Distributed Averaging in Dynamic Networks

by

Shreevatsa Rajagopalan

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

© Massachusetts Institute of Technology 2010. All rights reserved.

Author .....
Shreevatsa Rajagopalan

Certified by .....
Devavrat Shah
Associate Professor
Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Patrick Jaillet
Professor
Electrical Engineering and Computer Science
Co-director, Operations Research Center

Distributed Averaging in Dynamic Networks

by

Shreevatsa Rajagopalan

Submitted to the Sloan School of Management
on , in partial fulfillment of the
requirements for the degree of
Master of Science in Operations Research

Abstract

The question of computing average of numbers present at nodes in a network in a distributed manner using gossip or message-passing algorithms has been of great recent interest across disciplines — algorithms, control and robotics, estimation, social networks, etc. It has served as a non-trivial, representative model for an important class of questions arising in these disciplines and thus guiding intellectual progress over the past few decades. In most of these applications, there is inherent dynamics present, such as changes in the network topology in terms of communication links, changes in the values of numbers present at nodes, and nodes joining or leaving. The effect of dynamics in terms of communication links on the design and analysis of algorithms for averaging is reasonably well understood, e.g. [14][2][8][4]. However, little is known about the effect of other forms of dynamics.

In this thesis, we study the effect of such types of dynamics in the context of maintaining average in the network. Specifically, we design *dynamics-aware* message-passing or gossip algorithm that maintains good estimate of average in presence of continous change in numbers at nodes. Clearly, in presence of such dynamics the best one can hope for is a tradeoff between the accuracy of each node's estimate of the average at each time instant and the rate of dynamics. For our algorithm, we characterize this tradeoff and establish it to be *near* optimal. The dependence of the accuracy of the algorithm on the rate of dynamics as well as on the underlying graph structure is quantified.

Thesis Supervisor: Devavrat Shah
Title: Associate Professor
Electrical Engineering and Computer Science

Acknowledgments

I would like to express the deepest gratitude to my supervisor, Professor Devavrat Shah, not only for his extraordinary patience and generosity through the course of my stay at MIT, but also for the amazing example he set, of pursuit and vision of research. Even if through my own personal failings I have yet to successfully emulate them, he will always remain an inspiration.

Thanks are also due to all my fellow students who made up the wonderful environment that is MIT, especially Claudio Telha, Jose Soto, David Goldberg, and many others at ORC and LIDS for many intellectually rewarding discussions, and my suite-mates Shreerang Chhattre, Shashank Dwivedi, and Arpit Agarwal. Finally, it is superfluous to acknowledge my parents, without whose love and support there would be nothing worth speaking of.

Contents

Chapter 1

Introduction

Dynamics is inherent to any networked control and processing system. Typically, the observed or sensed state of the system is summarized in the form of parameters that change over time. To achieve desired functionality of the system, it is necessary to continually or periodically evaluate or compute some function of these parameters. For instance, in a communication network such as the Internet, the problem of *routing* is to decide routes based on network parameters such as the load on the links, the connectivity of the network, etc.; the popular Border Gateway Protocol (BGP) for routing does precisely the same [11], i.e. it updates routes continually based on measured link loads, connectivity, etc. More generally, to share network resources in a generic constrained data network modeled as a queueing network, a theoretically well accepted algorithm, known as the maximum weight or pressure [13], makes decision at each time instant based on the network state summarized via queue-sizes.

In summary, the generic problem of interest is of the following form: the system has external input which is observed, and an appropriate algorithm is devised by a "controller" that often performs some form of optimization. Subsequently "control" or "action" is fed back into the system. And the goal is to keep the system "stable"—close to some "desired" state. In a networked setup, this task has to be performed in a decentralized manner by means of simple algorithm. This is because of the need of scalability as well as engineering constraints. This has led to the study of simple, distributed or gossip or message-passing algorithms across disciplines in recent years.

Most of the known results implicitly or explicitly assume a certain *time-scale separation* in the system dynamics: the feedback is assumed to be *instantaneous*, or, equivalently, the algorithm is assumed to be executed at a *much faster* timescale than the timescale at which changes happen in the system state. Thus, while algorithms designed with such a timescale separation assumption work well when the system is static or slowly changing, they may not work properly when the system is dynamic, which is the case in practice. For example, this is the primary conceptual reason why routing protocols like BGP suffer from instability: BGP updates routing tables to reflect changes in the network, but the network state changes at the same timescale; consequently, this results in the instability of BGP, known as *route flapping* – a major limitation of existing Internet architecture [6].

Therefore, it would be ideal to have algorithms that can deal with the dynamics. It is to be expected that robust algorithms come at a cost in terms of performance – an algorithm may become resistant to changes in the system by giving up some exactness in performance. That is, it is reasonable to expect a trade-off between the accuracy of the algorithm and the degree of system dynamics that it can handle. Therefore, the goal is to design robust algorithms that are "dynamics-aware": an ideal algorithm would work across a range of dynamics, adapting itself according to what is feasible, and it would achieve as good a trade-off between dynamics and performance accuracy as possible. Now, roughly speaking, there are three categories of dynamics: highly dynamic scenario, moderately dynamic scenario and static or slowly changing scenario. Our goal is to investigate performance of averaging algorithms in this entire range of scenarios and understand the interplay between dynamics, network structure and performance.

1.1 Related work

Tsitsiklis [14] initiated the inquiry of interplay between network dynamics and performance of decentralized algorithms in the context of networked control. Specifically, in [14] the question of reaching *consensus* among a collection agents is studied. The al-

gorithm proposed performs linear iterative averaging. The effect of network dynamics in terms of link connectivity on performance of algorithm was studied. This seemingly simple task of (weighted) averaging is used as a "subroutine" for performing estimation as well as solving a class of optimization problems in a distributed manner. An interested reader is referred to the book by Bertsekas and Tsitsiklis [1]. In the above mentioned work as well as in more recent works (see Jadbabaie, Lin and Morse [8], Blondel et al. [3]), the dynamics in terms of the network link connectivity or topology was addressed in a bounded adversarial setup. In a nutshell, the above results established that asymptotically, the algorithm will find the correct average at all nodes (or consensus will be reached) if and only if the network, formed by links that are present infinitely often over time, is connected; the rate of convergence depends on "how often" each link is present. Similar concerns and many more applications have brought this question to life in recent years – for example, work by Boyd et al. [4] on gossip algorithms considers effect of such topological dynamics in a randomized model inspired by peer-to-peer networks. All of these models consider dynamics in terms of the topology, but *not* in terms of (a) the values at the nodes, or (b) node arrival and departure. In this thesis, we shall focus on the effect of dynamics of type (a) and (b) on the performance of algorithm.

