Foundations and Trends® in
Networking
Vol. 3, No. 1 (2008) 1–125

DOI: 10.1561/1300000014

now
the essence of knowledge

Gossip Algorithms

By Devavrat Shah

Contents

Foundations and Trends® in
Networking
Vol. 3, No. 1 (2008) 1–125

DOI: 10.1561/1300000014

now
the essence of knowledge

Gossip Algorithms

Devavrat Shah

Massachusetts Institute of Technology, Cambridge, MA, USA,
devavrat@mit.edu

Abstract

Unlike the Telephone network or the Internet, many of the next generation networks are not *engineered* for the purpose of providing efficient communication between various networked entities. Examples abound: sensor networks, peer-to-peer networks, mobile networks of vehicles and social networks. Indeed, these emerging networks do require algorithms for communication, computation, or merely spreading information. For example, estimation algorithms in sensor networks, broadcasting news through a peer-to-peer network, or viral advertising in a social network. These networks lack infrastructure; they exhibit unpredictable dynamics and they face stringent resource constraints. Therefore, algorithms operating within them need to be extremely simple, distributed, robust against networks dynamics, and efficient in resource utilization.

Gossip algorithms, as the name suggests, are built upon a *gossip* or *rumor* style unreliable, asynchronous information exchange protocol. Due to their immense simplicity and wide applicability, this class of algorithms has emerged as a canonical architectural solution for the next generation networks. This has led to exciting recent progress to understand the applicability as well as limitations of the Gossip

algorithms. In this review, we provide a systematic survey of many of these recent results on Gossip network algorithms. The algorithmic results described here utilize interdisciplinary tools from Markov chain theory, Optimization, Percolation, Random graphs, Spectral graph theory, and Coding.

1

Introduction

The twentieth century has seen a revolution in terms of our ability to communicate at very long distances at very high speeds. This has fundamentally changed the way we live in the present world. The development of reliable and high-performance massive communication networks has been at the heart of this revolution. The telephone networks and the Internet are prime examples of such large networks. These networks were carefully engineered (and are still being engineered) for the single purpose of providing efficient communication given the available resources. In contrast to these networks, there has been a sudden emergence of different types of large networks in the past few years where the primary purpose is not that of providing communication. Examples of such networks include sensor networks, peer-to-peer (P2P) networks, mobile ad-hoc networks, and social networks.

A sensor network, made of a large number of unreliable cheap sensors, is usually deployed for the purpose of 'sensing', 'detecting' or 'monitoring' certain events. For example, smoke sensors capable of wireless transmission deployed for smoke detection in a large building, or a collection of interconnected camera sensors deployed for surveillance in a secure facility. The ability to deploy such networks anywhere with

minimal cost of infrastucture has made them particularly attractive for these applications. Clearly, the primary purpose of such networks is to collect and process the sensed information by sensors rather than provide efficient communication.

The peer-to-peer networks are formed by connecting various users (e.g., computers or handheld devices) over an already existing network such as the Internet. Usually such networks are formed with minimal infrastructural support. The peers (or neighbors) are connected over an existing network and hence the advantage of using such networks is not in terms of efficiency of utilizing resources. However, a significant benefit arises in terms of reduced infrastructural support in situations like wide information dissemination. For example, in the absence of a P2P network an Internet content provider (e.g., BBC) needs to maintain a high bandwidth 'server farm' that 'streams' a popular movie or a TV show to a large number of users simultaneously. In contrast, in the presence of a P2P network a user is likely to obtain the desired popular content from a 'nearby' peer and thus distributing a large cost of 'streaming' from the 'server farm' to many 'peers'. Therefore, such an architecture can reduce the cost of content dissemination for a content provider drastically. Of course, it is likely to come at an increased cost of the network utilization. Now, whether or not the benefits obtained in terms of reduced infrastructure by utilizing P2P network for a content provider offset the increased network cost incurred by the network provider is indeed intriguing both in an engineering and an economic sense. While the recent trend suggests that it is indeed the case (e.g., advent of the BBCiPlayer [70] and adaptation of Korean ISPs [31]), the equilibrium solution is yet to be reached.

The mobile ad-hoc network formed between vehicles arises in various scenarios, including future smart cars traveling on road, or fleets of unmanned aerial vehicles deployed for surveillance. These networks, by design, are formed for a purpose other than communication. They need algorithms for the purpose of co-ordination, consensus or flocking (e.g., see classical work by Tsitsiklis [69], more recently [6, 32, 63]).

Finally, we have noticed a very recent emergence of massive social networks between individuals connected over a heterogenous collection of networks. Until recently, an individual's social network usually

involved only a small number of other acquintances, relatives or close friends. However, the arrival of 'social network applications' (e.g., Orkut, Facebook, etc.) has totally changed the structure of existing social networks. Specifically, the social network of an individual now includes many more acquintances than before thanks to these online applications. Furthermore, the use of handheld devices like smart phones are likely to create new ways to 'socialize' through P2P networks formed between them in the near future. Naturally, this 'globalization' and 'ubiquitous presence' of social networks bring many exciting opportunities along with extreme challenges. To realize these opportunities and to deal with the challenges, we will need new algorithms with efficient effective social communication under uncertain environmental conditions.

1.1 NextGen Networks: Through an Algorithmic Lens

Algorithms are key building blocks of any network architecture. For example, the Internet provides efficient communication between users through a collection of algorithms operating at the end-users and inside the network. Popular instances of such algorithms are the Transmission Control Protocol (TCP) for congestion control or Border Gateway Protocol (BGP) for routing. The above discussed emerging or next generation networks are not designed to provide efficient communication between the entities or the users networked by them. But, they do require algorithms to enable their primary applications. For example, a sensor network may require an estimation algorithm for event detection given the sensor observations; a P2P network may require a dissemination algorithm using peer information; a network of aerial vehicles may need an algorithm to reach consensus to co-ordinate their surveillance efforts, and an advertiser may need a social network algorithm for efficient 'viral' advertisement.

In most of these next generation networks, algorithms usually need to operate under an 'adverse' environment. First of all, since these networks are not built for providing communication, there is usually a lack of a reliable network infrastructure. Second, these networks are highly dynamic in the sense that nodes may join the network, leave the

network, or even become intermittently unavailable in an unpredictable manner. Third, the network is usually highly resource constrained in terms of communication, computation and sometimes energy resources.

The highly constrained environment in which algorithms are operating suggest that the algorithm must posses certain properties so as to be implementable in such networks. Specifically, an algorithm operating at a node of the network should utilize information 'local' to the node and should not expect any static infrastructure. It should attempt to achieve its task iteratively and by means of asynchronous message exchanges. The algorithm should be robust against the network dynamics and should not prescribe to any 'hard-wired' implementation. And finally, the algorithm should utilize minimal computational and communication resources by performing few logical operations per iteration as well as require light-weight data structures. These constraints naturally lead to 'Gossip' algorithms, formally described next, as a canonical algorithmic architectural solution for these next generation networks.

1.2 The Formal Agenda

We shall formally describe the quest for algorithm design for the next generation networks in this section. This will give rise to the formal definition of 'Gossip' algorithms, which will serve as the canonical algorithmic solution.

To this end, let us consider a network of n nodes denoted by $V = \{1, \ldots, n\}$. Let $E \subset V \times V$ denote the set of (bidirectional) links along which node pairs can communicate. That is, $(i, j) \in E$ if and only if nodes $i, j \in V$ can communicate with each other. Let this network graph be denoted by $G = (V, E)$. This network graph G should be thought of as changing over time in terms of V and E. As the reader will notice, the algorithms considered here will not utilize any static property of G and hence will be applicable in the presence of explicit network dynamics. For simplicity of the exposition, we shall not model the network dynamics explicitly. Let d_i denote the degree of node i in G, i.e., $d_i = |\{j \in V : (i, j) \in E\}|$. We will assume that the network G is connected without loss of generality; or else we can focus on different connected components separately.

We consider a class of algorithms, called 'Gossip' algorithms, that are operating at each of the n nodes of the network. Now, we present the formal definition of these algorithms.

Definition 1.1 (Gossip algorithms). Under a Gossip algorithm, the operation at any node $i \in V$, must satisfy the following properties:

(1) The algorithm should only utilize information obtained from its neighbors $\mathcal{N}(i) \stackrel{\triangle}{=} \{j \in V : (i,j) \in E\}$.
(2) The algorithm performs at most $O(d_i \log n)$ amount of computation per unit time.
(3) Let $|F_i|$ be the amount of storage required at node i to generate its output. Then the algorithm maintains $O(\text{poly}(\log n) + |F_i|)$ amount of storage at node i during its running.
(4) The algorithm does not require synchronization between node i and its neighbors, $\mathcal{N}(i)$.
(5) The eventual outcome of the algorithm is not affected by 'reasonable'[1] changes in $\mathcal{N}(i)$ during the course of running of the algorithm.

We wish to design Gossip algorithms for computing a generic network function. Specifically, let each node have some information, and let x_i denote the information of node $i \in V$. The node $i \in V$ wishes to compute a function $f_i(x_1, \ldots, x_n)$ using a Gossip algorithm. Also, it would like to obtain a good estimate of $f_i(x_1, \ldots, x_n)$ as quickly as possible. The question that is central to this survey is that of identifying the dependence of the computation time of the Gossip algorithm over the graph structure G and the functions of interest $f_1, \ldots, f_n$.

Before we embark on the description and organization of this survey, some remarks are in order. First, property (3) rules out 'trivial' algorithms like *first collect values $x_1, \ldots, x_n$ at each node and*

[1] By a reasonable change, here we mean dynamics that allow for a possibility of eventual computation of the desired function in a distributed manner. For example, if a node i becomes disconnected from the rest of the graph forever, then it will consist of unreasonable change as per our terminology.

then compute $f_i(x_1, \ldots, x_n)$ *locally* for functions like summation, i.e., $f_i(x_1, \ldots, x_n) = \sum_{k=1}^{n} x_k$. This is because for such a function the length of the output is $O(1)$ (we treat storage of each distinct number by unit space) and hence collection of all n items at node i would require storage $\Omega(n)$ which is a violation of property (3). Second, the computation of complex function (e.g., requiring beyond poly$(\log n)$ space) are beyond this class of algorithms. This is to reflect that the interest here is in functions that are easily computable, which is usually the case in the context of network applications. Third, the definition of a Gossip algorithm here should be interpreted as a rough guideline on the class of simple algorithms that are revelant rather than a very precise definition.

1.3 Organization

In the remainder of this survey, we provide a systematic description of the class of network functions that can be computed by means of a Gossip algorithm. A salient feature of the analysis of the algorithms described in this survey is the ability to describe the precise dependance of computation time on the network graph structure G and the function of interest. These dependancies are described in terms of 'spectral-like' graph properties. Therefore, we start with *Preliminaries* on graph properties and some known results that will be useful in the algorithm design and analysis. These are explained through examples of a collection of graph models throughout the survey.

The network functions for which we describe Gossip algorithms in this survey are naturally designed in a 'layered' fashion. At the bottom of the layer lies the design of a robust information layer using a Gossip algorithm. This is described in detail in *Information dissemination*. Here we will describe information dissemination Gossip algorithm for both unicast and multicast types of traffic scenarios. We will describe a natural relation between Percolation on graphs, information dissemination and certain spectral-like graph properties.

The simplest class of iterative algorithms, built upon an unreliable information layer, are based on linear dynamics. These algorithms have been used for solving consensus or multi-agent co-ordination problems

classically. We provide a detailed account on the optimal design and analysis of such algorithms in *Linear computation*. Here, we shall describe the interplay between Markov chain theory, mixing times and Gossip algorithms. We also report some advances in the context of Markov chain theory due to considerations from the viewpoint of Gossip algorithms.

Linear function computation is an instance of, and essentially equivalent to, separable function computation. The quest for designing the fastest possible Gossip algorithm, in terms of its dependence on the graph structure, for separable function computation, which will be left partly unresolved by the linear dynamics based algorithms, will be brought to a conclusion in *Separable function computation*. Here, we shall describe an algorithm based on an 'extremal' property of the Exponential distribution. This algorithm will utilize the unreliable information layer designed in *Information dissemination* for the purpose of information exchange. The appropriately quantized version of this algorithm as well as information theoretic arguments suggesting its fundamental optimality will be discussed (see 'Summary') as well.

Next, we consider Gossip algorithm design for the task of scheduling in constrained queueing networks. This is a key operational question for networks such as those operating over a common wireless medium. For such a network a scheduling algorithm is required for the media access control (MAC). We describe Gossip scheduling algorithm in *Network scheduling*. This algorithm builds upon the separable function computation algorithm using clever randomization.

Network resource allocation is another fundamental problem that is faced while operating a communication network. Under flow-level modeling of a network, this involves solving certain network-wide or global constrained convex optimization problems. Therefore, we consider the question of designing a Gossip algorithm for a class of convex optimization problems in *Network convex optimization*. This algorithm, like network scheduling, builds upon the separable function computation algorithm. Specifically, it utilizes the separable function computation algorithm to design a 'distributed computation' layer.

