill [17] which suggests using a reversibilization of the chain as a way for symmetrizing Q. For a Markov chain described by a stochastic matrix Q with a stationary distribution π, we compute the reverse Markov chain $\tilde{Q} = [\tilde{q}_{ij}]$ as

$$\tilde{q}_{ij} = \frac{\pi_j q_{ji}}{\pi_i}. \tag{21}$$

We then define the additive reversibilization of Q as

$$U(Q) = \frac{Q + \tilde{Q}}{2}. \tag{22}$$

The rate of convergence of Q is bounded using the spectral properties of $U(Q)$. Let us define $\lambda_2(\cdot)$ to return the second largest eigenvalue of a matrix and $\|\mathbf{x}-\mathbf{y}\|_{TV} = \frac{1}{2}\sum_{i=1}^{n}|x_i - y_i|$ to be the total variation distance between two discrete distributions $\mathbf{x}, \mathbf{y} \in [0,1]^n$. If Q is strongly aperiodic (meaning that $q_{ii} \geq \frac{1}{2}, \forall i \in V$) we have

$$\|Q(i,\cdot)^k - \pi^T\|_{TV}^2 \leq \frac{\lambda_2(U(Q))^k}{4\pi_i} \tag{23}$$

where and $Q(i,\cdot)$ is the i-th row of Q (Fill [17], Theorem 2.9 and Corollary 2.9)). This additive reversibilization bound does not apply directly in our case however since Q is not strongly aperiodic. To use Fill's result, we employ a standard trick and consider a lazy version $Q_{lazy} = \frac{1}{2}(I + Q)$ of Q. The latter is expected to converge slower than Q but is by definition strongly aperiodic and has the same stationary distribution as Q (i.e., $\pi_{lazy} = \pi$). Based on (24) we obtain

$$\|Q^t(i,\cdot) - \pi\|_{TV}^2 \leq \|Q_{lazy}^t(i,\cdot) - \pi_{lazy}\|_{TV}^2 \tag{24}$$

$$\leq \frac{(\lambda_2(U(Q_{lazy})))^t}{4\pi_{lazy}(i)}. \tag{25}$$

In Figure 5 we plot this bound in black and the actual total variation distance in red for a graph with 3 nodes and delay 3 on edge $(1,2)$. As we see the bound is of the right order of magnitude.

Optimizing P: Introducing delays lets us pose a matrix design question. Given fixed delays on the edges, what is the optimal P such that convergence of the delayed matrix Q to $\pi(Q)$ is as fast as possible? Allowing for an initial general doubly stochastic P leads to a stationary distribution for Q of the form

$$\pi = [\pi_V \mathbf{1}_n^T \quad \pi_1 \mathbf{1}_{b_1}^T \quad \ldots \pi_m \mathbf{1}_{b_m}^T]^T \tag{26}$$

for m delay edges with b_r delays nodes on edge r and $\sum_{r=1}^{m} b_r = b$. We are looking to maximize the convergence speed which is bounded by the second largest eigenvalue of $U(Q_{lazy})$. Specifically, we would like $\lambda_2(U(Q_{lazy}))$ to be as small as possible. Our constraints are that P remains doubly stochastic with positive entries, and that the topology of the network is not altered, i.e., the zeros of the initial P remain zero[1].

We write the optimization problem as follows:

$$\begin{aligned} minimize_P \quad & \lambda_2(U(Q_{lazy})) && (27)\\ subject\ to \quad & P\mathbf{1} = \mathbf{1}, P^T\mathbf{1} = \mathbf{1}, p_{ij} \geq 0 && (28)\\ & p_{ij} = 0 \ \text{ if } \ (i,j) \notin E && (29) \end{aligned}$$

where the first set of constraints forces P to be doubly stochastic with positive entries. The last constraints, says that every P considered by the optimization routine, must keep at zero all the entries for which the original P has zeros. It is not explicitly shown, but remember that the objective function $\lambda_2(U(Q_{lazy}))$ is a function of P.