Before we proceed towards the content of this thesis, it is worth noting similar aspects of dynamics captured in online and streaming algorithms, as well as stochastic networks. Online and streaming algorithms can be viewed as setup in which the state of the system is gradually revealed. However, it is different from the setup where the system state is continually changing with time. Generically, such algorithms are studied in a centralized setup and hence the network structure does not play any role in determining performance. In the context of stochastic queueing networks, dynamics is inherent. Here, the primary performance goal is the stochastic stability. This is an asymptotic measure, and it usually does not quantify the precise tradeoffs between performance and dynamics that can be achieved by designing dynamics-aware algorithms.

1.2 Our contributions

In this thesis, we address the question of designing dynamics-aware algorithm for the problem of estimating the sum (or equivalently the average) in a network, where the values of nodes are changing continually with time. This problem is chosen because it is the simplest non-trivial problem. As noted above, it is also a well-studied problem and of great interest in control theory, computer science and networks, both independently and as well as a subroutine for many applications such as linear estimation, consensus, projections for dimensionality reduction, maintaining 'sketches', computing gradients in convex optimization, etc. An interested reader is referred to a recent monograph by Shah [12].

In order to study this problem, design algorithms for it, and analyze their performance, we introduce a natural model for the dynamics in terms of changes in the values or numbers present at nodes in the network. These changes may involve either high dynamics or moderate dynamics. We model the former as *multiplicative changes* in the values, and the latter as *additive changes*. As the main result, we design algorithms for both of these scenarios that are dynamics-aware and analyze the tradeoff it induces in terms of accuracy of estimation and the rate of dynamics.

For the high or fast dynamics, modeled as "multiplicative changes", we design an algorithm using extremal properties of the exponential distribution, relation between exponential random variables of different rates and a novel distributed algorithm to maintain minimum of numbers in a network in dynamic setup. This algorithm can be viewed as a (non-trivial) generalization of the algorithm by Mosk-Aoyama and Shah [10] to compute summation in a static network. Specifically, the algorithm of [10] does not extend to dynamic setup readily for two reasons: (1) The algorithm of [10] requires estimation of minimum of numbers in the network and the obvious scheme used does not work in dynamic setup (see Section 1.3 for details). (2) The natural extension of algorithm of [10] would require drawing new random numbers every time nodes change their values. This will lead to a situation where some form of 'time-stamp' would be required associated with each random number and destroying

elegance (as well as distributed property) of the algorithm. Our algorithm is presented in Section 3 and its near optimality properties are stated in Theorems 1 and 2.

For the moderate or slow dynamics, modeled as "additive changes", we study the property of the known linear iterative averaging algorithm. We find that the error in the estimation in such a scenario is bounded in terms of the spectral gap of the communication matrix. The precise result is stated in Theorem 3.

In both cases, the accuracy of the algorithm depends on the 'rate of dynamics' and the graph topology. For multiplicative changes, the network diameter and for additive changes, the spectral gap of the "averaging matrix" affects the accuracy.

1.3 Dynamic minimum

As mentioned earlier, an important hurdle that one needs to overcome to adapt algorithm by [10] in the presence of multiplicative changes, is to estimate minimum of these dynamic numbers in the network in a distributed manner. While this question of 'dynamically computing minimum' is not the main result of this thesis, it is an interesting byproduct that contains the essence of the challenges encountered in designing algorithms in presence of dynamics. Therefore, we shall describe the problem and challenges involved here. Appropriate algorithm is described in Section 3.1.

Now, the problem. Given a network with connectivity graph $G = (V, E)$, value $Y_v(t)$ at node $v \in V$ that changes with time which is indexed by $t \in \{0, 1, 2, \dots\}$. We wish to maintain $Y_{\min}(t) = \min_{v \in V} Y_v(t)$ at each node using simple, distributed algorithm. That is, an algorithm that can only maintain limited data structure (does not scale with network size $|V|$), communicate with its neighbors as per G and preferably does not utilized global network structure.

Now when $Y_v(t) = Y_v$ for all t, i.e. values at nodes does not change, an obvious algorithm for this is as follows: each node $v \in V$ maintains estimate $\tilde{Y}_v$ with initially $\tilde{Y}_v = Y_v$; at each subsequent time step, it updates them as $\tilde{Y}_v = \min(\tilde{Y}_v, \min_{u \in N(v)} \tilde{Y}_u)$, where $N(v) = \{u \in V : (u, v) \in E\}$ denotes the set of neighbors of v. If G is connected, then within diameter many steps, each node learns the precise value of

the minimum.

However, this simple algorithm does not work when the values are changing. The reason is that once the values $\tilde{Y}_v(t)$ all become small (e.g. if all $Y_v(t)$ remain 0 for some time), then, even if the $Y_v(t)$ increase unboundedly, the small estimates $\tilde{Y}_v$ will continue to remain small! In essence, this approach forever "remembers" the lowest value ever attained. Conceptually, the algorithm needs to "forget" the history quickly enough. At the same time, since the algorithm is distributed, it can not "forget" it too quickly or else it may not even be good in the static case ! In other words, an appropriate algorithm must learn *discounting* of old information, while not entirely discarding it. This is a general problem for online algorithms dealing with dynamic systems. For instance, consider the updating rule (used by e.g. Kalman filtering) which maintains a model $y(t)$, and, on obtaining new information $n(t)$ at time t, updates $y(t+1) = \alpha y(t) + (1-\alpha)n(t)$ for some α. In this case, we have $y(t+1) = \sum_{k\geq 0} \alpha^k (1-\alpha) n(t-k)$. By doing so, the effects of older $n(s)$, $s < t$, are dampened; for instance the contributions of $\{n(s)$ for $s \leq t - 1/(1-\alpha)\}$ can effectively be ignored. Of course, in our setup finding such proper α requires knowledge of entire network structure and such property α may not even exists in our setup.

Indeed, there is an elegant (and different) way to resolve this question as explained in Section 3.1. The algorithm presented maintains estimate of the minimum to within an accuracy that depends on the diameter of the network and it is (order) optimal. In general, devising similar function-dependent discounting procedures for other functions would be an interesting direction of future research.

Chapter 2

Model and results

This section describes setup and problem statement followed by main results.

2.1 Setup

We have a network with an underlying connectivity graph $G = (V, E)$ that has $n = |V|$ nodes. We shall assume that the network graph G is connected and has D as its diameter. At each node v, there are variables $X_v(t)$ taking non-negative real values that depend on time $t \geq 0$ with initially $t = 0$ and $X_v(0) = 1$ for all $v \in V$. We shall consider deterministic communication with synchronous time: time is discrete with $t \in \{0, 1, 2, \dots\}$ and communication is synchronous. At each time t, each node can communicates with all of its neighbors, and exchange one number[1] per unit time. That is, if $(u, v) \in E$ then nodes u and v can send a number to each other. The values of the nodes change over time and at most once each discrete time. Specifically, values at nodes change in bounded manner either additively or multiplicatively: let $\delta > 0$ be fixed and given.