In summary, the algorithms presented in this survey provide 'layers' of computation in a network. The key reason for the existence of such

a 'layered' algorithmic architecture lies in the ability to 'functionally decompose' many interesting problems with separable function computation central to the decomposition. For this reason, Gossip algorithm for separable function computation becomes a key 'subroutine' in designing Gossip algorithms for many seemingly complex network computation problems. For these reasons, in addition to applications described in this survey, the separable function computation algorithm can be used to design Gossip algorithms for other important applications including spectral decomposition (using the algorithm of Kempe and McSherry [38] and the separable function computation algorithm) and Kalman filtering.

2

Preliminaries

2.1 Graphs and Random Walks

We will introduce notations, definitions and some known results that will be useful throughout the paper. We start with very basic notions. An object that will play central role is the graph. A graph consists of some finite number, say n, of nodes, which will be numbered and represented as a vertex set $V = \{1, \ldots, n\}$; edges representing connections between the nodes are denoted by $E \subset V \times V$. Thus, a graph denoted by G will be defined by sets V and E and will be represented as $G = (V, E)$. We will assume a graph to be undirected, i.e., if $(i, j) \in E$ then $(j, i) \in E$ as well.

A node $i \in V$ has neighbors $\mathcal{N}(i) \stackrel{\Delta}{=} \{j \in V : (i, j) \in E\}$. The degree of node i, denoted by d_i is defined as $d_i = |\mathcal{N}(i)|$. A path between two nodes $i \neq j$ is denoted by a collection of edges $(u_k, u_{k+1}) \in E, 0 \leq k < \ell$ for some $\ell \geq 1$, where $u_0 = i$ and $u_\ell = j$. The length of a path is defined as the number of edges that belong to the path. A path between two nodes of the shortest length is called a shortest path. If there is a path between any two nodes in V then we call the graph connected. Any connected undirected graph induces a finite valued metric on nodes in

V as follows: define metric $\mathbf{d}_G : V \times V \to \mathbf{N}$ by assigning the length of shortest path between nodes $i \neq j \in V$ as $\mathbf{d}_G(i,j)$ and $\mathbf{d}_G(i,i) = 0$ for all $i \in V$. Diameter D of a connected graph G is defined as

$$D = \max_{i,j \in V} \mathbf{d}_G(i,j).$$

A random walk on a graph $G = (V,E)$ or equivalently a Markov chain with its states represented by V and transitions represented by E, is defined by an $n \times n$ non-negative valued probability transition matrix $P = [P_{ij}]$, where P_{ij} is the probability of transition from state or node i to j. We shall use the terms random walk or Markov chain associated with graph G and/or probability matrix P interchangeably throughout.

Now by definition, the probability matrix $P \in \mathbf{R}_+^{n \times n}$ must satisfy

$$\sum_{j=1}^{n} P_{ij} = 1, \quad \forall\ i \in V.$$

Further, it is graph G conformant, i.e., if $(i,j) \notin E$ then $P_{ij} = P_{ji} = 0$. A matrix P is called irreducible if for any $i \neq j \in V$, there exists a path $(u_k, u_{k+1}) \in E, 0 \leq k < \ell$ for some $\ell \geq 1$ with $u_0 = i, u_\ell = j$ and $\prod_{k=0}^{\ell-1} P_{u_k u_{k+1}} > 0$. The matrix P is said to have period d if V can be decomposed into a disjoint union of d non-empty sets $V_0, \ldots, V_{d-1}$, i.e., $V = \cup_{k=0}^{d-1} V_k; V_k \cap V_{k'} = \emptyset$, for $0 \leq k \neq k' < d$ and for any $i \in V_k$, if $P_{ij} > 0$ then $j \in V_{(k+1) \bmod d}$. P is called aperiodic if it does not have periods larger than 1. Throughout, we will be interested in P that are irreducible and aperiodic unless specified otherwise.

A probability distribution $\pi = [\pi_i] \in \mathbf{R}_+^n$ is called a stationary distribution of probability matrix P if $\pi^T P = \pi^T$ or equivalently if it satisfies the balance equations

$$\pi_j = \sum_{i=1}^{n} \pi_i P_{ij}, \quad \forall\ j \in V.$$

By the Perron–Frobenius Theorem (see book by Horn and Johnson [30]) it follows that any irreducible and aperiod P has a unique stationary distribution π with all components being strictly positive. Further, for such a P if we consider its tth power, P^t, then $P_{ij}^t \to \pi_j$ as

$t \to \infty$, for all $i, j \in V$. A matrix P with such a convergence property is called 'ergodic'. For an ergodic random walk, we define the notion of an ergodic flow. Specifically, it is an $n \times n$ matrix $Q = [Q_{ij}]$ where $Q_{ij} = \pi_i P_{ij}$. For an ergodic random walk ergodic flow Q defines P and π uniquely. This is because, $\pi_i = \sum_j \pi_j P_{ji} = \sum_j Q_{ji}$. And if π, Q are known then $P_{ij} = Q_{ij}/\pi_j$.

Finally, we classify irreducible and aperiodic Markov chains into two classes: reversible and non-reversible. A Markov chain or random walk with transition matrix P and stationary distribution π is called reversible if $\pi_i P_{ij} = \pi_j P_{ji}$ for all $i, j \in V$. A Markov chain that is not reversible is called non-reversible. A special class of reversible Markov chains are those with symmetric P, i.e., $P_{ij} = P_{ji}$ for all $i, j \in V$. For such a P, it can be easily checked that $\pi = (1/n)\mathbf{1} = [1/n]$. That is,

$$\mathbf{1}^T P = \mathbf{1}^T \quad \text{and} \quad P\mathbf{1} = \mathbf{1}.$$

2.2 Mixing Time and Conductance

As noted above, for any irreducible and aperiodic P, P^t converges to a matrix with all of its rows equal to π^T. The question of interest in the context of algorithms based on random walk (such as the ones we will consider throughout) is that of determining the rate at which this convergence happens. More precisely, for a given $\varepsilon > 0$ we will be interested in finding out how large a t is needed so that P^t is ε close to this eventual matrix in some 'norm'. This is formalized in terms of the 'mixing time' of the random walk based on P.

Specifically, we will introduce two seemingly different but closely related definitions of mixing times. As we shall see, both of them are closely related and will be useful in characterizing mixing times for different situations depending upon the 'type' of P. First, we introduce a notion of mixing time based on a 'total variation' distance between distributions.

Definition 2.1 (Mixing time: total variation). For a random walk (or Markov chain) with transition matrix P and stationary distribution $\pi = [\pi_i]$, let $\Delta_i(t) = \frac{1}{2}\sum_{j=1}^{n} |P_{ij}^t - \pi_j|$. Then, the ε-mixing time is

defined as

$$\tau(\varepsilon, P) = \max_i \inf\{t{:}\Delta_i(s) \leq \varepsilon,\ \forall\ s \geq t\}. \tag{2.1}$$

The other notion of a mixing time is based on stopping rules. A stopping rule Γ is a stopping time based on the random walk of P: at any time, it decides whether to stop or not, depending on the walk seen so far and possibly additional randomness (or coin flips). Suppose the starting node w^0 is drawn from distribution σ. The distribution of the stopping node w^Γ is denoted by $\sigma^\Gamma = \tau$ and call Γ the stopping rule from σ to τ. Let $\mathcal{H}(\sigma, \tau)$ be the infimum of mean length over all such stopping rules from σ to τ. This is well-defined as there exists the following stopping rule from σ to τ: *select i with probability τ_i and walk until getting to i.*

Definition 2.2(Mixing time: Stopping rule). Given Markov chain P with stationary distribution π, the stopping rule based mixing time $\mathcal{H}(P)$ is defined as:

$$\mathcal{H}(P) = \max_\sigma \mathcal{H}(\sigma, \pi),$$

where σ is over the space of all distributions on V.

A related important notion is that of 'conductance'. Given Markov chain with transition matrix P and stationary distribution π, its conductance $\Phi(P)$ is defined as

$$\Phi(P) = \min_{S \subset V} \frac{\sum_{i \in S, j \in V \setminus S} \pi_i P_{ij}}{\pi(S)\pi(V \setminus S)}, \tag{2.2}$$

where $\pi(A) = \sum_{i \in A} \pi_i$.

Some remarks about $\Phi(P)$ are in order. For symmetric P, the stationary distribution is uniform, i.e., $\pi = (1/n)\mathbf{1}$. Further, symmetry of P implies that $\sum_{i \in S, j \in S^c} P_{ij} = \sum_{i \in S, j \in S^c} P_{ji}$. Using these two properties, the $\Phi(P)$ for such symmetric P simplifies to

$$\Phi(P) = \min_{S \subset V : |S| \leq n/2} \frac{\sum_{i \in S, j \in S^c} P_{ij}}{|S|}. \tag{2.3}$$

In the context of symmetric P (which will be the case in many scenarios considered), we will use the term 'conductance' for (2.3) instead of (2.2).

2.2.1 Techniques: Characterizing Mixing Time

There are many algebraic, analytic, and combinatorial techniques available to characterize the mixing time, either based on total variation or stopping time for Markov chains. Here, we list a set of known results that will be useful throughout. To learn details of these results/techniques as well as a host of other results/techniques we refer an interested reader to the excellent surveys by Lovasz and Winkler [43], Montenegro and Tetali [52].

First, we state a result that relates $\tau(\varepsilon, P)$ and $\mathcal{H}(P)$. Specifically, for any $\varepsilon > 0$, we have

$$\tau(\varepsilon, P) = O\left(\mathcal{H}(P) \log \frac{1}{\varepsilon}\right).$$

Next, we present a bound on $\tau(\varepsilon, P)$ in terms of eigenvalues. If P is reversible, then one can view P as a self-adjoint operator on a suitable inner product space and this permits us to use the well-understood spectral theory of self-adjoint operators. It is well-known that P has $n = |V|$ real eigenvalues $1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-1} > -1$. Then for any $\varepsilon > 0$, $\tau(\varepsilon, P)$ is bounded as follows:

$$\frac{\lambda_P}{2(1-\lambda_P)} \log \frac{1}{2\varepsilon} \leq \tau(\varepsilon, P) \leq \left\lceil \frac{1}{1-\lambda_P} \log \frac{1}{\varepsilon \pi_0} \right\rceil,$$

where $\lambda_P = \max\{|\lambda_1|, |\lambda_{n-1}|\}$ and $\pi_0 = \min_i \pi_i$. The $1 - \lambda_P$ is also called the spectral gap of reversible matrix P.

When, P is non-reversible we consider PP^* where P^* is the adjoint of P with respect to the appropriate inner product space. For P with uniform stationary distribution, $P^* = P^T$. It follows by definition that the Markov chain with PP^* as transition matrix is reversible. Let $1 - \lambda_{PP^*}$ be the spectral gap of this reversible Markov chain. Then, for any $\varepsilon > 0$, the ε-mixing time of the original Markov chain (with transition matrix P) is bounded above as

$$\tau(\varepsilon, P) \leq \left\lceil \frac{2}{1-\lambda_{PP^*}} \log \frac{n}{\varepsilon \sqrt{\pi_0}} \right\rceil. \tag{2.4}$$

Now, we present a technique known as 'the fill-up lemma' (see [1]). It will be useful to bound the stopping rule based mixing time, $\mathcal{H}(P)$

in certain scenarios involving non-reversible random walks. This is particularly useful when it is difficult to design an exact stopping rule with small average time-length. In that situation, the 'fill-up lemma' suggests the following two step approach.

Step 1. For a positive constant $\delta > 0$ and any starting distribution σ, design a stopping rule whose stopping distribution γ is δ-far from π in the sense that $\gamma \geq (1-\delta)\pi$, with inequality holding component-wise. Such a construction by definition provides an upper bound on $\mathcal{H}(\sigma, \gamma)$.

Step 2. Now, the $\mathcal{H}$ can be bounded using $\mathcal{H}(\sigma, \gamma)$ for such a choice of γ, as

$$\mathcal{H} \leq \frac{1}{1-\delta} \mathcal{H}_{\underline{\delta}},$$

where $\mathcal{H}_{\underline{\delta}} = \max_{\sigma} \min_{\gamma \geq (1-\delta)\pi} \mathcal{H}(\sigma, \gamma)$. That is, $\mathcal{H} \leq \frac{1}{1-\delta}$ $\mathcal{H}(\sigma, \gamma)$.

Finally, we provide a relation between mixing time and the conductance. For any P, the stopping rule based mixing time, $\mathcal{H}(P)$ is bounded as

$$\frac{1}{\Phi(P)} \leq \mathcal{H}(P) \leq O\left(\frac{\log n}{\Phi^2(P)}\right).$$

Let us restrict P to be reversible and let us assume that $\lambda_P = \lambda_1$ as per above notation. This is not a restrictive assumption because for any P, the Markov chain with transition matrix $(I+P)/2$ has this property and it can be easily checked that the mixing time of P and that of $(I+P)/2$ differ only by a constant factor. Now for such a P, the conductance $\Phi(P)$ bounds the spectral gap, $1-\lambda_1$ as

$$\frac{\Phi^2(P)}{2} \leq 1 - \lambda_1 \leq 2\Phi(P).$$

2.2.2 A Related Notion: *k*-Conductance

Here, we introduce a notion related to 'conductance' $\Phi(P)$ of a probability matrix P. We define this notion building upon the simplification achieved in (2.3) for symmetric matrices. Specifically, for any

$1 \leq k \leq n-1$ we define k-conductance $\Phi_k(P)$ as

$$\Phi_k(P) = \min_{S \subset V : |S| \leq k} \frac{\sum_{i \in S, j \in S^c} P_{ij}}{|S|}. \tag{2.5}$$

The notion of k-conductance will be utilized to study information spreading algorithms.