To verify that solving the above optimization problem can yield meaningful results, we tried to solve the following very simple example numerically. We consider a network of 3 nodes with all edges included:

$$P = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}. \qquad (30)$$

Figure 2 shows the total variation distance of P^k, Q^k and Q^k_{opt} when P has been optimized as k increases. The initial P has a total variation distance of 0 since the graph it represents is complete. If we add a delay of 3 on edge $(1,2)$, then Q takes about 16 iterations to converge as shown by the red curve. Next, we use our optimization routine, to get a new matrix

$$P_{opt} = \begin{bmatrix} 0.4034 & 0 & 0.5966 \\ 0.3113 & 0.4383 & 0.2504 \\ 0.2853 & 0.5617 & 0.1530 \end{bmatrix}. \qquad (31)$$

The corresponding matrix Q_{opt} converges now much faster needing only 6 iterations as shown by the green line. The optimization routine is doing something reasonable. The edge with delay is turned off receiving a weight of zero. The second largest eigenvalue of the additive reversibilization of Q_{lazy} is reduced from 0.8198 to 0.8067 yielding faster convergence.

V. Random Delay Model

A fixed delay model is appealing because it is represented by linear updates as

$$x_i(k+1) = \sum_{j=1}^{n+b} q_{ij} x_j(k) \qquad (32)$$

[1]The contrary does not have to hold. A non zero entry, can be set to zero meaning that a node chooses never to use a certain link if for example that link experiences a large delay

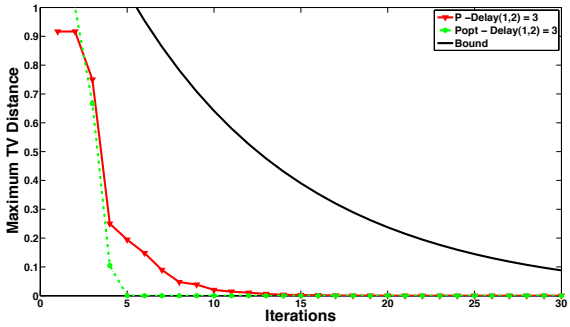

Fig. 2. Optimizing P can yield faster convergence in a network with delays. The total variation distance of P^k without delays is 0 as P represents a complete graph. (Red) Total variation distance of Q^k when edge $(1,2)$ or P experiences a delay of 3. (Green) Total variation distance of Q_{opt} when edge $(1,2)$ of P_{opt} experiences a delay of 3 and P_{opt} is an optimized version of P as described in the text. (Black) Convergence rate bound (24).

where $Q = [q_{ij}]$ is the delayed matrix defined earlier. We can model scenarios where the actual delay per edge does not fluctuate too much around its mean value to which we set the fixed delay. However, network conditions are typically volatile and edge delays are rarely constant. A more realistic model is one where each link experiences a random delay which we assume bounded. Allowing for random delays in discrete time has a significant implication: it is now possible to receive multiple delayed messages from the same sender during the same time slot. In the fixed delay model this obviously cannot happen as equation (32) suggests. Also in continuous time, only one message is received at each time moment since the probability that two messages are delivered at the exact same instant is zero. The situation changes when we consider random delays. For example, take an edge (i,j) whose delay could be 1 or 2. Assume at iteration k node i sends a message m_k to j and at time $k+1$, i sends a new message m_{k+1} to j. If m_k is delayed by 2 time units and m_{k+1} is delayed by 1 unit, then both m_k and m_{k+1} will be delivered to node j at time $k+2$. This scenario can easily occur in practice when the act of receiving a message takes itself a non trivial amount of time during which a second message can arrive. When this happens, the receiving node polling its buffer experiences the arrival of two messages during the same time slot. Under random delays, equation (32) can no longer represent the consensus dynamics. In this section we show how we can still represent the consensus process via (time varying) linear update equations.