1. Additive change: for any $v \in V$, $|X_v(t+1) - X_v(t)| \leq \delta$ for any $t \geq 0$.

2. Multiplicative change: for any $v \in V$, $e^{-\delta} \leq \frac{X(t+1)}{X(t)} \leq e^{\delta}$ for any $t \geq 0$.

[1] In practice the bit-rate of communication is bounded and a number can be exchanged only up to some accuracy. We shall ignore this issue here, but the analysis can be extended to deal with the number of bits of accuracy.

2.2 Problem statement

The goal is to estimate at each node v, using a distributed (message-passing) algorithm, a certain function of these values $X_u(t)$, namely the sum $\sum_{u \in V} X_u(t)$ (or equivalently when n is known, the average), and maintain these estimates as the values change with time. Ideally, one wishes to minimize the error in estimation over all times.

2.3 Results

The results are described separately for multiplicative changes and additive changes.

2.3.1 Multiplicative changes

Theorem 1 *For any given $p \in (0,1)$ and $\epsilon \in (0, 0.35)$, there exists a randomized algorithm (described in Section 3) that maintains estimates $\tilde{X}_v(t)$ at each $v \in V$, so that under deterministic communication model and for any $m = \lceil \frac{3 \ln(2/p)}{\epsilon^2} \rceil$,*

$$(1-\epsilon)e^{-m(D+1)\delta} \leq \frac{\tilde{X}_v(t)}{\sum_v X_v(t)} \leq (1+\epsilon)e^{m(D+1)\delta} \text{ for all } t \geq mD,$$

with probability at least $1-p$.

Theorem 1 suggests the trade-off between dynamics rate δ and accuracy which depends on diameter D. Next, we state lower bound on the accuracy of estimation achievable by *any* randomized (or deterministic) algorithm.

Theorem 2 *If there exists an algorithm that maintains estimates $\tilde{X}_v(t)$ that satisfies $e^{-\Delta} \leq \frac{\tilde{X}_v(t)}{\sum_v X_v(t)} \leq e^{\Delta}$ for all $v \in V$ with probability at least $\frac{3}{4}$, then $\Delta \geq D\delta$.*

Ignoring the ϵ and p, note that the exponents in Theorems 1 and 2 match, up to a factor of m. This shows that the exponent $D\delta$ is inherent effect of dynamics on the performance of algorithm.

2.3.2 Additive changes

The algorithm that we study is the well studied, known linear iterative algorithm. It utilizes a doubly-stochastic (averaging) matrix $A = [A_{uv}] \in \mathbb{R}_+^{n \times n}$ that is graph G conformant, i.e. $A_{uv} = 0$ if $(u, v) \notin E$, and irreducible. The following result states the accuracy of estimation of the algorithm in presence of additive changes. This result is elementary and should be known in literature. We state it here for completeness.

Theorem 3 *The linear iterative algorithm utilizing matrix A described in Section 4 maintains estimate $\widehat{X}_v(t)$ at each node $v \in V$ so that for $t \geq 0$,*

$$\left\| \widehat{X}(t) - X_{\text{ave}}(t)\mathbf{1} \right\| \leq \frac{\delta\sqrt{n}}{1-\lambda},$$

where $\widehat{X}(t) = [\widehat{X}_v(t)]$ is the vector of estimates, $X_{\text{ave}}(t) = \frac{1}{n}\sum_v X_v(t)$ is the actual average at time t and $\lambda = \lambda(A)$ defined as

$$\lambda(A) = \sup_{x \in \mathbb{R}^n : \sum_v x_v = 0} \frac{\|Ax\|}{\|x\|}.$$

2.4 Examples: graphical models

For concreteness, we apply the results to several graph models and see how the performance scales under multiplicative model. We consider line and ring graphs, complete graphs, star graphs and expander graphs.

2.4.1 Line/ring graphs

A line graph consists of n nodes, say $1, 2, \ldots, n$, with an edge between i and $i+1$ for $1 \leq i \leq n-1$. A ring graph (see figure 2-1) is similar, with, in addition to the edges in the line graph, an edge between n and 1. For a line graph or ring graph with n nodes, its diameter is $D = \Theta(n)$. Therefore, in Theorems 1 and 2, for the deterministic communication model, we have, with probability at least $1 - p$ at all

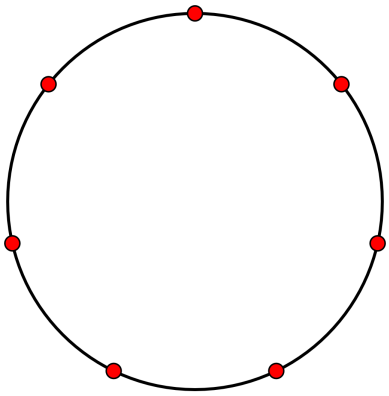

Figure 2-1: A ring graph on $n = 7$ nodes. Each node communicates with its two neighbors.

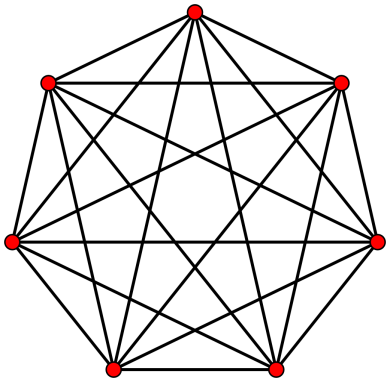

Figure 2-2: A complete graph.

times t,

$$e^{-\Theta(n\delta)} \leq \frac{\tilde{X}_v(t)}{\sum_v X_v(t)} \leq e^{\Theta(n\delta)}$$

ignoring the constants that depend on p and ϵ.