2.3 Some Graph Models

Here, we describe various network graph models that are used in applications. We shall use these models for illustration purposes.

2.3.1 Complete Graph

The complete graph represents situation when all nodes can communicate with each other. That is, essentially no graphical communication constrains. In a sense, the complete graph can be used to provide absolute bound on the best performance an algorithm can achieve when the graph structure can affect performance of the algorithm. In that sense, the complete graph serves as an important conceptual graphical structure.

For a complete graph of n nodes, a natural symmetric probability matrix is $P = [1/n]$, i.e., all entries being equal to $1/n$. For such a matrix, it is easy to check that it is irreducible, aperiodic and hence ergodic. It has a uniform stationary distribution. The mixing time of P is 1 since for any starting distribution, within 1 step the distribution becomes uniform. Note that this applies to both definitions of mixing time.

Let us consider the conductance of P, $\Phi(P)$ defined by (2.3). For any $S \subset V$ with $|S| = k \leq n/2$, we have

$$\frac{\sum_{i \in S, j \in S^c} P_{ij}}{|S|} = \frac{\frac{1}{n}|S||S^c|}{|S|} = \frac{n-k}{n}.$$

Therefore, in order to minimize the above for $k \leq n/2$, we should choose $k = \lfloor n/2 \rfloor$ and this yields that

$$\Phi(P) = \frac{\lceil n/2 \rceil}{n} \approx 1/2.$$

Now, we compute the k-conductance. Using the same calculations as above, but in place of $n/2$ if we use k, we obtain

$$\Phi_k(P) \approx 1 - \frac{k}{n}.$$

We shall be interested in the following 'harmonic' like mean of $\Phi_k(P)$ for $1 \leq k < n$, denoted by $\hat{\Phi}(P)$ and defined as

$$\hat{\Phi}(P) = \sum_{k=1}^{n-1} \frac{k}{\Phi_k(P)}.$$

Then, for the above described P for the complete graph, we have

$$\hat{\Phi}(P) = \sum_{k=1}^{n-1} \frac{kn}{n-k} = \Theta(n^2 \log n).$$

2.3.2 Ring Graph

The ring graph of n nodes is formed by placing n nodes on a circle (or a ring) and connecting each node to two of its nearest neighbors. This is essentially the most communication constrained graph. Qualitatively, complete graph represents one end of the spectrum of graphs while the ring graph presents the other end. In a sense, the complete graph has no 'geometry' while the ring graph has the 'strongest' possible 'geometry'. For this reason, in a sense ring graph provides an absolute bound on the worse performance a graph algorithm can achieve. Therefore, the ring graph serves as another important conceptual graphical structure.

For a ring graph of n nodes, a natural symmetric probability matrix P is as follows: for each i, $P_{ii} = 1/2$, $P_{ii^+} = P_{ii^-} = 1/4$ where i^+ and i^- represent neighbors of i on either side. The mixing time $\tau(\varepsilon, P)$ is $\Omega(n^2)$ and $O(n^2 \log n/\varepsilon)$. It can be checked that the conductance for such P is $\Phi(P) = \Theta(1/n)$. The k-conductance is $\Phi_k(P) = \Theta(1/k)$. Therefore, it follows that $\hat{\Phi}(P) = \Theta(n^3)$.

2.3.3 Expander Graph

Informally, an expander graph refers to a class of graphs indexed by the number of nodes that have good 'expansion' property. There are various equivalent definitions for expander graphs. Here, we will define

expander graphs in terms of the graph conformant ergodic probability matrix P. Specifically, we call a sequence indexed by the number of nodes n of random walk with probability transition matrix P over graph G as 'expanding' (or G, P as expander) if there exists $\delta > 0$, independent of n so that $\Phi(P) \geq \delta$. This will imply that the mixing time of P essentially scales as $O(\log n)$, i.e., the random walk mixes very fast. There are various popular network models that posses this property. Specific examples include the Preferential Connectivity Model for the Internet graph and the Small World Model for social network (for certain range of parameters) with 'natural' random walk.

A special class of expander graphs that we will use for the purpose of illustration throughout is the d-regular expander for $d \geq 3$. These are graph sequences such that for any number of nodes n, each node has degree d, where d is independent of n. For any graph in this sequence, it has good 'expansion' property. Specifically, for any subset $S \subset V$ of size at most $n/2$ there exists $\alpha > 0$ (independent of n) such that

$$E(S, S^c) \geq \alpha |S|,$$

where $E(S, S^c) = |\{(i, j) \in E \colon i \in S,\ j \in S^c\}|$. For such a graph consider a natural P defined as follows:

$$P_{ij} = \begin{cases} \frac{1}{2}, & \text{if } i = j \\ \frac{1}{2d}, & \text{if } j \in \mathcal{N}(i) \\ 0, & \text{otherwise.} \end{cases}$$

Such a P is symmetric and has a uniform stationary distribution. Consider the conductance of P:

$$\begin{aligned} \Phi(P) &= \min_{S \subset V : |S| \leq n/2} \frac{\sum_{i \in S; j \in S^c} P_{ij}}{|S|} \\ &= \min_{S \subset V : |S| \leq n/2} \frac{\sum_{i \in S; j \in S^c} \frac{1}{2d}}{|S|} \\ &= \frac{1}{2d} \min_{S \subset V : |S| \leq n/2} \frac{E(S, S^c)}{|S|} \\ &\geq \min_{S \subset V : |S| \leq n/2} \frac{\alpha |S|}{2d|S|} \ \geq \ \frac{\alpha}{2d}. \end{aligned} \tag{2.6}$$

Thus, the natural P has conductance which is at least $\alpha/2d > 0$, independent of n. Thus, such a graph with natural probability matrix P described above is indeed an expander as per our definition.

Now, we compute the k-conductance. Using the same calculations as above, but in place of $n/2$ if we use k, we obtain for $k \leq n/2$

$$\Phi_k(P) \geq \frac{\alpha}{2d} \;\Rightarrow\; \Phi_k(P) = \Theta(1).$$

For $k > n/2$, consider the following. For S such that $|S| = k > n/2$

$$E(S, S^c) = E(S^c, S) \geq \alpha |S^c| = \alpha(n - |S|) = \alpha(n - k).$$

Using this, it follows that for $k > n/2$,

$$\Phi_k(P) \geq \frac{\alpha}{2d} \frac{n-k}{k} = \Omega\left(\frac{n-k}{k}\right).$$

Then, calculations similar to those done for the complete graph imply that

$$\hat{\Phi}(P) = O(n^2 \log n).$$

Existence of d-regular expanders was first established by Pinsker by means of a probabilistic argument: for $d \geq 3$, random d-regular connected graphs are expanders with positive probability. Now, we know various explicit constructions of such graphs. For example, the Zig Zag construction of d-regular expander by Reingold et al. [61].

2.3.4 Geometric Random Graph

The Geometric Random Graph has been used successfully to model ad-hoc wireless networks. A d-dimensional Geometric Random Graph of n nodes, modeling a wireless ad-hoc network of n nodes with wireless transmission radius r, denoted as $G^d(n, r)$ is obtained as follows: place n nodes on a d-dimensional unit cube uniformly at random and connect any two nodes that are within distance r of each other.

An example of a two dimensional graph, $G^2(n, r)$ is shown in Figure 2.1. The following is a well-known threshold result about the

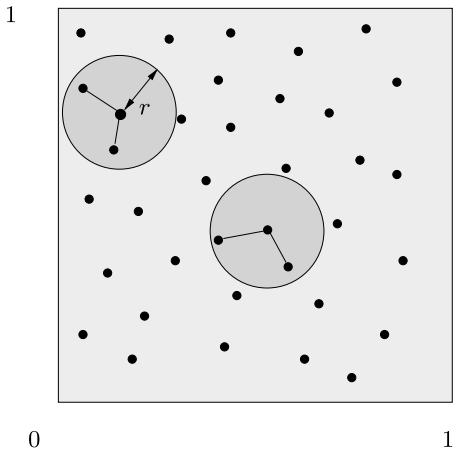

Fig. 2.1 An example of a Geometric Random Graph in two-dimensions. A node is connected to all other nodes that are within distance r of itself.

connectivity of $G^d(n,r)$ (for a proof, see [19, 28, 58]):

Lemma 2.1. There exists a constant $\alpha_d > 0$ such that for any $\varepsilon > 0$, if $(nr^d(n)/\log n) \geq \alpha_d + \varepsilon$ then the $G^d(n,r(n))$ is connected with probability $1 - o(1)$ and if $(nr^d(n))/\log n \leq \alpha_d - \varepsilon$ then $G^d(n,r(n))$ is not connected with probability $1 - o(1)$.

Here, we shall consider $r(n)$ such that $(nr^d(n))/\log n = \omega(1)$, i.e., $G(n,r(n))$ is connected with high probability. Let $d_{\max}$ and $d_{\min}$ denote, respectively, the maximum and minimum vertex degree of such a $G(n,r(n))$. Then, it can be argued that with probability $1 - o(1)$,

$$d_{\max} = (1 + o(1))d_{\min}.$$

That is, $G(n,r(n))$ is almost regular. Therefore, we can define the natural random walk on $G(n,r(n))$ with transition matrix P where

$$P_{ij} = \begin{cases} \frac{1}{2}, & \text{if } i = j, \\ \frac{1}{2d_i}, & \text{if } j \in \mathcal{N}(i), \\ 0, & \text{otherwise.} \end{cases}$$

Clearly P is aperiodic (due to self-loop) and irreducible (since $G(n, r(n))$ is connected) with probability $1 - o(1)$. Let π be the stationary distribution of such a random walk P. Then, it follows that $\pi_i = (1 + o(1))/n$ with probability $1 - o(1)$. It has been established that the mixing time of such a random walk is

$$\tau(\varepsilon, P) = \Omega(r(n)^{-2}) \quad \text{and} \quad \tau(\varepsilon, P) = O(r(n)^{-2} \log n/\varepsilon).$$

Further, it has been established that the fastest mixing reversible random walk on $G(n, r(n))$ with uniform stationary distribution has mixing time no faster than $r(n)^{-2}$. That is, the natural random walk over $G(n, r)$ has the mixing time of the same order as that of the random walk with the fastest mixing time. We note that for the natural random walk with probability matrix P, and for $r(n) = \Theta(\log^{3/2} n/\sqrt{n})$ with a large enough constant, the conductance $\Phi(P)$ scales like $\Theta^*(r(n))$.

2.4 Historical Notes

We presented preliminaries about random walks, mixing times and graph models. The theory of mixing time is quite well developed. Various results are reported here and many others not reported here can be found in excellent surveys by Lovasz and Winkler [43] and by Montenegro and Tetali [52]. We make note of a result about the characterization of the fastest mixing reversible random walk on a given graph in terms of an appropriately defined Semi-Definite Program (SDP) by Boyd et al. [7] due to its applicability in proving bounds on mixing times for various reversible random walks. For mixing time analysis of the natural random walk on a ring graph and on a Geometric random graph we refer the reader to the work by Boyd et al. [8]. For evaluation of conductance on a ring and on a Geometric random graph we refer the reader to work by Madan et al. [45]. For various expander related results we refer the reader to a set of lecture notes by Linial and Widgerson [42].

3

Information Dissemination

3.1 Setup

We are given a connected network of n nodes with a connectivity graph $G = (V, E)$. Each node, say $i \in V$, has its own information denoted by x_i. The information x_i can be thought of as a collection of bits, a packet or even a real number. The interest is in spreading or disseminating this information to possibly different subsets of nodes in V. We will consider two special scenarios motivated by applications to distributed computation, estimation in sensor networks, and content distribution in peer-to-peer networks.

The first scenario consists of a situation when exactly one of the n nodes wishes to spread its information to all the remaining $n - 1$ nodes. We will denote this scenario of single multicast as *Single-piece dissemination*. Formally, let an arbitrarily selected node, say $i \in V$, has a piece of information. The goal is to spread this information to all nodes in V as quickly as possible by means of a Gossip algorithm. This situation naturally arises in a sensor network deployed for some event detection such as 'smoke detection' in a building — one (or few) sensors detect smoke and they wish to 'alarm' all the sensors in the

building as quickly as possible for evacuation. Another such situation is that of content distribution over a peer-to-peer network where content is arising from a large content distributor. For example, the use of the BBCiplayer by BBC [70]. Finally, an important 'sub-routine' in many distributed computation problems corresponds to the single-piece information dissemination scenario. This is explained in detail in *Separable function computation*.

The second scenario corresponds to a situation where all nodes have their individual pieces of information and each node wishes to disseminate its own information to all the other nodes. This all-to-all multicast scenario will be denoted by *multi-piece dissemination*. Again, the goal will be to disseminate all information to all nodes as quickly as possible by means of a Gossip algorithm. This situation arises in content distribution over a peer-to-peer network when content is generated not only by one (or few) distributors like BBC, but also by many users. This is the case for advertisement over a P2P network where many advertizers have their own information that they wish to disseminate very widely.