To model random bounded delays, we propose replacing each directed edge of the original graph by multiple delay chains with varying amounts of delay. Every time a message is sent, a random decision is made which delay chain the message will take to reach its destination. Recall that our communication network with n computing nodes has $m = 2|E|$ directed edges (not counting the self loops). Each edge can deliver messages with some bounded delay. For simplicity let us assume that every edge has a random delay

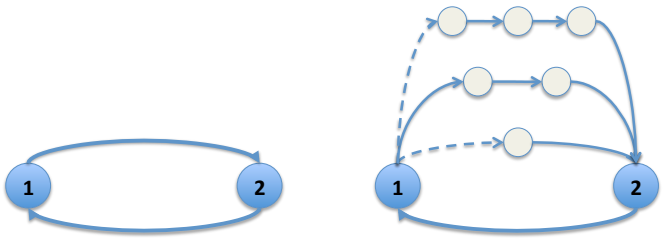

Fig. 3. Adding a random bounded delay on edge $(1, 2)$. At this particular instant, 1 sends with delay 2 since the connections to delays 1 and 3 are deactivated.

between 1 and B. We replace every edge by B delay chains. Similar to the fixed delay case, we insert delay nodes to represent different values of delay. For example for an edge (i, j) with a maximum delay of 3 we add three delay chains $(d_1^1), (d_1^2, d_2^2), (d_1^3, d_2^3, d_3^3)$ to send messages from i to j(see Figure 3). We avoid indexing the delay nodes by edge number to not clutter notation. We augment the graph with $\frac{B(B+1)}{2}$ delay nodes per edge or $b = \frac{mB(B+1)}{2}$ delay nodes total.

Our goal is to write a matrix Q that will describe the consensus dynamics under random delays using linear updates. Every time a message is sent, it is routed randomly through one of the B delay chains. Outgoing edges to the other chains leading to the same recipient are cut off. We can impose a discrete probability distribution on the integers $1, \ldots, B$ to control the expected delay of an edge. For simplicity of exposition, we take a uniform distribution i.e., each delay chain on an edge is selected with probability $\frac{1}{B}$. As we see, the augmented graph topology changes at every iteration based on which outgoing edges to delay chains are active. To describe the consensus update equations we need to model the changing topology. Specifically, at each iteration, each computing node selects for each neighbour how much delay each outgoing message is going to experience. Based on those choices, at iteration k the graph adjacency matrix $A(k)$ is a sample from the set $\{A^1, \ldots, A^{B^m}\}$ of possible adjacency matrices. Moreover, it is important to see that a delay node could either contain a message or be empty and a zero message is not the same as the node being empty. To keep track of which delay nodes are empty we define an indicator sequence $\{\phi(k)\}_{k=1}^{\infty}, \phi(k) \in \{0, 1\}^b$. Using $A(k)$ and $\phi(k)$ we show how to write a transition matrix $Q(k)$ at each iteration k.

We begin by noticing that adjacency matrices $A(k)$ have some structure. Specifically,

$$A(k) = \begin{bmatrix} I_{n\times n} & R(k) \\ J_{b\times n} & C_{b\times b} \end{bmatrix}. \tag{33}$$

The upper left block is the identity representing the self loop edges. $R(k)$ is the only random part in the matrix. It contains a 1 whenever a computing node i sends to a computing node j using delay chain $r = 1, \ldots, B$. For example, if node i_1 sends to j_1 with a delay 2 and node i_2 sends to j_2 with delay 1, $R(k)$ will contain two blocks for those edges with delay encoded as follows:

$$R(k)^T = \begin{array}{c} \\ \vdots \\ d_1^1 \\ d_1^2 \\ d_2^2 \\ d_1^3 \\ d_2^3 \\ d_3^3 \\ d_1^1 \\ d_1^2 \\ d_2^2 \\ d_1^3 \\ d_2^3 \\ d_3^3 \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & \cdots & i_1 & \cdots & i_2 & \cdots & n \end{array} \\ \left[\begin{array}{ccccccc} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right] \end{array}. \tag{34}$$