2.4.2 Complete graphs

A complete graph on n nodes has an edge between each pair of nodes; thus its diameter is 1. In this case, the theorems give good results, but a straightforward algorithm would be even simpler: since each node communicates with *all nodes* at each time t, it has all the information that was in the network at time $t - 1$, and estimation is therefore trivial, to within a difference of δ or factor of e^{δ} for additive and multiplicative changes respectively. Note that this error exactly matches the

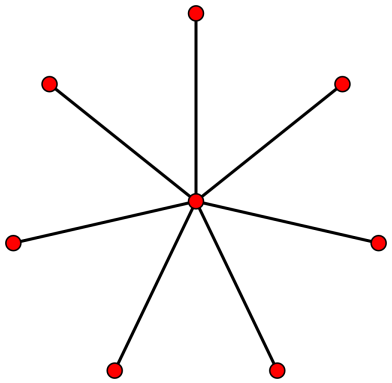

Figure 2-3: A star graph.

lower bound of $D\delta = \delta$ from Theorem 2:

$$e^{-\delta} \leq \frac{\tilde{X}_v(t)}{\sum_v X_v(t)} \leq e^{\delta}$$

2.4.3 Star graphs

A star graph is a graph whose edges are exactly all those between one fixed node and every other node. Compared to complete graphs which have $\binom{n}{2}$ edges, star graphs are very sparse, having only $n-1$ edges. However, since their diameter is small ($D = 2$), the behavior of the algorithms on the star graph is qualitatively the same as for complete graphs (with deterministic communication). This shows that the diameter is the predominant feature of the topology which affects the performance.

2.4.4 Expander graphs

Expander graphs are graphs that are sparse but have high connectivity. In expander graphs, the diameter is $O(\log n)$ [7]. Since $e^{\Theta(\log n)} = n^{\Theta(1)}$, the estimates can be made to satisfy

$$(1-\epsilon)n^{-\Theta(m\delta)} \leq \frac{\tilde{X}_v(t)}{\sum_v X_v(t)} \leq (1+\epsilon)n^{\Theta(m\delta)}$$

It should be noted that *small-world networks*[15][9] though not necessarily expander graphs, have a diameter of $O(\text{polylog}(n))$.

Chapter 3

Multiplicative changes

This section describes the algorithm for maintaining estimate of sum at each node in the presence of multiplicative changes in nodes.

Our main algorithm, as an important subroutine uses an algorithm for maintaining *minimum* (rather than the sum) of values in a network in presence of dynamics. This is described in Section 3.1. In a way, this answers question posed in Section 1.3.

Next, in Section 3.2, we show how the algorithm of Section 3.1 can be utilized to estimate the sum with values changing multiplicatively. We shall present two algorithms. The first algorithm described in Section 3.2.1) merely uses the maximum of the values as an estimate for the sum. The second algorithm described in Section 3.2.2 involves transforming problem of summation into the problem of minimum computation by means of randomization. These two algorithms, collectively establish Theorem 1.

Finally, in Section 3.3, we establish the lower bound, using adversarial arguments, on the performance achievable by *any* algorithm. This establishes Theorem 2.

3.1 Estimating the minimum

In this section, we state an algorithm for maintaining estimates of the minimum of values in the network: let $Y_v(t)$ be non-negative value at node $v \in V$ so that for any

$v \in V$ and $t \geq 0$,

$$e^{-\delta} \leq \frac{Y_v(t+1)}{Y_v(t)} \leq e^{\delta}.$$

For this section, consider this as the problem of interest. However, as mentioned earlier, in Section 3.2.2, we shall transform the problem of estimating $\sum X_v(t)$ into the problem of computing minimum of values $Y_v(t)$ where $Y_v(t)$ will be a (random) function of the actual value $X_v(t)$.

Let $N_k(v)$ denote the set of nodes at distance (with respect to the shortest path metric in G) $\leq k$ from v. Let $N(v) = N_1(v)$ denote the set of neighbors of v including v itself, and let $d(u, v)$ denote the (shortest-path) distance between u and v. Clearly, diameter of G can be defined as

$$D = \max_{u,v \in V} d(u, v).$$

As discussed in Section 1.3, if $\delta = 0$ or the case when $Y_v(t) = Y_v$ for all t, essentially a trivial message-passing algorithm can find the correct minimum in time D: each node maintains an estimate $\tilde{Y}_v(t)$ of the actual minimum $\min_v Y_v$, and, on communicating with all its neighbors and receiving their estimates $\tilde{Y}_v(t-1)$, updates its estimate simply as the minimum of all known values: $\tilde{Y}_v(t) = \min_{u \in N(v)} \tilde{Y}_u(t-1)$. While this algorithm works for $\delta = 0$, it can be arbitrarily bad for $\delta > 0$ as explained in Section 1.3. We present our algorithm for any $\delta \geq 0$.

3.1.1 Algorithm

Each node v maintains an estimate $\tilde{Y}_v(t)$ of $\min_u Y_u(t)$. At the beginning of each time t, each node v exchanges its current estimate $\tilde{Y}_v(t-1)$, from time $t-1$, with all its neighbors, and similarly receives all their estimates. Then, the new estimate $\tilde{Y}_v(t)$ is computed as:

$$\tilde{Y}_v(t) = \min\Big\{Y_v(t), \min_{u \in N(v)} \tilde{Y}_u(t-1)e^{\delta}\Big\}. \tag{3.1}$$

In other words, the new estimate of node v, at time t, is the minimum of its own new value, and the estimates of all its neighbors at time $t-1$, scaled up by e^{δ}. This last

factor is the essential component of the discounting. We shall assume that initially, $t = 0$ and $\tilde{Y}_v(0) = Y_v(0)$. And we shall use the definition $\tilde{Y}_v(s) = \infty$ for $s < 0$.

Theorem 4 *Under the above setup, for any $t \geq D$,*

$$\Big(\min_{u\in V} Y_u(t)\Big) \leq \tilde{Y}_v(t) \leq e^{2D\delta}\Big(\min_{u\in V} Y_u(t)\Big). \tag{3.2}$$

Proof. To start with, we claim the following identity: for any $v \in V$ and $t \geq 0$,

$$\tilde{Y}_v(t) = \min_{k\geq 0} \min_{u\in N_k(v)} \Big\{Y_u(t-k)e^{k\delta}\Big\} \tag{3.3}$$

$$= \min_{u\in V} \min_{k\geq d(u,v)} \Big\{Y_u(t-k)e^{k\delta}\Big\} \tag{3.4}$$

In above we use definition $Y_u(s) = \infty$ for $s < 0$. To establish this identity, observe that the two expressions (3.3) and (3.4) are equal — only the order of the mins is reversed; they both evaluate the minimum over the same set of pairs (u, k). Thus it is enough to prove the first equality.