In what follows, we shall describe natural Gossip algorithms for both of the above scenarios. We will charcterize the performance of these algorithms in terms of certain spectral-like properties of the underlying network graph G. This will suggest that Gossip protocols are quite efficient in network resource utilization for such information dissemination.

3.2 Single-Piece Dissemination

Given the network $G = (V, E)$ of n nodes, an arbitrary node $v \in V$ has a piece of information that it wishes to spread to all the other nodes as quickly as possible. We will describe a natural randomized Gossip algorithm which is very much like 'rumor mongering'.

To this end, let $P = [P_{ij}]$ be an $n \times n$ graph G conformant doubly stochastic and symmetric matrix. The algorithm utilizing P is as follows. Let time be discrete and denoted by $t \in \mathbf{N}$. Let $S(t) \subset V$ denote the set of nodes that contain node v's information at time t. Initially, $t = 0$ and $S(0) = \{v\}$. For information spreading at time $t \geq 1$, each node $i \in V$ contacts one of its neighbors, say j with probability P_{ij};

it will not contact any other node with probability P_{ii}. Upon contacting, if either i or j had v's information at time $t-1$, then both will have v's information at the end of time t.

A few remarks about the above described algorithm. First, each node can contact at most one other node; but it can be contacted by more than one node as part of the algorithm. Second, a node can spread v's information at time t only if it has received it by time $t-1$. Third, the information exchange protocol has both 'pull' and 'push' components, i.e., when two nodes are connected information is exchanged in both directions. As will become clear from analysis, the presence of both 'pull' and 'push' is necessary for quicker dissemination.

Now the quantity of interest. For any $\varepsilon > 0$, we wish to find the time by which all nodes have node v's information with probability at least $1-\varepsilon$, for any $v \in V$. Formally, the ε-dissemination time, denoted by $T^{\text{one}}_{\text{spr}}(\varepsilon)$ is defined as

$$T^{\text{one}}_{\text{spr}}(\varepsilon) = \sup_{v \in V} \inf \{t \colon \Pr(S(t) \neq V \,|\, S(0) = \{v\}) \leq \varepsilon\}.$$

Recall the definition of the 'conductance' of a symmetric, doubly stochastic matrix P, denoted by $\Phi(P)$, from *Preliminaries*:

$$\Phi(P) = \min_{S \subset V : |S| \leq n/2} \frac{\sum_{i \in S; j \in S^c} P_{ij}}{|S|}.$$

Now, we are ready to state the characterization of the spreading time of the above described natural Gossip algorithm based on P.

Theorem 3.1. Let P be an irreducible, doubly stochastic and symmetric matrix on graph G. Then, for the natural Gossip algorithm described above

$$T^{\text{one}}_{\text{spr}}(\varepsilon) = O\left(\frac{\log n + \log \varepsilon^{-1}}{\Phi(P)}\right).$$

3.2.1 Proof of Theorem 3.1

The proof is divided into two phases of the algorithm. The first phase corresponds to the time period starting at $t = 0$ until $|S(t)| \leq n/2$.

The second phase corresponds to the time period starting after the end of the first phase and until $S(t) = V$. We will show, starting with $S(0) = \{v\}$ for any $v \in V$, that each phase is at most $O(\log n + \log \varepsilon^{-1})/\Phi(P)$ with probability at least $1 - \varepsilon/2$. This will complete the proof of Theorem 3.1. We remark that, the bound on the time length of the first phase essentially relies on the 'push' aspect of the algorithm while the bound on the time length of the second phase essentially relies on the 'pull' aspect of the algorithm.

3.2.1.1 Phase 1: $|S(t)| \leq n/2$

Fix the node $v \in V$ with $S(0) = \{v\}$. We will study the evolution of the size of the set $S(t)$ containing the information of v. Here, we are concerned with the first phase, i.e., the time duration when $|S(t)| \leq n/2$. Through the time-evolution of $S(t)$, we will bound the length of the first phase. As stated above, we will ignore the effect due to the *pull* aspect of the algorithm in the first phase. It can be easily argued that ignoring the pull aspect can only yield a weaker bound. For notational simplicity, we will use Φ instead of $\Phi(P)$ by dropping reference to P.

Now, we want to study increase $|S(t+1)| - |S(t)|$ at time t, when $|S(t)| \leq n/2$. For $j \notin S(t)$, let X_j be an indicator random variable that is 1 if node j receives the information of v via a 'push' from some node $i \in S(t)$ in time slot $t+1$, and is 0 otherwise. The probability that j does not receive the information via a push is the probability that no node $i \in S(t)$ contacts j, and so

$$\begin{aligned}
\mathbf{E}[X_j \mid S(t)] &= 1 - \Pr(X_j = 0 \mid S(t)) \\
&= 1 - \prod_{i \in S(t)} (1 - P_{ij}) \\
&\geq 1 - \prod_{i \in S(t)} \exp(-P_{ij}) \\
&= 1 - \exp\left(- \sum_{i \in S(t)} P_{ij} \right). \qquad (3.1)
\end{aligned}$$

In the above equation, we used the inequality $1 - x \leq \exp(-x)$ for $x \geq 0$. The Taylor series expansion of $\exp(-z)$ about $z = 0$ implies that,

if $0 \le z \le 1$, then

$$\exp(-z) \le 1 - z + z^2/2 \le 1 - z + z/2 = 1 - z/2. \tag{3.2}$$

For a doubly stochastic matrix P, we have $0 \le \sum_{i \in S(t)} P_{ij} \le 1$, and so we can combine Eqs. (3.1) and (3.2) to obtain

$$\mathbf{E}[X_j \mid S(t)] \ge \frac{1}{2} \sum_{i \in S(t)} P_{ij}.$$

By linearity of expectation,

$$\begin{aligned} \mathbf{E}[|S(t+1)| - |S(t)| \mid S(t)] &= \sum_{j \notin S(t)} \mathbf{E}[X_j \mid S(t)] \\ &\ge \frac{1}{2} \sum_{i \in S(t), j \notin S(t)} P_{ij} \\ &= \frac{|S(t)|}{2} \frac{\sum_{i \in S(t), j \notin S(t)} P_{ij}}{|S(t)|}. \end{aligned}$$

When $|S(t)| \le n/2$, we have

$$\mathbf{E}[|S(t+1)| - |S(t)| \mid S(t)] \ge |S(t)| \frac{\Phi(P)}{2} \triangleq |S(t)| \frac{\Phi}{2}. \tag{3.3}$$

Using (3.3), we seek to upper bound the duration of the first phase. To this end, let

$$Z(t) = \frac{\exp\left(\frac{\Phi}{8} t\right)}{|S(t)|}.$$

Define the stopping time $L = \inf\{t \colon |S(t)| > n/2\}$, and $L \wedge t = \min(L, t)$. If $|S(t)| > n/2$, then $L \wedge (t+1) = L \wedge t$, and thus

$$\mathbf{E}[Z(L \wedge (t+1)) \mid S(L \wedge t)] = Z(L \wedge t).$$

Now, suppose that $|S(t)| \le n/2$, in which case $L \wedge (t+1) = (L \wedge t) + 1$. The function $g(z) = 1/z$ is convex for $z > 0$, which implies that, for $z_1, z_2 > 0$,

$$g(z_2) \ge g(z_1) + g'(z_1)(z_2 - z_1). \tag{3.4}$$

Applying (3.4) with $z_1 = |S(t+1)|$ and $z_2 = |S(t)|$ yields

$$\frac{1}{|S(t+1)|} \leq \frac{1}{|S(t)|} - \frac{1}{|S(t+1)|^2}(|S(t+1)| - |S(t)|). \quad (3.5)$$

We have ignored the effect of the 'pull' aspect in this phase to obtain bound on its time duration (it only worsens the bound). And due to the 'push', it easily follows that $|S(t+1)| \leq 2|S(t)|$. Therefore

$$\frac{1}{|S(t+1)|} \leq \frac{1}{|S(t)|} - \frac{1}{4|S(t)|^2}(|S(t+1)| - |S(t)|). \quad (3.6)$$

Combining (3.3) and (3.6), and using the fact that $1 - z \leq \exp(-z)$ for $z \geq 0$, we obtain that, if $|S(t)| \leq n/2$, then

$$\begin{aligned}\mathbf{E}\left[\frac{1}{|S(t+1)|} \,\middle|\, S(t)\right] &\leq \frac{1}{|S(t)|}\left(1 - \frac{\Phi}{8}\right) \\ &\leq \frac{1}{|S(t)|}\exp\left(-\frac{\Phi}{8}\right).\end{aligned}$$

This implies that

$$\begin{aligned}\mathbf{E}[Z(L \wedge (t+1)) \mid S(L \wedge t)] &= \mathbf{E}\left[\frac{\exp\left(\frac{\Phi}{8}(L \wedge (t+1))\right)}{|S(L \wedge (t+1))|} \,\middle|\, S(L \wedge t)\right] \\ &= \exp\left(\frac{\Phi}{8}(L \wedge t)\right)\exp\left(\frac{\Phi}{8}\right) \\ &\quad \times \mathbf{E}\left[\frac{1}{|S((L \wedge t)+1)|} \,\middle|\, S(L \wedge t)\right] \\ &\leq \exp\left(\frac{\Phi}{8}(L \wedge t)\right)\frac{1}{|S(L \wedge t)|} \\ &= Z(L \wedge t).\end{aligned}$$

Therefore, $Z(L \wedge t)$ is a supermartingale. That is,

$$\mathbf{E}[Z(L \wedge t)] \leq \mathbf{E}[Z(L \wedge 0)] = 1, \quad \text{for all } t > 0.$$

The fact that the set $S(t)$ can contain at most the n nodes in the graph implies that

$$\begin{aligned}Z(L \wedge t) &= \frac{\exp\left(\frac{\Phi}{8}(L \wedge t)\right)}{|S(L \wedge t)|} \\ &\geq \frac{1}{n}\exp\left(\frac{\Phi}{8}(L \wedge t)\right). \quad (3.7)\end{aligned}$$

Taking expectations on both sides of (3.7) yields

$$\begin{aligned}\mathbf{E}\left[\exp\left(\frac{\Phi}{8}(L \wedge t)\right)\right] &\leq n\mathbf{E}[Z(L \wedge t)] \\ &\leq n.\end{aligned}$$

Because $\exp(\Phi(L \wedge t)/8) \uparrow \exp(\Phi L/8)$ as $t \to \infty$, the monotone convergence theorem implies that

$$\mathbf{E}\left[\exp\left(\frac{\Phi L}{8}\right)\right] \leq n.$$

Applying Markov's inequality, we obtain that, for $t_1 = 8(\ln 2 + 2\ln n + \ln \varepsilon^{-1})/\Phi$,

$$\begin{aligned}\Pr(L > t_1) &= \Pr\left(\exp\left(\frac{\Phi L}{8}\right) > \frac{2n^2}{\varepsilon}\right) \\ &\leq \frac{\varepsilon}{2n}.\end{aligned}$$

Thus, the time duration of the first phase is $O((\log n + \log \varepsilon^{-1})/\Phi)$ with probability at least $1 - \varepsilon/2n$.

3.2.1.2 Phase 2: $n/2 < |S(t)| \leq n$

Now we wish to study the time duration of second phase starting at $|S(t)| \geq n/2$ until $|S(t)| = n$. Unlike the first phase, we will study the evolution of the size of the set of nodes that do not have the information, i.e., $|S(t)^c|$. This quantity will decrease as the information spreads from nodes in $S(t)$ to nodes in $S(t)^c$. For simplicity, let us consider restarting the process from clock tick 0 after L (i.e., when at least half the nodes in the graph have the information for the first time), so that we have $|S(0)^c| \leq n/2$. As stated earlier, in this phase we will ignore the effect of 'push' and only consider the effect of the 'pull' aspect of the algorithm.

In time $t + 1$, a node $j \in S(t)^c$ will receive the information if it contacts a node $i \in S(t)$ and pulls the information from i. As such,

$$\mathbf{E}[|S(t)^c| - |S(t+1)^c| \mid S(t)^c] \geq \sum_{j \in S(t)^c, i \notin S(t)^c} P_{ji}.$$

Thus, we have

$$\begin{aligned}\mathbf{E}[|S(t+1)^c| \mid S(t)^c] &\leq |S(t)^c| - \sum_{j \in S(t)^c, i \notin S(t)^c} P_{ji} \\ &= |S(t)^c| \left(1 - \frac{\sum_{j \in S(t)^c, i \notin S(t)^c} P_{ji}}{|S(t)^c|}\right) \\ &\leq |S(t)^c| (1 - \Phi). \end{aligned} \tag{3.8}$$

We note that this inequality holds even when $|S(t)^c| = 0$, and as a result it is valid for all time t in the second phase. Repeated application of (3.8) yields

$$\begin{aligned}\mathbf{E}[|S(t)^c|] &= \mathbf{E}[\mathbf{E}[|S(t)^c| \mid S(t-1)^c]] \\ &\leq (1-\Phi)\,\mathbf{E}[|S(t-1)^c|] \\ &\leq (1-\Phi)^t\,\mathbf{E}[|S(0)^c|] \\ &\leq \exp(-\Phi t)\left(\frac{n}{2}\right).\end{aligned}$$

For $t_2 = (2\ln n + \ln \delta^{-1})/\Phi$, we have $E[|S(t_2)^c|] \leq \varepsilon/(2n)$. Application of Markov's inequality now implies

$$\begin{aligned}\Pr(|S(t_2)^c| > 0) &= \Pr(|S(t_2)^c| \geq 1) \\ &\leq E[|S(t_2)^c|] \\ &\leq \frac{\varepsilon}{2n}.\end{aligned}$$

Thus, we have obtained that the duration of the second phase is no longer than $O((\log n + \log \varepsilon^{-1})/\Phi)$ with probability at least $1 - \varepsilon/2n$. Therefore, by union bound it follows that the combined duration of the two phases is no longer than $O((\log n + \log \varepsilon^{-1})/\Phi)$ with probability at least $1 - \varepsilon/n$. Note that these bounds are independent of the starting node $v \in V$. Therefore, it implies the desired claim of Theorem 3.1

$$T_{\text{spr}}^{\text{one}}(\varepsilon) = O\left(\frac{\log n + \log \varepsilon^{-1}}{\Phi(P)}\right).$$

3.2.2 Applications

Here, we describe application of the Gossip algorithm for single-piece information dissemination for various relevant network graphs.