$J_{b\times n}$ makes the end nodes d_r^r of each delay chain deliver messages to the computing nodes. It has 1s only from the delay nodes $d_r^r, r = 1, \ldots, B$ to the computing nodes. The part of $J_{b\times n}$ corresponding to the two edges of $R(k)$ just discussed will look like

$$J_{b\times n} = \begin{array}{c} \\ \vdots \\ d_1^1 \\ d_1^2 \\ d_2^2 \\ d_1^3 \\ d_2^3 \\ d_3^3 \\ d_1^1 \\ d_1^2 \\ d_2^2 \\ d_1^3 \\ d_2^3 \\ d_3^3 \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & \cdots & j_1 & \cdots & j_2 & \cdots & n \end{array} \\ \left[\begin{array}{ccccccc} 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \hline 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ \hline 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array}\right] \end{array}. \tag{35}$$

Finally, we define the matrix $C_{b\times b}$ for forwarding messages from one delay node to the next on each chain. On a specific delay chain of length h, we can forward messages using an $h \times h$ Toeplitz backward shift matrix with 1s on the first upper

diagonal, i.e.,

$$S_h = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & & \ddots & & \\ 0 & & & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{36}$$

For any edge $r = 1, \ldots, m$, to forward messages through all delay chains we use a block diagonal matrix $K_r = diag(S_1, S_2, \ldots, S_B)$. Finally, since we have m edges

$$C_{b\times b} = diag(K_1, K_2, \ldots, K_m). \tag{37}$$

Looking back at (33), observe that every column of $\begin{bmatrix} R(k) \\ C_{b\times b} \end{bmatrix}$ contains at most one non-zero element.

Next, we define a row vector $\phi(k) \in \mathbb{R}^b$ that keeps track of whether a delay node on any delay chain contains a message or is empty. Initially we have $\phi_0 = \mathbf{0}_b^T$. At iteration k, the first nodes in the delay chains may receive new information depending on which edges are activated by $R(k)$. The rest of the delay nodes will be non-empty depending on whether their predecessors in the chains were non empty in the previous iteration. In other words, $\phi(k)$ evolves as

$$\phi(k) = \mathbf{1}_n^T R(k) + \phi(k-1)C_{b\times b}. \tag{38}$$

By substituting $\phi(k-1)$ using the same recursion and remembering that $\phi(0)$ is zero we conclude that

$$\phi(k) = \mathbf{1}_n^T \sum_{t=0}^{k-1} R(k-t)C_{b\times b}^t. \tag{39}$$

The structure of $C_{b\times b}$ suggests that powers higher than B are equal to zero which makes sense since after at most B iterations a message is out of any delay chain by our bounded delay assumption.

After understanding the structure of changing topology adjacency matrices $A(k)$, to describe the consensus transition matrices $Q(k)$ we need to specify the weights used to combine incoming messages. Recall that each computing node might be receiving multiple messages from a neighbouring computing node arriving from different delay chains. We could specify some elaborate scheme for assigning smaller weight to older messages but for simplicity we will define a matrix that assigns equal weight to all incoming messages. If a computing node i receives w_i messages total at iteration k from all incoming delay chains of all neighbours, i will combine them with weights equal to $\frac{1}{1+w_i}$. The added 1 in the denominator comes from the self loop message. Notice that w_i for each node changes at every iteration and could be zero if all delay nodes at the chain ends are empty. We collect the weights used for combining incoming messages in an $n \times n$ matrix $D(k)$. Each diagonal entry $D(k)_{ii}$ has value $\frac{1}{1+w_i}$. We express $D(k)$ based on known quantities as follows:

$$D(k) = diag\left(\begin{bmatrix} \mathbf{1}_n^T & \phi(k-1) \end{bmatrix} A(k) \begin{bmatrix} I_{n\times n} \\ \mathbf{0}_{b\times n} \end{bmatrix}\right)^{-1} \tag{40}$$

$$= diag\left(\begin{bmatrix} \mathbf{1}_n^T & \phi(k-1) \end{bmatrix} \begin{bmatrix} I_{n\times n} \\ J_{b\times n} \end{bmatrix}\right)^{-1}. \tag{41}$$

The vector $\mathbf{1}_n^T$ accounts for the self loop messages while $\phi(k-1)$ selects out of all the delay nodes at the end of delay chains (indicated by $J_{b\times n}$), the ones that contain non-empty messages. Inverting a diagonal matrix amounts to inverting the individual diagonal elements which are always non-zero.