The equality follows by recursive application of the update rule (3.1). To that end, application of the rule once and twice respectively, we obtain

$$\begin{aligned}\tilde{Y}_v(t) &= \min\Big\{Y_v(t), \min_{u\in N(v)} \tilde{Y}_u(t-1)e^{\delta}\Big\} \\ &= \min\Big\{Y_v(t), \min_{u\in N(v)} Y_u(t-1)e^{\delta}, \min_{w\in N_2(v)} \tilde{Y}_w(t-2)e^{2\delta}\Big\}.\end{aligned}$$

More generally, for any $m \geq 1$, it follows that

$$\begin{aligned}\tilde{Y}_v(t) = \min\Big\{ &\min_{k=0}^{m} \min_{u\in N_k(v)} \left\{Y_u(t-k)e^{k\delta}\right\}, \\ &\min_{w\in N_{m+1}(v)} \tilde{Y}_w(t-m-1)e^{(m+1)\delta}\Big\}.\end{aligned}$$

By definition, we assume $\tilde{Y}_w(t) = \infty$ for $t < 0$. Therefore, by taking $m \to \infty$ and

with notation $Y_u(s) = \infty$ for any $u \in V$ and $s < 0$, we obtain

$$\tilde{Y}_v(t) = \min_{k \geq 0} \min_{u \in N_k(v)} \Big\{ Y_u(t-k) e^{k\delta} \Big\}.$$

This establishes identity (3.3) (and hence (3.4)) as desired.

Now, we shall utilize (3.4) for $\tilde{Y}_v(t)$ to establish (3.2) for any given $v \in V$. To this end, for a given $v \in V$ using (3.4) it follows that

$$\tilde{Y}_v(t) = \min_{u \in V} \left\{ Y_u(t - d(u,v)) e^{d(u,v)\delta} \right\} \tag{3.5}$$

This follows immediately from (3.4) and the fact that the value $Y_u(\cdot)$ increases (re. decreases) by multiplicative factor e^{δ} (re. $e^{-\delta}$) in unit time.

Now we are ready to establish (3.2). To start with observe that due to bound on multiplicative change, $Y_u(t) \leq Y_u(t - d(u,v)) e^{d(u,v)\delta}$. Therefore, from (3.5), it follows that for any $t \geq 0$,

$$\Big(\min_{u \in V} Y_u(t) \Big) \leq \tilde{Y}_v(t). \tag{3.6}$$

Similarly, due to bound on multiplicative change, it follows that $Y_u(t - d(u,v)) \leq Y_u(t) e^{d(u,v)\delta}$ for $t \geq d(u,v)$. Therefore, from (3.5) it follows that for $t \geq D$,

$$\begin{aligned} \tilde{Y}_v(t) &\leq \min_{u \in V} \Big\{ Y_u(t) e^{2d(u,v)\delta} \Big\} \\ &\leq \min_{u \in V} \Big\{ e^{2D\delta} Y_u(t) \Big\} \\ &\leq e^{2D\delta} \Big(\min_{u \in V} Y_u(t) \Big). \end{aligned} \tag{3.7}$$

From (3.6) and (3.7), the desired result (3.2) follows. □

Note that the error is one-sided, but by taking the estimate to be $\tilde{Y}_v(t) e^{D\delta}$ instead, the guarantee within a factor $e^{D\delta}$ can be provided. In fact, it can be shown using argument similar to that used in Section 3.3, that this is the best possible bound that any algorithm can guarantee.

Although we proved these results about estimating the minimum for multiplicative changes, since that is the form in which we will use this algorithm, it is worth noting that a similar bound, simpler in form, holds for estimating the minimum under additive changes. This requires using an update rule analogous to equation (3.1).

Theorem 5 (Additive version) *Suppose value at node $v \in V$ at time t be $Y_v(t)$. For all $t \geq 0$ and $v \in V$, let*

$$|Y_v(t+1) - Y_v(t)| \leq \delta.$$

Let the estimation at node v at time $t \geq 0$, denoted by $\tilde{Y}_v(t)$ be updated as

$$\tilde{Y}_v(t) = \min\left\{Y_v(t), \min_{u \in N(v)} \tilde{Y}_u(t-1) + \delta\right\}, \tag{3.8}$$

with $\tilde{Y}_v(s) = \infty$ for all $s < 0$. Then, for all $t \geq D$

$$\left(\min_{u \in V} Y_u(t)\right) \leq \tilde{Y}_v(t) \leq \left(\min_{u \in V} Y_u(t)\right) + 2D\delta \tag{3.9}$$

Proof. Define $Z_v(t) = \exp\left(Y_v(t)\right)$ and $\tilde{Z}_v(t) = \exp(\tilde{Y}_v(t))$. Observe that then $Z_v(\cdot)$ and $\tilde{Z}_v(\cdot)$ are similar to the setup of multiplicative change. Therefore, by Theorem 4, it follows that

$$\left(\min_{u \in V} Z_u(t)\right) \leq \tilde{Z}_v(t) \leq e^{2D\delta}\left(\min_{u \in V} Z_u(t)\right).$$

This is equivalent to (3.9). □

It is easy to see that similar algorithm can be devised to estimate the maximum instead with similar guarantees, a fact which we shall use in the next section.

3.2 Estimating the sum

In this section, we describe our algorithms for maintaining estimates of the sum, under multiplicative changes. We shall start with a simpler algorithm, that maintains

maximum as an estimate of the summation. This will be followed by a more involved algorithm that builds on the algorithm of Mosk-Aoyama and Shah[10].

3.2.1 A simple estimator using maximum

An estimate for the sum, that would be within a factor of n, is the *maximum*. This is because the sum of a set of n positive numbers is larger than the maximum of the numbers, but no more than n times the maximum. Therefore, $\sqrt{n}$ times the maximum provides an estimator of summation that is within factor $1/\sqrt{n}$ and $\sqrt{n}$ of the summation. In Section 3.1 we described an algorithm to maintain estimation of minimum within multiplicative factor $e^{D\delta}$. A similar algorithm (with min replaced by max and e^{δ} by $e^{-\delta}$ in (3.1)) for maximum provides its estimation within factor $e^{D\delta}$ as well. Therefore, effectively we have an estimation of summation within factor $\sqrt{n}e^{D\delta}$. Next, we shall describe a more involved algorithm for which the estimation will be within factor $e^{\Theta(D\delta)}$ of summation. Now, when δ is large enough, i.e. $\sqrt{n} \ll e^{D\delta}$, both the naive maximum based estimation and the more involved algorithm perform similarly. However, when $e^{D\delta} \ll \sqrt{n}$ the more involved algorithm dominates the naive maximum based estimator. In general, the randomized algorithm described next has essentially the best possible performance with error within factor $e^{\Theta(D\delta)}$.

3.2.2 Randomized estimator

We now turn to our algorithm for estimating the sum of values in a network, using a combination of the algorithm for estimating the minimum found in the previous section, and an apt transformation on exponentially distributed random variables.