3.2.2.1 Complete graph

The complete graph represents a situation where all nodes can communicate with each other. That is, there are essentially no graphical communication constraints. For complete graph of n nodes, a natural symmetric probability matrix is $P = [1/n]$, i.e., all entries being equal to $1/n$. For such a matrix, as explained in *Preliminaries*, $\Phi(P) = O(1)$. Therefore, $T^{\text{one}}_{\text{spr}}(\varepsilon) = O(\log n + \log \varepsilon^{-1})$. That is, for $\varepsilon = \Omega(1/\text{poly}(n))$ we have $T^{\text{one}}_{\text{spr}}(\varepsilon) = O(\log n)$. Now since each node can spread information to at most one other node in a given time instance, the spreading time is lower bounded by $\Omega(\log n)$ for *any* graph. Thus the natural gossip algorithm spreads information essentially as fast as possible on complete graphs.

3.2.2.2 Ring graph

The ring graph of n nodes is formed by placing n nodes on a circle (or a ring) and connecting each node to two of its nearest neighbors. This is essentially the most communication constrained graph. Qualitatively, the complete graph represents one end of the spectrum of graphs while the ring graph presents the other. In a sense, complete graph has no 'geometry' while ring graph has the strongest possible 'geometry'.

For the ring graph of n nodes, a natural symmetric probability matrix P is as follows: for each i, $P_{ii} = 1/2$, $P_{ii^+} = P_{ii^-} = 1/4$ where i^+ and i^- represent neighbors of i on either side. As established in *Preliminaries*, $\Phi(P) = \Theta(1/n)$. Therefore, for $\varepsilon = \Omega(1/\text{poly}(n))$ we have $T_{\text{ave}}(\varepsilon) = O(n \log n)$. Now the ring graph has diameter n, and hence $\Omega(n)$ is a lower bound on *any* spreading algorithm. Thus, again the natural Gossip algorithm is essentially the fastest possible algorithm on the ring graph as well. We make a note of the fact that a simple *centralized* algorithm on ring graph will take only $O(n)$ time to spread a single piece

of information. However, the whole point of *gossip* algorithm is to be topology unaware. Therefore, even though the algorithm may be slower than optimal centralized algorithm by $O(\log n)$, such a performance is still very acceptable.

3.2.2.3 Expander graph

The expander graph, by definition, is a class of graphs indexed by the number of nodes n that have good 'expansion' property. Specifically, consider a d-regular expander with all nodes having degree d. The natural probability matrix P will be such that $\Phi(P) = O(1)$ as discussed in *Preliminaries*. Thus, again the $T_{\text{spr}}^{\text{one}}(\varepsilon) = O(\log n)$ for all $\varepsilon = \Omega(1/\text{poly}(n))$. That is, the Gossip algorithm performs essentially as fast as possible.

3.2.2.4 Geometric random graph

The Geometric random graph over n nodes is formed by placing nodes uniformly at random in a geographic area and then connecting nodes within distance $r = r(n)$, the connectivity radius. Such a graph, denoted as $G(n,r)$ is extensively used for modeling wireless networks. The detailed description is provided in *Preliminaries*. As established there, the natural P on $G(n,r)$ has $\Phi(P)$ scaling as r for an appropriate choice of r. Therefore, for $\varepsilon = \Omega(1/\text{poly}(n))$ we will have $T_{\text{spr}}^{\text{one}}(\varepsilon) = O(r^{-1}\log n)$. But the diameter of $G(n,r)$ scales as r^{-1}. Thus, again the gossip algorithm spreads information essentially as fast as possible.

3.3 Multi-Piece Dissemination

Here, we consider scenario where each node wishes to spread its own distinct message to all the other nodes as quickly as possible in the given graph $G = (V,E)$ with natural Gossip algorithm based on a graph conformant doubly stochastic symmetric matrix P. Let m_i denote the message of node i and $M = \{m_1, \ldots, m_n\}$ denote the set of all of these n messages.

The algorithm based on P runs in discrete time, denoted by $t \in \mathbf{N}$, as before. Let $S_i(t) \subset M$ denote the set of messages node i has at

the beginning of time t. Initially, $t = 0$ and $S_i(0) = \{m_i\}$ for all i. Under the Gossip algorithm, at each time t node $i \in V$ contacts one of its neighbors, say node j with probability P_{ij} and does not contact any node with probability P_{ii}. Upon contact, node i sends an arbitrary message from $S_i(t) \backslash S_j(t)$ to node j, if $S_i(t) \backslash S_j(t) \neq \emptyset$; and node j sends an arbitrary message from $S_j(t) \backslash S_i(t)$ to node i, if $S_j(t) \backslash S_i(t) \neq \emptyset$.

Some remarks are in order. As in the single-piece dissemination, we have both 'pull' and 'push' as part of the algorithm. Each node contacts at most one other node, but each node can be contacted by multiple nodes. For information exchange at time t, the nodes can utilize information that was present at nodes in the beginning of time t. Finally, computing $S_i(t) \backslash S_j(t)$ at node i requires information of neighbor j's message set $S_j(t)$. This can be done by various efficient data structures. However, this computation can be avoided by means of random linear codes [14, 54]. However, it comes at the cost of coding and decoding.

Now the quantity of interest. For any $\varepsilon > 0$, we wish to find the time by which all nodes have all the messages M with probability at least $1 - \varepsilon$. Formally, this ε all-dissemination time, denoted by $T^{\text{all}}_{\text{spr}}(\varepsilon)$ is defined as

$$T^{\text{all}}_{\text{spr}}(\varepsilon) = \inf \left\{ t \colon \Pr \left(\bigcup_{i=1}^{n} \{S_i(t) \neq M\} \right) \leq \varepsilon \right\}.$$

Recall the definition of the 'k-conductance' of P, denoted by $\Phi_k(P)$, from *Preliminaries*:

$$\Phi_k(P) = \min_{S \subset V : |S| \leq k} \frac{\sum_{i \in S; j \in S^c} P_{ij}}{|S|}.$$

Let us define $\hat{\Phi}(P)$ as 'Harmonic' like mean of $\Phi_k(P), 1 \leq k < n$, as

$$\hat{\Phi}(P) = \sum_{k=1}^{n-1} \frac{k}{\Phi_k(P)}.$$

Now we are ready to state the characterization of the all-spreading time of the above described natural Gossip algorithm based on P.

Theorem 3.2. Let P be an irreducible, doubly stochastic, and symmetric matrix on graph G. Then, for the natural Gossip algorithm described above

$$T_{\text{spr}}^{\text{all}}(\varepsilon) = O\left(\frac{\hat{\Phi}(P)\log\varepsilon^{-1}}{n}\right).$$

3.3.1 Proof of Theorem 3.2

Recall that the message set at node i in the beginning of time t, denoted by $S_i(t)$ is a subset of M. Since M can have 2^n distinct subsets, effectively each node can be in one of the 2^n distinct states, each corresponding to a distinct subset of M, at any time t. Thus, under the Gossip algorithm the overall network state $(S_1(t), \ldots, S_n(t))$ evolves over a state space of size $2^{n\times n}$. This is in sharp contrast with single-piece dissemination where we are only interested in the evolution of one monotonically increasing set. Naturally, this makes the analysis to follow much more involved and hence delicate compared to that of single-piece dissemination.

To study this evolution over a very high-dimensional space, we introduce the notion of 'type'. Specifically, two nodes, say i and j, are called of the same 'type' at time t if $S_i(t) = S_j(t)$. At time t, this notion of 'type' defines a partition of all n nodes into disjoint 'type classes', with nodes having the same set of messages, i.e., nodes of the same 'type' belong to the same 'type class'. We will call the size of a maximal type class at time t as 'maximum type-size' and denote it by $A(t)$. We will call $|S_i(t)|$, the number of messages at node i, as the 'dimension' of node i at time t. By natural association, we will abuse notation (hopefully without causing confusion) to call 'dimension' of a 'type class' as the dimension of any of the node belonging to that type class.

Now when all nodes have received all messages, there is only one type class of size n and dimension n. Therefore, we wish to find the following stopping time:

$$\inf\{t\colon |S_i(t)| = n,\ \forall\ i \in V\}.$$

Initially, at $t = 0$ we have $|S_i(t)| = 1$ for all $i \in V$. Thus, the information spreads to all the nodes when the overall dimension increases among all the nodes is $n(n-1)$. Motivated by this, we study the overall dimension increase as the 'performance metric' of the algorithm. To this end, define the dimension increase at time t, denoted by $D(t)$ as

$$D(t) = \sum_{i=1}^{n} (|S_i(t)| - 1) = \left(\sum_{i=1}^{n} |S_i(t)| \right) - n.$$

By definition, $D(0) = 0$ and when all nodes have all messages then $D(t) = n(n-1)$. Therefore, we will bound $T_{\text{spr}}^{\text{all}}(\varepsilon)$ by bounding the time for $D(t)$ to become $n(n-1)$. To this end, define

$$L_k = \inf\{t \colon A(t) \geq k\} \quad \text{and} \quad Y_k = D(t_k).$$

In words, L_k is the first time when any type class has at least k nodes, and Y_k is the total dimension increase up to time L_k. By definition, $L_1 = Y_1 = 0$. The following result provides a lower bound on Y_k.

Lemma 3.3. For any $1 \leq k \leq n$, $Y_k \geq k(k-1)$.

Proof. Consider the time L_k, which is the first time any type class contains k nodes. At this time, there are nodes $i_1, \ldots, i_k \in V$ in the same type class. Since these nodes are of the same type,

$$S_{i_1}(L_k) = \cdots = S_{i_k}(L_k).$$

By definition, $m_i \in S_i(t)$ for all $i \in V, t \in \mathbf{N}$. Hence, for all $1 \leq \ell \leq k$,

$$\{m_{i_1}, \ldots, m_{i_k}\} \subset S_{i_\ell}(L_k).$$

This implies that $|S_{i_\ell}(L_k)| \geq k$ for $1 \leq \ell \leq k$. Therefore, the total dimension increase is at least $(k-1)k$. That is,

$$Y_k = D(L_k) \geq k(k-1). \qquad \square$$

Finally, by definition L_n is the time when all nodes have dimension n and $Y_n = D(L_n) = n(n-1)$. That is, by the time L_n all nodes have received all messages. Therefore, a revised goal is to bound L_n. For this, we will bound $L_{k+1} - L_k$ for all $0 \leq k < n$.