Using the weights $D(k)$ and activating only the delay nodes that are not empty via $diag(\phi(k-1))$ the transition matrix at iteration k is defined as

$$Q(k) = \begin{bmatrix} D(k) & D(k)J_{b\times n}^T diag(\phi(k-1)) \\ R(k)^T & C_{b\times b}^T \end{bmatrix}. \tag{42}$$

Observe that Q_k contains zero rows for the delay chains that are not used in this iteration. The non-zero rows are by construction stochastic. If $x(k)$ is the augmented state vector of compute and delay nodes, the consensus update equations are written as

$$x_i(k+1) = \sum_{j=1}^{n+b} q_{ij}(k)x_j(k). \tag{43}$$

To conclude this section let us note that forcing each edge to experience a delay of at least 1 is not crucial. The only modification to allow for edges with zero delay, is that the upper left block of $A(k)$ in equation (33) will be the identity to which we need to add a random matrix, call it $R_0(k)$, with 1s for edges (i, j) with zero delay at iteration k. This modification is reflected in the definition of $Q(k)$ in (42) by adding $R_0(k)^T D(k)$ to its upper left block.

VI. Distributed Optimization under Fixed and Random Bounded Delays

One area where consensus algorithms have potential to play a key role is distributed machine learning and optimization where we need to process a large amount of data distributed over a network of n processors. To learn, e.g., a classifier on the full dataset it is typically required to optimize a convex cost function of the form

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), x \in \mathbf{X} \tag{44}$$

where f_i is the local cost function of processor i. One very recent distributed algorithm to minimize $f(x)$ is Distributed Dual Averaging [4]. The minimization proceeds in rounds updating the local estimates of the optimum $x_i(k)$. At each round, a processor i takes a gradient step in the direction determined by the accumulated past gradients from i's neighbouring processors in the communication graph (stored in the dual variable $z_i(k)$), plus the most recent local subgradient $g_i(k)$ of f_i computed at $x_i(k)$. A consensus algorithm using matrix

$P = [p_{ij}]$ is responsible for synchronizing the processors and bringing all the dual variables in agreement about the direction to the optimum. Given a sequence of non-increasing learning rates $a(k)_{k=0}^{\infty}$ each processor repeats two steps:

$$z_i(k+1) = \sum_{j \in Ni} p_{ij} z_j(k) - g_i(k) \tag{45}$$

$$x_i(k+1) = \Pi_{\mathbf{X}}^{\psi}(-z_i(k+1), a(k)) \tag{46}$$

$$= \arg\min_{x \in \mathbf{X}} \left[-z_i^T(k+1) \cdot x + \frac{1}{a(k)} \psi(x) \right], \tag{47}$$

where $\psi(x)$ is a strictly convex proximal function [4]. For large scale problems, we expect that the exchanged messages can easily be many Megabytes in size inducing noticeable delays in communication. Observe that our delay models fit nicely in the distributed dual averaging framework. We can analyze the effect of delays by replacing P with a communication matrix with delays Q. In fact, it can be shown that the convergence results of [4] can be extended in the case of delays; due to space limitations the proofs are omitted. In the case of fixed delays using learning rates $a(k) \propto \sqrt{k}$ we can prove that distributed dual averaging converges like

$$f(\hat{x}_i(k)) - f(x^*) \leq \frac{\theta_1 + \theta_2 \log k}{\sqrt{k}} \tag{48}$$

where $\hat{x}_i(k) = \frac{1}{k} \sum_{t=1}^{k} x_i(t)$ is the running average of processor i's estimate, x^* is the optimum value and θ_1, θ_2 are constants depending on the stationary distribution of Q (which we can compute using the methods introduced in Section IV), the second largest eigenvalue of Q_{lazy} and the bounds we have for $\psi(x)$ and $\|g_i\|$. This result is an extension of Theorem 2 in [4] and shows that order-wise the convergence rate is not affected by the presence of fixed delays.