Recall that $X_v(t)$ denote the non-negative values at nodes $v \in V$ at time t. Without loss of generality, let us assume that $X_v(0) = 1$ for all $v \in V$. We shall assume that for any $t \geq 0$ and $v \in V$,

$$e^{-\delta} \leq \frac{X_v(t+1)}{X_v(t)} \leq e^{\delta}.$$

We shall associate an auxiliary variable $Y_v(t)$ to each node as follows: $Y_v(t)$ is a random

function of $X_v(t)$, distributed as an exponential random variable with parameter $X_v(t)$, denoted $Y_v(t) \sim \text{Exp}(X_v(t))$. That is,

$$\mathbb{P}\big(Y_v(t) \geq \zeta\big) = e^{-X_v(t)\zeta}, \qquad \text{for any } \zeta \geq 0.$$

As we shall see, $Y_v(t)$ will also change dynamically so that for any $v \in V$ and $t \geq 0$,

$$e^{-\delta} \leq \frac{Y_v(t+1)}{Y_v(t)} \leq e^{\delta}.$$

Let $\tilde{Y}_v(t)$ be estimation of $\min_{u \in V} Y_u(t)$ node v as per algorithm described in Section 3.1 using the multiplicative factor e^{δ}. Using $\tilde{Y}_v(t)$, we shall obtain estimate $\tilde{X}_v(t)$ of the sum $\sum_{u \in V} X_u(t)$. Next, we describe key intuition behind such an algorithm followed by precise algorithm.

The basic idea behind the algorithm, which enables us to transform the problem of computing the sum to that of the minimum, is the following elementary but unique extremal property of the exponential distribution: *The minimum of a set of exponential random variables is an exponential random variable whose rate is the sum of their rates.* Therefore, if we had random variables $Y_v(t) \sim \text{Exp}(X_v(t))$, then their minimum is distributed $\sim \text{Exp}\big(\sum_{u \in V} X_u(t)\big)$. Therefore, the inverse of mean (or average) of minimum $\min_{u \in V} Y_u(t)$ is $\sum_{u \in V} X_u(t)$. Therefore, by computing several independent samples of minimum $\min_{u \in V} Y_u(t)$, we can obtain reasonable estimate of $\sum_{u \in V} X_u(t)$.

The above is precisely the idea (and algorithm) used by [10] for static setup. However, its adaption to dynamic scenario is not immediate. Specifically, we need to make sure that $Y_v(t) \sim \text{Exp}(X_v(t))$ for all t and $v \in V$ while ensuring that $Y_v(t+1)/Y_v(t)$ is within factor $e^{-\delta}$ and e^{δ} with probability 1 for all t. This is where another property of exponential distribution comes to our rescue. Specifically, consider the following rule for generating $Y_v(t)$ for all $t \geq 0$ and $v \in V$:

$$\begin{aligned} &\text{at } t = 0, && Y_v(0) \sim \text{Exp}(X_v(0)), \\ &\text{for } t > 0, && Y_v(t+1) = \frac{X_v(t)}{X_v(t+1)} Y_v(t). \end{aligned} \tag{3.10}$$

Thus the variable $Y_v(0)$ is randomly generated once at $t = 0$, and for all subsequent t, $Y_v(t)$ is a deterministic function of $Y_v(0)$ and changes in $X_v(\cdot)$. It is easy to check that this achieves $Y_v(t) \sim \text{Exp}(X_v(t))$. Further, we have

$$e^{-\delta} \leq \frac{Y_v(t)}{Y_v(t+1)} \leq e^{\delta}.$$

Next, we shall describe the precise algorithm.

Algorithm Given $p \in (0,1)$ and $\epsilon \in (0, 0.35)$, let $m = \lceil \frac{3\ln(2/p)}{\epsilon^2} \rceil$. For each $v \in V$ and $t \geq 0$, define $Y_{v,i}(t)$ with $1 \leq i \leq m$ as follows:

- for $t = 0$, generate $Y_{v,i}(0)$ independently of everything else and as per exponential distribution with parameter $X_v(0)$.
- for $t \geq 1$, update $Y_{v,i}(t+1) = X_v(t)Y_{v,i}(t)/X_v(t+1)$.

As discussed above, $Y_{v,i}(t) \sim \text{Exp}(X_v(t))$ for all (v,i) and for all t. The m different indices i can be thought of as m independent copies of the algorithm running in parallel. The update rule involves the different i's "taking turns" in round-robin fashion: at each time t, if $t \equiv i \mod m$, then the nodes exchange their i-values and update them as

$$\tilde{Y}_{v,i}(t) = \min\{Y_{v,i}(t), \min_{u \in N(v)} \tilde{Y}_{u,i}(t-1)e^{m\delta}\}. \tag{3.11}$$

Define

$$\tilde{Y}_v(t) = \frac{1}{m}\sum_{i=1}^{m} \tilde{Y}_{v,i}(t). \tag{3.12}$$

Then, the estimate of $\sum_{u \in V} X_u(t)$ at node v is given by

$$\tilde{X}_v(t) = \frac{e^{m(D+1)\delta}}{\tilde{Y}_v(t)}. \tag{3.13}$$

Next, we establish the bound claimed in Theorem 1 for this algorithm.

Proof. (Theorem 1) As $\tilde{Y}_{v,i}(t)$ changes only at $t \equiv i \mod m$, we have $\tilde{Y}_{v,i}(t-1) = \tilde{Y}_{v,i}(t-2) = \cdots = \tilde{Y}_{v,i}(t-m)$. In addition, $e^{-m\delta} \leq \frac{Y_{v,i}(t)}{Y_{v,i}(t-m)} \leq e^{m\delta}$, so focusing on the index i and times $t \equiv i \mod m$, (3.11) is simply the update rule (3.1) for the values $Y_{v,i}(t)$ and associated estimates $\tilde{Y}_{v,i}(t)$:

$$\tilde{Y}_{v,i}(t) = \min\Big\{Y_{v,i}(t), \min_{u\in N(v)} \tilde{Y}_{u,i}(t-m)e^{m\delta}\Big\}.$$

Therefore, from Theorem 4 of Section 3.1, we have that for $t \equiv i \mod m$,

$$\Big(\min_{u\in V} Y_{u,i}(t)\Big) \leq \tilde{Y}_{vi}(t) \leq e^{2Dm\delta}\Big(\min_{u\in V} Y_{u,i}(t)\Big), \text{ for each } i. \tag{3.14}$$

Therefore, for any $t \geq mD$

$$e^{-m\delta}\Big(\min_{u\in V} Y_{u,i}(t)\Big) \leq \tilde{Y}_{vi}(t) \leq e^{(2D+1)m\delta}\Big(\min_{u\in V} Y_{u,i}(t)\Big), \text{ for each } i. \tag{3.15}$$