Now, consider a time $t \in [L_k, L_{k+1})$. Note that all L_k are stopping times. Let there be b type classes, $C_1, \ldots, C_b$ at time t. For a pair of nodes i, j, let X_{ij} be an indicator random variable that is 1 if node i contacts node j at time t and the dimension of i increases as a result of the communication, and is 0 otherwise. Node i will contact j with probability P_{ij}. Similarly, j contacts i with probability P_{ji}, which is equal to P_{ij} by symmetry of P. If $S_i(t) = S_j(t)$, then there will be no increase in total dimension of either i and j upon communication between them. Now suppose $S_i(t) \neq S_j(t)$. Then either $S_i(t) \backslash S_j(t) \neq \emptyset$ or $S_j(t) \backslash S_i(t) \neq \emptyset$. Now if $S_i(t) \backslash S_j(t) \neq \emptyset$, then upon j contacting i, dimension of the j will increase by 1. Similarly, if $S_j(t) \backslash S_i(t) \neq \emptyset$ then dimension of i will increase by 1 upon i contacting j. In summary, if $S_i(t) \neq S_j(t)$ then

$$\mathbf{E}[X_{ij}] + \mathbf{E}[X_{ji}] \geq P_{ij}.$$

Now let $F(t) = D(t+1) - D(t)$ denote the total dimension increase of all nodes by the end of time t. Note that X_{ij} indicates an increase in dimension due to the 'pull' effect. Since each node contacts at most one other node for the 'pull' aspect of the protocol, by ignoring the 'push' effect we obtain

$$F(t) \geq \sum_{i \in V} \sum_{j \in V : j \neq i} X_{ij}. \tag{3.9}$$

From the above discussion, it follows that

$$\begin{aligned}
\mathbf{E}[F(t)] &\geq \sum_{i \in V} \sum_{j > i} (\mathbf{E}[X_{ij}] + \mathbf{E}[X_{ji}]) \\
&\geq \sum_{i \in V} \sum_{j > i : S_i(t) \neq S_j(t)} P_{ij} \\
&= \frac{1}{2} \sum_{i,j \in V : S_i(t) \neq S_j(t)} P_{ij} \\
&= \frac{1}{2} \sum_{\ell=1}^{b} |C_\ell| \frac{\sum_{i \in C_\ell, j \notin C_\ell} P_{ij}}{|C_\ell|} \\
&\geq \frac{n \Phi_k(P)}{2} \\
&\triangleq n p_k.
\end{aligned} \tag{3.10}$$

In the analysis above, we have utilized the fact that $t \in [L_k, L_{k+1})$ and hence the maximal type-class size, $A(t) \leq k$. This provides a lower bound on the expected total dimension increase during any round in the period $[L_k, L_{k+1})$. Note that this lower bound holds for any $t \in [L_k, L_{k+1})$ uniformly. Define

$$Z_k(t) = \sum_{s=L_k}^{t-1} (F(s) - np_k)\mathbf{1}_{\{s<L_{k+1}\}}, \tag{3.11}$$

where $Z_k(L_k) = 0$. For $t \geq L_k$, $Z_k(t)$ is a submartingale, i.e.,

$$\mathbf{E}[Z_k(t+1) \mid Z_k(t)] \geq Z_k(t). \tag{3.12}$$

The quantity L_{k+1} is a stopping time with respect to the history of the algorithm. It is easy to show that $\mathbf{E}[L_{k+1}] < \infty$ via a stochastic upper bound using a certain geometric random variable with positive probability. Moreover, the submartingale $Z_k(t)$ has bounded increments. A stopped submartingale is a submartingale, and hence we obtain

$$\mathbf{E}[Z_k(L_{k+1})] \geq \mathbf{E}[Z_k(L_k)] = 0. \tag{3.13}$$

Now, from the definitions of $L_k, L_{k+1}, Y_k, Y_{k+1}$, and (3.13), we obtain

$$\mathbf{E}[Y_{k+1} - Y_k] \geq np_k\mathbf{E}[L_{k+1} - L_k]. \tag{3.14}$$

Recall that L_n is the time when all nodes can decode all the messages. Summing the inequality in (3.14) for all $1 \leq k \leq n-1$ yields

$$\mathbf{E}[L_n] \leq \sum_{k=1}^{n-1} \frac{\mathbf{E}[Y_{k+1} - Y_k]}{np_k}. \tag{3.15}$$

From Lemma 3.3 and the fact that p_k is monotonically non-increasing in k, the quantity in the right-hand side of the inequality in (3.15) is maximized when $Y_k = k(k-1)$. Hence,

$$\begin{aligned} \mathbf{E}[L_n] &\leq \sum_{k=1}^{n-1} \frac{2k}{np_k} \\ &= \frac{2\hat{\Phi}(P)}{n}. \end{aligned} \tag{3.16}$$

By Markov's inequality, the inequality in (3.16) implies that

$$\Pr(L_n > 4\hat{\Phi}(P)/n) < 1/2.$$

Now, for the purpose of analysis, consider dividing time into epochs of length $4\hat{\Phi}(P)/n$, and executing the information dissemination algorithm from the initial state in each epoch, independently of the other epochs. The probability that, after $\log \varepsilon^{-1}$ epochs, some execution of the algorithm has run to completion in its epoch is greater than $1 - \varepsilon$. Using the running time of this virtual process as a stochastic upper bound on the running time of the actual algorithm, we conclude that

$$T^{\text{all}}_{\text{spr}}(\varepsilon) = O\left(\frac{\hat{\Phi}(P)\log \varepsilon^{-1}}{n}\right).$$

3.3.2 Applications

Here, we describe application of the Gossip algorithm for multi-piece information dissemination on various relevant network graphs.

3.3.2.1 Complete graph

As before, for the complete graph we consider the natural probability matrix $P = [1/n]$. For such a matrix, as explained in *Preliminaries*, $\hat{\Phi}(P) = O(n^2 \log n)$. Therefore, $T^{\text{all}}_{\text{spr}}(\varepsilon) = O(n \log n \log \varepsilon^{-1})$. That is, for $\varepsilon = \Omega(1/\text{poly}(n))$ we have $T^{\text{one}}_{\text{spr}}(\varepsilon) = O(n \log^2 n)$. This is only $\log^2 n$ 'slower' compared to the 'fastest' possible time of $\Theta(n)$ because each node can receive at most one message in a time-slot. Thus, the natural Gossip algorithm spreads information essentially as fast as possible on the complete graph.

3.3.2.2 Ring graph

For the ring graph of n nodes, as before we consider the following probability matrix P: for each i, $P_{ii} = 1/2$, $P_{ii^+} = P_{ii^-} = 1/4$ where i^+ and i^- represent neighbors of i on either side. As established in *Preliminaries*, $\hat{\Phi}(P) = \Theta(n^3)$. Therefore, for $\varepsilon = \Omega(1/\text{poly}(n))$ we have $T^{\text{all}}_{\text{spr}}(\varepsilon) = O(n^2 \log n)$. Now, the ring has diameter n, and by 'proper'

scheduling (everyone passes new messages to the left) it is indeed possible to spread all data to all nodes in time $O(n)$. Thus, the above algorithm seem to be 'slower' by a factor of n compared to the best possible. We believe that this is due to weakness of the analytic method presented here and not the algorithm.

3.3.2.3 Expander graph

The expander graph, like the complete graph, has $\hat{\Phi} = (n^2 \log n)$. Therefore, $T^{\text{all}}_{\text{spr}}(\varepsilon) = O(n \log n)$ for $\varepsilon = \Omega(1/\text{poly}(n))$. That is, the Gossip algorithm performs essentially as fast as possible.

3.4 Summary and Historical Notes

Here, we described Gossip based information dissemination algorithms for the single-piece and many-piece dissemination scenarios. These are the simplest possible natural algorithms. We presented a detailed analysis for the information spreading time in both single-piece and many-piece scenarios. We found that the spreading time depends on the different spectral properties of the graph: for the single-piece scenario it is the conductance and for the multi-piece scenario it is the 'harmonic-like' mean of k-conductances. The analysis presented here for single-piece information spreading is based on work by Mosk-Aoyama and Shah [55] and the analysis for multi-piece spreading is based on another work by Mosk-aoyama and Shah [54]. It should be noted that though here for multi-piece information spreading we use a 'BitTorrent-like' scheme, one can use a 'network coding' based approach as well (see [54] for details).

Historically, the analysis of multi-piece information spreading for BitTorrent like systems was studied using a 'mean-field' approximation by Qiu and Srikant [59] as well as Massoulie and Vojnovic [47]. The use of network coding for efficient gossiping was introduced by Deb et al. [14] for the complete graph.

4

Linear Computation

4.1 Setup

We are given a connected network of n nodes with connectivity graph $G = (V, E)$. Each node $i \in V$ has its own real value $x_i \in \mathbf{R}$. Let $\mathbf{x} = [x_i]$ be the vector of these n numbers. The quantity of interest is the average of these n numbers, denoted by $\mathbf{x}_{\text{ave}}$, where $\mathbf{x}_{\text{ave}} = \sum_{i=1}^{n} x_i/n = \mathbf{x}^T\mathbf{1}/n$. This question of computing averages in a distributed manner arises in many applications including distributed estimation, distributed control (popularly known as consensus), and distributed optimization.

A natural iterative and distributed algorithm for computing $\mathbf{x}_{\text{ave}}$ is based on linear dynamics. To this end, let time be discrete (or slotted) and indexed by $t \in \mathbf{N}$. Let $\mathbf{y}(t) = [y_i(t)]$ denote the vector of estimates at nodes of G at time t. Initially, $t = 0$ and $\mathbf{y}(0) = \mathbf{x}$. The estimates at time $t + 1$, $\mathbf{y}(t + 1)$ are obtained from $\mathbf{y}(t)$ by selecting an $n \times n$ matrix $W(t) = [W_{ij}(t)] \in \mathbf{R}_+^{n\times n}$ and performing a linear update

$$\mathbf{y}(t + 1) = W(t)\mathbf{y}(t).$$

That is, each node j sends the value $W_{ij}(t)y_j(t)$ to node i; upon receiving these values from the nodes (including from itself), node i sums them up to form its updated estimate $y_i(t + 1)$. To make

such an algorithm distributed with repsect to the network graph G, we require that $W(t)$ be graph conformant, i.e., if $(i,j) \notin E$ then $W_{ij}(t) = W_{ji}(t) = 0$. From results on products of matrices [20] it follows that if $W(t)$ belongs to a finite set of paracontracting matrices (i.e., $W(t)\mathbf{y} \neq \mathbf{y} \Leftrightarrow \|W(t)\mathbf{y}\| < \|\mathbf{y}\|$) then $\mathbf{y}(t) \to \mathbf{y}^*$ where $\mathbf{y}^* \in \cap_{i\in\mathcal{I}}\mathcal{H}(W_i)$. Here, $\mathcal{I} = \{i\colon W_i$ appear infinitely often in the sequence $W(t)\}$ and for $i \in \mathcal{I}$, $\mathcal{H}(W_i)$ denotes the eigenspace of W_i associated with eigenvalue 1. Therefore, to guarantee convergence to $\mathbf{x}_{\text{ave}}\mathbf{1}$, the algorithms considered here will choose $W(t)$ with $\mathbf{1}$ as an eigenvector with eigenvalue 1.

4.1.1 Example Algorithm

Here, we present an example of a linear dynamics based algorithm that belongs to the class of algorithms described above. Specifically, we describe a time-invariant algorithm, i.e., $W(t) = W$. The choice of W is done in a very simple manner based on the well known Metropolis–Hasting method. Specifically, for each node $i \in V$,

$$W_{ij} = \begin{cases} \frac{1}{2d}, & \text{if } j \neq i \quad \text{and} \quad (i,j) \in E, \\ 0, & \text{if } j \neq i \quad \text{and} \quad (i,j) \notin E, \\ 1 - \frac{d_i}{2d}, & \text{if } j = i. \end{cases}$$

Here d_i denotes the degree of node $i \in V$ and $d = \max_{i\in V} d_i$. It is well-known that $\mathbf{1}$ is an eigenvector of $W = [W_{ij}]$ thus defined with corresponding eigenvalue 1, which is the largest possible value. Further, since G is connected it defines an irreducible, aperiodic random walk on G. Therefore, it follows that

$$\mathbf{y}(t) = W^t\mathbf{y}(0) = W^t\mathbf{x} \to \mathbf{x}_{\text{ave}}\mathbf{1}, \quad \text{as } t \to \infty.$$

Thus the above class of linear dynamics based algorithms is indeed non-empty and it is quite easy to find an algorithm, i.e., a matrix W with the desired properties.

4.1.2 Quantity of Interest

Our interest is in finding a good estimate of $\mathbf{x}_{\text{ave}}$ as quickly as possible using the linear dynamics based algorithm. To this end, for any $\varepsilon > 0$

define the ε-computation time, denoted by $T_{\text{ave}}(\varepsilon)$, as

$$T_{\text{ave}}(\varepsilon) = \sup_{\mathbf{x}\in\mathbf{R}^n} \inf \left\{ t\colon \Pr\left(\frac{\|\mathbf{y}(t) - \mathbf{x}_{\text{ave}}\mathbf{1}\|}{\|\mathbf{x}\|} \geq \varepsilon \right) \leq \varepsilon \right\}. \tag{4.1}$$

The probability in the above definition is with respect to possible randomness in the algorithm (i.e., the randomness in selection of sequence $W(t), t \in \mathbf{N}$).

The algorithm performs a total of $|W(t)|_o \stackrel{\Delta}{=} |\{(i,j)\colon W_{ij}(t) \neq 0\}|$ computations in the time slot t. Therefore, the total computation performed to obtain an ε approximation (with probability at least $1-\varepsilon$) of $\mathbf{x}_{\text{ave}}$ at all nodes is

$$C_{\text{ave}}(\varepsilon) \stackrel{\Delta}{=} \sum_{t=0}^{T_{\text{ave}}(\varepsilon)-1} |W(t)|_o.$$

Our goal is to design an algorithm, that is to choose a $W(t), t \in \mathbf{N}$, so that ideally both $T_{\text{ave}}(\varepsilon)$ and $C_{\text{ave}}(\varepsilon)$ are minimized. In Gossip algorithms, it is reasonable to assume that all nodes are performing some computation in each time slot. Therefore, we will have $|W(t)|_o = \Omega(n)$ for all t. That is, $C_{\text{ave}}(\varepsilon) = \Omega(nT_{\text{ave}}(\varepsilon))$. In what follows, we first consider randomized algorithms that indeed incur a cost $C_{\text{ave}}(\varepsilon) = \Theta(nT_{\text{ave}}(\varepsilon))$. Therefore in such algorithms, the goal will be to minimize $T_{\text{ave}}(\varepsilon)$. We will characterize the $T_{\text{ave}}(\varepsilon)$ for these algorithms and relate them to the 'mixing time' of certain reversible random walk on the graph G. This leads to the conclusion that for graphs with good expansion property the randomized algorithms are essentially optimal in terms of $T_{\text{ave}}(\varepsilon)$ as well. Thus, randomized algorithms will be optimal both in terms of $T_{\text{ave}}(\varepsilon)$ and $C_{\text{ave}}(\varepsilon)$ for graphs with good expansion property.