We conclude this section with two experiments that show the effects of delays when optimizing a hinge loss function as explained in example 1 of [4]. We generate labeled data $\{(u_j, y_j)\}$ on a unit hypersphere and separate them into two classes adding 5% of noise in the labels. We distribute the data among the processors and minimize the misclassification margin

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \quad \text{where} \quad f_i(x) = \sum_{j=1}^{n_i} [1 - y_j u_j^T \cdot x_j]_+. \tag{49}$$

Experiment 1: We run optimization on a random network with $n = 20$ nodes whose average edge degree is 8.1 and maximum degree is 13 not counting self-loops. We use the communication matrix $P = I - \frac{D-A}{d_{max}+1}$. On total we have 1000 datapoints in $\mathbb{R}^{20}$. In the *fixed delays* case we add on each edge r a fixed delay b_r chosen randomly in the interval $[0, B]$. In the *random delays* version, we define a probability distribution on the integers $\{0, \ldots, B\}$ with mean b_r for each edge. The results for $B = 5$ and $B = 10$ are shown in Figure 4. Obviously, adding delays slows down convergence. Interestingly, the random delay model seems to converge faster. Although in expectation random delay and fixed delay

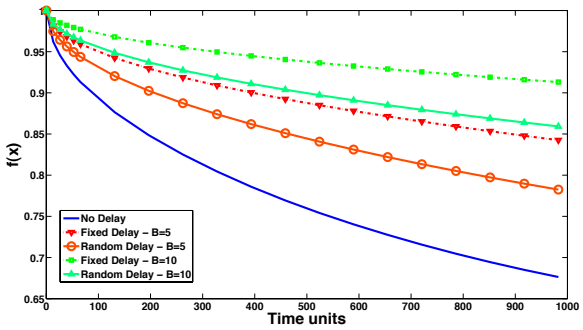

Fig. 4. Evolution of maximum value of $f(x)$ being minimized among 10 processors in the case of fixed and random delays bounded by 5 and 10.

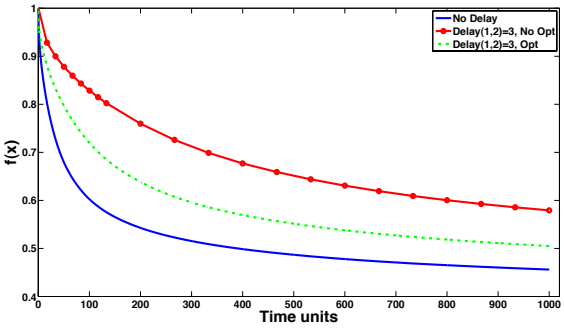

Fig. 5. Optimizing P can yield faster convergence in a delayed network. (Blue) Progress of optimization in complete graph of three nodes without delays. (Red) Progress when edge $(1, 2)$ experiences a delay of 3. (Green) Progress when edge $(1, 2)$ experiences a delay of 3 and P is optimized as described.

models should behave the same as time goes to infinity, for finite time, the non-zero probability of delivering messages earlier than b_r makes a difference. In particular, for $B = 10$ if we divide the total delay experienced by all messages by the total number of messages received, the average delay is 4.5659 for random delays and 5.0358 for fixed delays. The average number of messages each processor receives at each iteration, is 8.0962 for random delays and 8.0956 for fixed delays showing that since the sending rate is the same, we do tend to receive a few more messages in the random delay model. In other words, receiving multiple messages from the same sender does happen in practice.

Experiment 2: It is also interesting to see what happens when we optimize our network according to the delays. Looking back at the example from Section IV-C, we ran dual averaging to minimize $f(x)$ with the original and the optimized network on 3 nodes. Figure 5 shows that adding delays slows down the optimization but using the optimized matrix P_{opt} can considerably improve performance.