For ease of notation, let $Z_i = \min_{u\in V} Y_{u,i}(t)$ with $1 \leq i \leq m$ and $Z = \frac{1}{m}\sum_{i=1}^{m} Z_i$. Summing up the inequalities in (3.15) over i, and using (3.12), we can write

$$e^{-m\delta} Z \leq \tilde{Y}_v(t) \leq Z e^{(2D+1)m\delta}.$$

The Z_is are IID random variables, each exponentially distributed with rate $\lambda = \sum_{v\in V} X_v(t)$, which is the sum we want to estimate. By large deviation estimation for exponential distribution, it follows that

$$\begin{aligned}
\mathbb{P}\Big(Z \geq \frac{c}{\lambda}\Big) &\leq \exp\left(-m\left(c - 1 - \ln c\right)\right) \qquad \text{for } c > 1, \\
\mathbb{P}\Big(Z \leq \frac{c}{\lambda}\Big) &\leq \exp\left(-m\left(c - 1 - \ln c\right)\right) \qquad \text{for } c < 1.
\end{aligned}$$

Therefore, it can be verified that for any $\epsilon \in (0, 0.35)$,

$$\mathbb{P}\Big(\frac{1}{\lambda(1-\epsilon)} \leq Z \leq \frac{1}{\lambda(1+\epsilon)}\Big) \leq \exp\left(-m\epsilon^2/3\right).$$

Thus, for any $\epsilon \in (0, 0.35)$ and $p \in (0, 1)$, if $m = \lceil \frac{3\ln(2/p)}{\epsilon^2} \rceil$, then with probability at least $1 - p$

$$\frac{e^{-m\delta}}{\lambda(1+\epsilon)} \leq \ e^{-m\delta} Z \ \leq \ \tilde{Y}_v(t) \ \leq \ e^{(2D+1)m\delta} Z \ \leq \ \frac{e^{(2D+1)m\delta}}{\lambda(1-\epsilon)}.$$

Finally, since the estimate of node v is $\tilde{X}_v(t) = \frac{e^{m(D+1)\delta}}{Y_v(t)}$, it follows that with probability at least $1 - p$ and for $t \geq mD$,

$$(1+\epsilon)e^{m(D+1)\delta} \geq \frac{\tilde{X}_v(t)}{\lambda} \geq (1-\epsilon)e^{-(1+D)m\delta}.$$

This completes the proof of Theorem 1. □

3.3 Lower bound

This section establishes lower bound on the accuracy for any algorithm operating under multiplicative changes. Specifically, we shall prove Theorem 2.

To that end, suppose we have an algorithm that, for any values $X_v(t)$ with the promise that

$$e^{-\delta} \leq \frac{X_v(t+1)}{X_v(t)} \leq e^{\delta},$$

can maintain estimates $\tilde{X}_v(t)$ with the guarantee that for any given $t \geq 0$ and all v,

$$e^{-\Delta} \leq \frac{\tilde{X}_v(t)}{\sum_u X_u(t)} \leq e^{\Delta}$$

with probability at least $\frac{3}{4}$.

We observe that the estimate $\tilde{X}_v(t)$ can depend only on $X_v(t)$, $X_u(t-1), u \in N(v)$, and $\tilde{X}_u(t-1), u \in N(v)$, and not on $X_u(t)$. (Formally, $\tilde{X}_v(t)$ is independent of $X_u(t)$ when conditioned on $X_u(t-1)$ and $\tilde{X}_u(t-1)$, for $u \neq v$.) More generally, $\tilde{X}_v(t)$ can depend on $X_u(t-k)$ or $\tilde{X}_u(t-k)$ only if $k \geq d(u, v)$, since any information at node u takes time at least $d(u, v)$ to reach node v.

Let u, v be any two nodes, with $d(u, v) = k$. Consider the following scenario. For

all times t up to a certain time (say some large enough time t_0), let $X_w(t) = 1$ for all $w \neq u$, and $X_u(t) = M$ where M is some sufficiently large constant. Now consider the following two cases for $t > t_0$:

1. Values at all other nodes remain the same and $X_u(t)$ increases by e^{δ} at each $t > t_0$.

2. Values at all other nodes remain the same and $X_u(t)$ decreases by factor e^{δ}, i.e. multiplied by $e^{-\delta}$ at each $t > t_0$.

In either case, at time $t = t_0 + k$, the information that is made available at node v (including that about node u) is exactly the same. Therefore, the (randomized) decision taken by the algorithm is the same (distributionally) in either case. Since algorithm can predict the correct value within factor e^{Δ} with probability at least $3/4$, it must be that $e^{2\Delta}$ is at least the ratio of the summation of values at node u in the above two cases. Now in the first case, the summation could be as large as $Me^{k\delta} + (n-1)$ and in the second case summation could be as small as $Me^{-k\delta} + (n-1)$. Therefore,

$$e^{2\Delta} \geq \frac{Me^{k\delta} + (n-1)}{Me^{-k\delta} + (n-1)}.$$

Applying this argument for pair (v, u) so that $d(u, v) = D$ and taking M arbitrary large, we obtain that

$$\Delta \geq D\delta.$$

This establishes Theorem 2.

Chapter 4

Additive changes

Here we consider the scenario with additive changes. As mentioned in Section 2.3.2, we shall study the known linear iterative algorithm and establish Theorem 3.

To that end, let $X_v(t)$ be value at node $v \in V$ at time $t \geq 0$. Let $\mathbf{\Delta}(t) = X(t+1) - X(t)$ denote the vector of changes in the values at nodes from time t to $t+1$. By the definition of additive change model, for all $t \geq 0$ and $v \in V$

$$|\mathbf{\Delta}_v(t)| \;=\; |X_v(t+1) - X_v(t)| \;\leq\; \delta.$$

The aim here is to estimate average $X_{\text{ave}}(t)$ where

$$X_{\text{ave}}(t) = \frac{1}{n}\sum_{i=1}^{n} X_i(t).$$

Note that this is equivalent to computing summation assuming n is known. In what follows, we describe the known algorithm maintains estimate $\widehat{X}_v(t)$ of $X_{\text{ave}}(t)$ at each node $v \in V$ for all $t \geq 0$. The algorithm utilizes a doubly stochastic, graph G conformant and irreducible matrix $A = [A_{uv}] \in \mathbb{R}_+^{n\times n}$. That is, A satisfies the following properties:

1. $A_{uv} = 0$ if $(u, v) \notin E$.

2. $\sum_v A_{uv} = 1$ for all $u \in V$.

3. $\sum_u A_{uv} = 1$ for all $v \in V$.