In contrast, for graphs with 'geometry' (e.g., ring graph) the computation time $T_{\text{ave}}(\varepsilon)$ can be quite poor due to its relation to the mixing time of reversible random walk. To overcome this limitation, we consider determinstic linear dynamics in the second part. We present a time-invariant deterministic algorithm, i.e., $W(t) = W$ for all $t \in \mathbf{N}$, that has optimal $T_{\text{ave}}(\varepsilon)$ and near optimal $C_{\text{ave}}(\varepsilon)$ for graphs with 'geometry'. This algorithm is built upon certain non-reversible random walk on graph G. As a by product, this construction provides the

fastest mixing random walk (Markov chain) on a graph G with mixing time being of the order of the diameter of G.

4.2 Randomized Algorithms and Reversible Random Walks

This section describes an algorithm where $W(t)$ is chosen randomly in each time $t \in \mathbf{N}$. The $T_{\mathrm{ave}}(\varepsilon)$ of this randomized algorithm will relate to the mixing time of a reversible random walk. Therefore, the performance of this algorithm will be effectively characterized by reversible random walks.

4.2.1 Algorithm

Let $P = [P_{ij}]$ be any $n \times n$ doubly-stochastic matrix that is network graph G conformant. Here, we assume that P is symmetric, i.e., $P_{ij} = P_{ji}$ for all i, j. We assume that P is chosen such that it is aperiodic and irreducible. That is, a random walk with P as its transition matrix on G is ergodic and has the uniform distribution on G as its unique stationary distribution. We will call P the probability matrix corresponding to the algorithm, because we will choose $W(t)$ according to an i.i.d. distribution with $\mathbf{E}[W(t)]$ related to P. Now we describe the distribution from which $W(t)$ is chosen. By Birkhoff-Von Neumann's theorem(e.g. [30]), a non-negative doubly-stochastic matrix P can be decomposed into at most n^2 ($n^2 - 2n + 1$ to be precise) permutation matrices:

$$P = \sum_{m=1}^{n^2} \alpha_m \Pi_m, \quad \alpha_m \geq 0, \quad \sum_{m=1}^{n^2} \alpha_m = 1.$$

Define a random (matrix) variable Π with distribution

$$\Pr(\Pi = \Pi_m) = \alpha_m, \ 1 \leq m \leq n^2.$$

As part of the algorithm, we sample a permutation matrix as per the distribution of Π every time $t \in \mathbf{N}$ independently. Let $\Pi(t)$ be the permutation matrix sampled at time t. The $\Pi(t)$ can be thought of as a 'directed' matching of vertices: i is matched to j if $\Pi_{ij}(t) = 1$. Here i matched to j ($\Pi_{ij}(t) = 1$) is different from j matched to i ($\Pi_{ji}(t) = 1$).

As part of the algorithm, if i is matched to j at time t (i.e., $\Pi_{ij}(t) = 1$) then i updates its estimate by setting it to be the average of its own current estimate and that of j. That is, $y_i(t+1) = (y_i(t) + y_j(t))/2$. In matrix form, this corresponds to

$$\mathbf{y}(t+1) = \frac{1}{2}\left(I + \Pi(t)\right)\mathbf{y}(t).$$

That is, $W(t) = (I + \Pi(t))/2$. This completes the description of the algorithm. Note that exactly two operations per node and $2n$ total operations are performed in each time t in the whole network.

4.2.2 Performance

Here, we characterize $T_{\text{ave}}(\varepsilon)$ of the above stated randomized algorithm. Later, we shall relate it to the mixing time of a random walk on G with transition matrix given by $(I + P)/2$.

Theorem 4.1. For any $\varepsilon \in (0, 1/2)$, the ε-computation time of the randomized algorithm based on P described above is bounded by

$$\frac{0.5\log(3\varepsilon)^{-1}}{\log\lambda^{-1}} \leq T_{\text{ave}}(\varepsilon) \leq \frac{3\log\varepsilon^{-1}}{\log\lambda^{-1}},$$

where $\lambda = \frac{1}{2}(1 + \lambda_2(P))$.

The proof of Theorem 4.1 is stated in terms of upper bound and lower bounds separately, next.

4.2.2.1 Upper bound

Recall that, under the randomized algorithm, we have

$$\mathbf{y}(t+1) = W(t)\mathbf{y}(t), \tag{4.2}$$

where for a random permutation matrix $\Pi(t)$

$$W(t) = \frac{1}{2}\left(I + \Pi(t)\right). \tag{4.3}$$

Note that $W(t)$ is a doubly stochastic matrix for all t. That is, it preserves the ℓ_1 norm across t, i.e.

$$\mathbf{y}(t)^T\mathbf{1} = \mathbf{y}(0)^T\mathbf{1} = \sum_{i=1}^{n} x_i. \tag{4.4}$$

We wish to bound how quickly $\mathbf{y}(t) \to \mathbf{x}_{\text{ave}}\mathbf{1}$. That is, we wish to find out how quickly, the 'error' $\mathbf{z}(t) \stackrel{\Delta}{=} \mathbf{y}(t) - \mathbf{x}_{\text{ave}}\mathbf{1}$ goes to 0 in terms of its ℓ_2 norm. For this, we will compute the expectation of $\mathbf{z}(t)^T\mathbf{z}(t)$ and use Markov's inequality to obtain the desired conclusion.

To this end, let us first compute $\mathbf{E}[\mathbf{z}(t)^T\mathbf{z}(t)]$. Recall that, $\mathbf{z}(t) = \mathbf{y}(t) - \mathbf{x}_{\text{ave}}\mathbf{1}$ and $W(t)\mathbf{1} = \mathbf{1}$. Therefore,

$$W(t)\mathbf{z}(t) = W(t)\mathbf{y}(t) - \mathbf{x}_{\text{ave}}W(t)\mathbf{1} = \mathbf{y}(t+1) - \mathbf{x}_{\text{ave}}\mathbf{1} = \mathbf{z}(t+1).$$

Therefore, for any $t > 0$

$$\mathbf{z}(t)^T\mathbf{z}(t) = \mathbf{z}(t-1)^T W(t-1)^T W(t-1)\mathbf{z}(t-1). \tag{4.5}$$

Now, $W(t-1)$ is independent of $z(t-1)$. Therefore,

$$\begin{aligned}&\mathbf{E}[\mathbf{z}(t)^T\mathbf{z}(t)|\mathbf{z}(t-1)]\\ &\quad= \mathbf{z}(t-1)^T\mathbf{E}[W(t-1)^T W(t-1)]\mathbf{z}(t-1).\end{aligned} \tag{4.6}$$

Due to the time-invariance of the distributions of $W(\cdot)$, we have that $\mathbf{E}[W(t-1)^T W(t-1)] = \mathbf{E}[W(0)^T W(0)]$. Consider the following.

$$\begin{aligned}W(0)^T W(0) &= \frac{1}{4}\left(I + \Pi(0)^T\right)\left(I + \Pi(0)\right)\\ &= \frac{1}{4}\left(2I + \Pi(0) + \Pi(0)^T\right).\end{aligned} \tag{4.7}$$

Therefore,

$$\overline{W} \stackrel{\Delta}{=} \mathbf{E}[W(0)^T W(0)] = \frac{1}{2}\left(I + \overline{P}\right), \tag{4.8}$$

we recall that $\overline{P} = (P + P^T)/2$. Since P is symmetric, we have $\overline{P} = P$. The doubly stochasticity of P implies that of $\overline{W}$. Similarly, irreducibility and aperiodicity of P implies that of $\overline{W}$. Hence, $\mathbf{1}$ is the eigenvector with the unique largest eigenvalue 1 of $\overline{W}$. $\overline{W}$ is a symmetric matrix. By the variational characterization of eigenvalues, it follows that

$$\mathbf{z}(t-1)^T\overline{W}\mathbf{z}(t-1) \leq \lambda\mathbf{z}(t-1)^T\mathbf{z}(t-1), \tag{4.9}$$

where λ is the second largest eigenvalue of $\overline{W}$. It should be noted that all eigenvalues of $\overline{W}$ are non-negative due to its form (4.8). Also it follows from the property of eigenvalues that

$$\lambda = \frac{1}{2}\left(1 + \lambda_2(P)\right),$$

where $\lambda_2(P)$ is the second-largest eigenvalue of P.

Now recursive application of (4.5) and (4.9) yields

$$\mathbf{E}[\mathbf{z}(t)^T\mathbf{z}(t)] \leq \lambda^t \mathbf{z}(0)^T\mathbf{z}(0). \tag{4.10}$$

Now,

$$\begin{aligned}\mathbf{z}(0)^T\mathbf{z}(0) &= \mathbf{x}^T\mathbf{x} - n\mathbf{x}_{\text{ave}}^2 \\ &\leq \mathbf{x}^T\mathbf{x}.\end{aligned} \tag{4.11}$$

From (4.10), (4.11) and an application of Markov's inequality, we have

$$\begin{aligned}\Pr\left(\frac{\|\mathbf{y}(t) - \mathbf{x}_{\text{ave}}\mathbf{1}\|}{\|\mathbf{x}\|} \geq \varepsilon\right) &= \Pr\left(\frac{\mathbf{z}(t)^T\mathbf{z}(t)}{\mathbf{x}^T\mathbf{x}} \geq \varepsilon^2\right) \\ &\leq \varepsilon^{-2}\frac{\mathbf{E}[\mathbf{z}(t)^T\mathbf{z}(t)]}{\mathbf{x}^T\mathbf{x}} \\ &= \varepsilon^{-2}\lambda^t.\end{aligned} \tag{4.12}$$

From (4.12), it follows that

$$T_{\text{ave}}(\varepsilon) \leq \frac{3\log\varepsilon^{-1}}{\log\lambda^{-1}} \;=\; \frac{3\log\varepsilon}{\log\lambda}.$$

This completes the proof of upper bound claimed in Theorem 4.1.

4.2.2.2 Lower bound

Here, we present the proof of lower bound on $T_{\text{ave}}(\varepsilon)$ claimed in Theorem 4.1. Recall from the arguments above that $\mathbf{z}(t+1) = W(t)\mathbf{z}(t)$. Therefore,

$$\mathbf{E}[\mathbf{z}(t)] = \overline{W}^t\mathbf{z}(0). \tag{4.13}$$

By definition, $\overline{W} = (I + P)/2$ and hence it is a symmetric positive-semidefinite doubly stochastic matrix. It is irreducible and aperiodic as well. Hence it has non-negative real valued eigenvalues

$$1 = \lambda_1(\overline{W}) \;> \lambda_2(\overline{W}) \geq \cdots \geq \lambda_n(\overline{W}) \geq 0,$$

and corresponding orthonormal eigenvectors $\frac{1}{\sqrt{n}}\mathbf{1},\ \mathbf{v}_2,\ldots,\mathbf{v}_n$. Select

$$\mathbf{x} = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{n}}\mathbf{1} + \mathbf{v}_2\right) \Rightarrow \mathbf{z}(0) = \frac{1}{\sqrt{2}}\mathbf{v}_2.$$

For this choice of $\mathbf{x}$, $\|\mathbf{x}\| = 1$. Now from (4.13),

$$\mathbf{E}[\mathbf{z}(t)] = \frac{1}{\sqrt{2}}\lambda^t \mathbf{v}_2, \tag{4.14}$$

where as per earlier notation $\lambda = \frac{1}{2}(1 + \lambda_2(P)) = \lambda_2(\overline{W})$. For this particular choice of $\mathbf{x}$, we are going to lower bound the ε computation time by lower bounding $\mathbf{E}[\|\mathbf{z}(t)\|^2]$ and using Lemma 4.2 as stated below. By Jensen's inequality and (4.14),

$$\begin{aligned}
\mathbf{E}[\mathbf{z}(t)^T\mathbf{z}(t)] &\geq \sum_{i=1}^{n}\mathbf{E}[z_i(t)]^2 \\
&= \mathbf{E}[\mathbf{z}(t)]^T\mathbf{E}[\mathbf{z}(t)] \\
&= \frac{1}{2}\lambda^{2t}(\overline{W})\mathbf{v}_2^T\mathbf{v}_2 \\
&= \frac{1}{2}\lambda^{2t}. \qquad (4.15)
\end{aligned}$$

Lemma 4.2. Let X be a random variable such that $0 \leq X \leq B$. Then, for any $0 < \varepsilon < B$,

$$\Pr(X \geq \varepsilon) \geq \frac{\mathbf{E}[X] - \varepsilon}{B - \varepsilon}.$$

Proof.

$$\begin{aligned}
\mathbf{E}[X] &\leq \varepsilon\Pr(X < \varepsilon) + B\Pr(X \geq \varepsilon) \\
&= \Pr(X \geq \varepsilon)(B - \varepsilon) + \varepsilon.
\end{aligned}$$

Rearranging terms gives us the lemma. □

From (4.14), $\|\mathbf{z}(t)\|^2 \leq \|\mathbf{z}(0)\|^2 \leq 1/2$. Hence Lemma 4.2 and (4.15) imply that for $\varepsilon \in (0, 1/2)$

$$\Pr\left(\|\mathbf{z}(t)\| \geq \varepsilon\right) > \varepsilon, \tag{4.16}$$

for

$$t \leq \frac{0.5 \log(3\varepsilon)^{-1}}{\log \lambda^{-1}}.$$

This implies the desired lower bound in Theorem 4.1.