VII. Summary and Future Work

In this paper we study the problem of delayed communication in distributed consensus algorithms. We propose two

ways to model delays in discrete time systems based on augmenting the communication graph with delay nodes. The fixed delay model is simpler and can be used when the network delay stays consistently around the average behaviour. We compute the stationary distribution exactly and use results from the theory of non-reversible Markov chains to bound the convergence rate. Moreover, we set up and empirically solve an optimization problem to find weights that minimize convergence time. Our second contribution is the description of a random delay model that seems to have been missing from the literature. For this model we show how to describe the consensus dynamics via linear time varying update equations. This opens the door for analyzing systems with random delays just as we did for the fixed delay model. Finally, we make a connection with distributed optimization showing how our delay models can be used to analyze delays in that context. We also perform simulations using out models to understand the effects of delays.

There are plenty of future directions to pursue from here. For example, although we have a convergence bound, it would be interesting to characterize the precise effect of adding delays on some edge on the second eigenvalue of the transition matrix. Moreover, we have posed and solved the network optimization problem numerically as a proof of concept. To see if the optimization can be solved efficiently is an important step. Furthermore, the convergence of the random delay model has not been discussed here. Very recently we were able to show theoretically what our experiments already suggest, i.e., that the random delay model also converges. We expect this result to appear in an extended version of this paper. In addition to delays, we could study the effect of bounded intercommunication where processors do not send out information at every iteration but rather at least once in a finite number of iterations. Finally, we have briefly investigated the effect of delays for the dual averaging algorithm. We feel confident that our models can be helpful in understanding the effects of delays in other distributed optimization algorithms such as [3], [18].

REFERENCES

[1] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847 – 1864, November 2010.

[2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proceedings of the IEEE*, vol. 95:1, 2007, pp. 215 – 233.

[3] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, January 2009.

[4] J. Duchi, A. Agarwal, and M. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *arXiv:1005.2012v3*, 2010.

[5] S. S. Rama, A. Nedic, and V. Veeravalli, "A new class of distributed optimization algorithms: Application to a new class of distributed optimization algorithms: Application to regression of distributed data," *Optimization Methods and Software*, vol. 00, no. 00, pp. 1–18, 2009.

[6] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics*, Y. Lechevallier and G. Saporta, Eds. Paris, France: Springer, 2010, pp. 177–187.

[7] P.-A. Bliman and G. Ferrari-Trecate, "Average consensus problems in networks of agents with delayed communications," *Automatica*, vol. 44, 2008.

[8] J.-P. Richard, "Time-delay systems: an overview of some recent advances and open problems," *Automatica*, vol. 39, pp. 1667–1694, 2003.

[9] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, September 2004.

[10] A. Seuret, D. V. Dimarogonas, and K. H. Johansson, "Consensus under communication delays," in *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.

[11] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[12] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *IEEE Conference on Decision and Control*, 2006, pp. 2996 – 3000.

[13] M. Cao, S. A. Morse, and B. D. O. Anderson, "Reaching a consensus in a dynamically changing environment: Convergence rates, measurement delays, and asynchronous events," *SIAM Journal on Control and Optimization*, vol. 47, pp. 601–623, 2008.

[14] A. Nedic and A. Ozdaglar, "Convergence rate for consensus with delays," *Journal of Global Optimization*, vol. 47, no. 3, pp. 437–456, 2010.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, pp. 2508–2530, 2006.

[16] R. Montenegro, "The simple random walk and max-degree walk on a directed graph," *Random Structures and Algorithms*, vol. 34, no. 3, pp. 395–407, May 2009.

[17] J. A. Fill, "Eigenvalue bounds on convergence to stationarity for non reversible markov chains, with an application to the exclusion process," *The Annals of Applied Probability*, vol. 1, no. 1, pp. 62–87, 1991.

[18] K. Srivastava and A. Nedic, "Distributed asynchronous constrained stochastic optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 99, 2011.