4. Directed graph $G(A) = (V, E(A))$ is connected where a directed edge $(u, v) \in E(A)$ iff $A_{uv} > 0$.

4.1 Algorithm

In words, the algorithm is simple: at each time, each node communicates with its neighbor and exchanges their current estimates. Subsequently, each node updates its own estimate as weighted summation of the estimates received from its neighbors and the change in its value. Specifically, for any node $v \in V$, its estimate $\widehat{X}_v(t+1)$ is updated as

$$\widehat{X}_v(t+1) = \sum_u A_{vu}\widehat{X}_u(t) + \mathbf{\Delta}_v(t). \tag{4.1}$$

Equivalently,

$$\widehat{X}(t+1) = A\widehat{X}(t) + \mathbf{\Delta}(t). \tag{4.2}$$

This is an immediate extension of the known linear iterative averaging algorithm (for static case) whose update rule is given by

$$\widehat{X}(t+1) = A\widehat{X}(t).$$

In literature, it is well known that when values at nodes do not change (static case) and A satisfies properties listed above, then $\widehat{X}(t) \to X_{\text{ave}}$ where X_{ave} is the average of the node values. Here, node values and $X_{\text{ave}}(t)$ change over time. We shall establish that $\widehat{X}(t)$ remains 'close to' $X_{\text{ave}}(t)\mathbf{1}$.

4.2 Example of A

There are many ways to design such doubly stochastic, graph conformant and irreducible matrices A. A simple choice, that requires only a known upper bound d on the max-vertex degree, i.e. $d \geq \max_v d_v$, is derived from the Metropolis–Hastings rule[5]. As per this, define A as

$$A_{uv} = \begin{cases} 0 & \text{if } v \text{ is not a neighbor of } u, \\ \frac{1}{d} & \text{if } v \neq u \text{ is a neighbor of } u, \\ 1 - \frac{d_u}{d} & \text{if } v = u. \end{cases}$$

Since row u or column u has exactly d_u entries equal to $1/d$, and one (diagonal) entry equal to $1 - d_u/d$, it follows that A is doubly stochastic. It is graph G conformant by definition and since $G(A) = G$, if G is connected (which we assume here) then so is $G(A)$.

4.3 Analysis

Here we establish Theorem 3. Define $Y(t) = \widehat{X}(t) - X_{\text{ave}}(t)\mathbf{1}$, the error vector in estimation at time t. Initially, we assume that $X_v(0) = 1$ for all v and $\widehat{X}_v(0) = X_v(0) = 1$. Therefore, $Y(0) = \mathbf{0}$. Also define

$$\mathbf{\Delta}_{\text{ave}}(t) = \frac{1}{n} \sum_v \mathbf{\Delta}_v(t).$$

Observe that, since A is doubly stochastic, $A(t)\mathbf{1} = \mathbf{1}$. Therefore, using (4.2) it follows that

$$\begin{aligned} Y(t+1) &= \widehat{X}(t+1) - X_{\text{ave}}(t+1)\mathbf{1} \\ &= A(t)\widehat{X}(t) + \mathbf{\Delta}(t) - \big(X_{\text{ave}}(t) + \mathbf{\Delta}_{\text{ave}}(t)\big)\mathbf{1} \\ &= A(t)Y(t) + e(t). \end{aligned} \tag{4.3}$$

where $e(t) = \mathbf{\Delta}(t) - \mathbf{\Delta}_{\text{ave}}(t)\mathbf{1}$. The goal is to bound $\|Y(\cdot\|$ using identity (4.3). To that end, consider $e = e(t)$ for any t. Under additive change model $|e_v| \leq 2\delta$ for all $v \in V$ and $\sum_v e_v = 0$. This means that

$$\|e\|^2 = \sum_v e_v^2 \;\leq\; \sum_v 4\delta^2 \;=\; 4n\delta^2. \tag{4.4}$$

Using this, we shall bound $\|Y(\cdot)\|$. By recursive application of (4.3), it follows that

$$Y(t+1) = \sum_{k=0}^{t} A^k e(t-k) + Y(0) \;=\; \sum_{k=0}^{t} A^k e(t-k).$$

Therefore, using triangle's inequality

$$\begin{aligned}
\|Y(t+1)\| &\leq \sum_{k=0}^{t} \left\|A^k e(t-k)\right\| \\
&\overset{(a)}{\leq} \sum_{k=0}^{t} \lambda^k 2\delta\sqrt{n} \\
&\leq \frac{2\delta\sqrt{n}}{1-\lambda},
\end{aligned}$$

where recall that $\lambda = \lambda(A)$ is defined as

$$\lambda(A) = \sup_{x \in \mathbb{R}^n : \sum_v x_v = 0} \frac{\|Ax\|}{\|x\|}.$$

To obtain (a), we have used the fact that if $x \in \mathbb{R}^n$ such that $\sum_v x_v = 0$, then $\sum_v z_v = 0$ where $z = Ax$ since A is doubly stochastic. This completes the proof of Theorem 3.

Chapter 5

Conclusion

In this thesis, we presented "dynamics-aware" distributed algorithms for estimating the sum or average of values in a network, under dynamics. The algorithm described exhibit near-optimal tradeoff between accuracy of estimation and rate of dynamics. Specifically, the error in accuracy depends on network topology.

Some extensions can be considered, which have not been developed here. If we do not know the actual rate of change that we denoted δ, we could still estimate it through a message-passing algorithm similar to the one we used to find the minimum. This may be possible even if the rate changes, e.g. if the values become stable after a while.

Going forward, developing such dynamics-aware algorithms for distributed computation of generic functions remains an important research challenge.

Bibliography

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

[2] D.P. Bertsekas, J.N. Tsitsiklis, and M. Athans. Convergence theories of distributed iterative processes: a survey. *Laboratory for Information and Decision Systems, Massachusetts Institute of Technology*, 1983.

[3] 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*, volume 44, page 2996, 2005.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *IEEE INFOCOM*, volume 3, pages 1653–1664, March 2005.

[5] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[6] T.G. Griffin and G. Wilfong. An analysis of BGP convergence properties. *ACM SIGCOMM Computer Communication Review*, 29(4):277–288, 1999.

[7] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin-American Mathematical Society*, 43(4):439, 2006.

[8] A. Jadbabaie, J. Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[9] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170. ACM New York, NY, USA, 2000.

[10] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, page 122. ACM, 2006.

[11] Y. Rekhter, T. Li, S. Hares, et al. RFC 1771: A border gateway protocol 4 (BGP-4), March 1995.

[12] Devavrat Shah. Gossip algorithms. *Foundations and Trends in Networking*, 3(1):1–125, 2009.

[13] L. Tassiulas and A. Ephremides. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Transactions on Automatic Control*, 37:1936–1948, Dec 1992.

[14] J.N. Tsitsiklis. *Problems in decentralized decision making and computation*. PhD thesis, M. I. T., Dept. of Electrical Engineering and Computer Science, 1984.

[15] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.