4.2.3 Relation to Mixing Time

We describe the relation between the ε-computation time, $T_{\text{ave}}(\varepsilon)$ of the randomized algorithm based on P and the ε-mixing time based on total variation distance, $\tau(\varepsilon, \overline{W})$ (defined in *Preliminaries*), of the random walk on G with transition matrix given by $\overline{W} = (I + P)/2$.

Theorem 4.3. Let $\varepsilon = n^{-\delta}$ for some $\delta > 0$. Then $T_{\text{ave}}(\varepsilon)$ of the randomized algorithm based on P and the mixing time $\tau(\varepsilon, \overline{W})$ of the random walk with transition matrix $\overline{W}$ are related as

$$T_{\text{ave}}(\varepsilon) = \Theta\left(\log n + \tau(\varepsilon, \overline{W})\right).$$

Proof. Recall that $\overline{W} = (I + P)/2$ has all eigenvalues non-negative and real valued. Therefore, the second largest (in norm) eigenvalue, λ of $\overline{W}$ is equal to $(1 + \lambda_2(P))/2$. Since P has non-negative entries, and hence the trace of P is non-negative, the sum of all eigenvalues of P is non-negative. Since the largest eigenvalue has value 1, it must be that $\lambda_2(P) \geq -1/n$. Therefore, $\lambda \geq (1 - 1/n)/2 \geq 1/4$ for all $n \geq 2$ and a network must have at least two nodes! Now, for any $x \in (0, 1)$ it follows that

$$\frac{x}{2} \leq \log(1 + x) \leq x \Rightarrow \log(1 + x) = \Theta(x).$$

Given this, it follows that

$$\log \lambda = \log(1 - (1 - \lambda)) = \Theta(1 - \lambda).$$

Therefore,

$$T_{\text{ave}}(\varepsilon) = \Theta\left(\frac{\log \varepsilon^{-1}}{1 - \lambda}\right).$$

For $\varepsilon = n^{-\delta}$, it follows that $T_{\text{ave}}(\varepsilon) = \Omega(\log n)$. Also, for such an ε,

$$\begin{aligned} T_{\text{ave}}(\varepsilon) &= \Theta\left(\frac{\log \varepsilon^{-1} + \log n}{1-\lambda}\right) \\ &= \Omega(\tau(\varepsilon, \overline{W})), \end{aligned} \tag{4.17}$$

where the last inequality follows from *Preliminaries*. From the two lower bounds on $T_{\text{ave}}(\varepsilon)$, we have for $\varepsilon = n^{-\delta}$

$$T_{\text{ave}}(\varepsilon) = \Omega(\log n + \tau(\varepsilon, \overline{W})). \tag{4.18}$$

Use of the lower bound on mixing time in terms of the spectral gap for reversible random walks as explained in *Preliminaries*, suggests that for the choice of $\varepsilon = n^{-\delta}$,

$$\tau(\varepsilon, \overline{W}) = \Omega\left(\frac{\log \varepsilon^{-1}}{1-\lambda}\right) = \Omega(\log n).$$

Therefore, from (4.17) it follows that

$$T_{\text{ave}}(\varepsilon) = O(\log n + \tau(\varepsilon, \overline{W})).$$

This completes the proof of Theorem 4.3. □

Figure 4.1 is a pictorial description of Theorem 4.3. The x-axis denotes the mixing time and y-axis denotes the computation time.

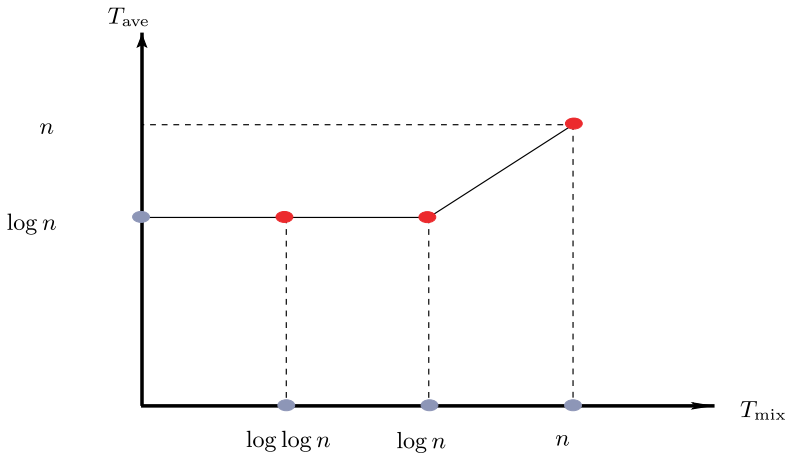

Fig. 4.1 Graphical interpretation of Theorem 4.3.

The plot assumes $\varepsilon = n^{-\delta}$ for some $\delta > 0$. The scale of the axis is in order notation. As shown in the figure, for P such that $\tau(\varepsilon, \overline{W}) = o(\log n)$, $T_{\text{ave}}(\varepsilon) = \Theta(\log n)$; for P such that $\tau(\varepsilon, \overline{W}) = \Omega(\log n)$, $T_{\text{ave}}(\varepsilon) = \Theta(\tau(\varepsilon, \overline{W}))$. Thus the mixing time of the random walk based on $\overline{W}$ essentially characterizes the computation time of the algorithm.

Remark 4.1. If one utilized a deterministic algorithm, i.e., using $W(t) = \overline{W}$ every time, then the convergence will be dictated by $\tau(\varepsilon, \overline{W})$. Thus, Theorem 4.3 suggests that by using randomization, the computation time suffers a minor additive penalty of $\log n$; but it potentially saves computations per iteration. Specifically the deterministic algorithm based on $\overline{W}$ will require $|\overline{W}|_o$ operations per iteration compared to $\Theta(n)$ operations per iteration under randomized algorithm.

4.2.4 Applications

Here we will utilize Theorems 4.1 and 4.3 to evaluate the performance of randomized algorithms on various graph models. We describe applications to four classes of graph models as stated below.

4.2.4.1 Complete graph

The complete graph represents situation when all nodes can communicate with each other. That is, essentially no graphical communication constrains. For a complete graph of n nodes, a natural symmetric probability matrix is $P = [1/n]$, i.e., all entries being equal to $1/n$. For such a matrix, as explained in *Preliminaries*, the $\tau(\varepsilon, P) = O(1)$ for all $\varepsilon > 0$. Therefore, by Theorem 4.3 the $T_{\text{ave}}(\varepsilon) = O(\log n)$ for all $\varepsilon = \Omega(1/\text{poly}(n))$. That is, the randomized algorithm performs essentially as fast as possible; both in terms of $T_{\text{ave}}(\varepsilon)$ and total cost, $C_{\text{ave}}(\varepsilon)$.

4.2.4.2 Ring graph

The ring graph of n nodes is formed by placing n nodes on a circle (or a ring) and connecting each node to two of its nearest neighbors. This

is essentially the most communication constrained graph. For a ring graph of n nodes, a natural symmetric probability matrix P, obtained by the Metropolis Hasting method, is as follows: for each i, $P_{ii} = 1/2$, $P_{ii^+} = P_{ii^-} = 1/4$ where i^+ and i^- represent neighbors of i on either side. As established in *Preliminaries*, the mixing time $\tau(\varepsilon, P) = \Omega(n^2)$ for any $\varepsilon \in (0, 1/2)$ and is $O(n^2 \log n)$ for $\varepsilon = \Omega(1/\text{poly}(n))$. Therefore, $T_{\text{ave}}(\varepsilon)$ essentially scales as n^2 and $C_{\text{ave}}(\varepsilon)$ scales as n^3. This clearly seems wasteful as a simple graph like a ring should have $T_{\text{ave}}(\varepsilon)$ more like n and not n^2. As we shall discuss in the next section, this waste is inherently due to P being symmetric.

4.2.4.3 Expander graph

As before, consider d-regular expander graphs. The natural probability matrix P has mixing time $\tau(\varepsilon, P) = O(\log n)$ for $\varepsilon = \Omega(1/\text{poly}(n))$ as discussed in *Preliminaries*. Thus, again $T_{\text{ave}}(\varepsilon) = O(\log n)$ for all $\varepsilon = \Omega(1/\text{poly}(n))$. That is, the randomized algorithm performs essentially as fast as possible; both in terms of $T_{\text{ave}}(\varepsilon)$ and the total cost, $C_{\text{ave}}(\varepsilon)$ (which is $O(n \log n)$).

4.2.4.4 Geometric random graph

The Geometric random graph over n nodes is formed by placing nodes uniformly at random in a geographic area and then connecting nodes within distance $r = r(n)$, the connectivity radius. Recall that the detailed description is provided in *Preliminaries*. As established there, the natural random walk has the fastest possible mixing time scaling as n/r^2. However, the diameter of $G(n, r)$ is of order $\sqrt{n}/r$. This suggests that the randomized algorithm is wasteful in such graphs.

4.3 Deterministic Algorithms and Non-Reversible Random Walks

The randomized algorithm described above has $T_{\text{ave}}(\varepsilon)$ essentially scaling like the mixing time. Though it performs minimal $\Theta(n)$ operations per unit time, its $T_{\text{ave}}(\varepsilon)$ can be rather poor for certain graphs. Specifically, for a complete graph or more generally graphs with good

expansion, the randomized algorithm performs almost optimally both in terms of $T_{\text{ave}}(\varepsilon)$ and hence $C_{\text{ave}}(\varepsilon)$. However, for graphs with 'geometry', like the ring or the geometric random graph, it performs rather poorly. And, most graphs arising in practice involving wireless networks or physical entities do have 'geometry', such as wireless sensor networks deployed in some geographic area [8, 18] or a nearest neighbor network of unmanned vehicles [63]. In this section, we will put forth a determinstic algorithm based on a novel construction of random walks on geometric graphs to overcome the inefficiency of the randomized algorithm. Before we proceed further, some understanding of inherent reasons for the inefficiency of the randomized algorithm will be useful.

Now the poor performance of the randomized algorithm on graphs with geometry is inherently related to the poor mixing property of the symmetric random walks on such graphs. To this end, recall from *Preliminaries* that for any random walk with transition matrix P, its mixing time is bounded as

$$\frac{1}{\Phi(P)} \leq \mathcal{H}(P) \leq O\left(\frac{\log n}{\Phi^2(P)}\right).$$

In general, in most graphs with geometry that are of interest, the mixing time of the reversible walk P scales like $1/\Phi^2(P)$. The conductance $\Phi(P)$ relates to diameter D of a graph G as $1/\Phi(P) \geq D$. Therefore, in such situations the mixing time of a reversible random walk is likely to scale like D^2, the square of the diameter. Indeed, Diaconis and Saloff-Coste [17] established that for a class of graphs with *geometry* (i.e., polynomial growth or finite doubling dimension) the mixing time of any reversible random walk scales like at least D^2 and it is achieved by the Metropolis–Hastings' approach. Thus, reversible random walks result in rather poor performance for graphs with geometry, i.e. its mixing time is far from the best hope, the diameter D. This suggests that in order to design an efficient linear dynamics based algorithm on graphs with 'geometry', we need to look for an algorithm that relates to non-reversible random walks on the graph. It should be noted that Theorem 4.1 implies that even for non-symmetric (i.e., non-reversible) P the bound on $T_{\text{ave}}(\varepsilon)$ is governed by the second largest eigenvalue of (or mixing time corresponding to) $\frac{1}{2}(I + P/2 + P^T/2)$, which is sym-

metric. Therefore, it is essential to consider a non-randomized, i.e., deterministic algorithm. Next, we describe a motivating example of a fast mixing random walk design based on non-reversibility.

4.3.1 Non-Reversible Random Walk: An Example

The example we describe here is based on work by Diaconis et al. [16]. They introduced a clever construction of a non-reversible random walk on the ring (and more generally ring-like) graph. To this end, consider Figure 4.2(a) which describes the symmetric random walk on a ring. The non-reversible random walk constructed in [16] runs on the *lifted* ring graph, which is denoted G_2 in Figure 4.2(b). Here, by lifting we mean making additional copies of the nodes of the original graph and adding edges between some of these copies while preserving the original graph topology. Figure 4.2(b) explains their construction for the ring graph. Note that each node has two copies and the lifted graph is essentially composed of two rings: an inner ring and an outer ring. The transition on the inner circle forms a clockwise circulation and the transition on the outer circle forms a counterclockwise circulation. And the probability of changing from the inner circle to the outer circle and vice versa are $1/n$ at each time. By defining transitions in this way, the stationary distribution is also preserved, i.e., the sum of stationary distributions of copies is equal to their original one. Somewhat

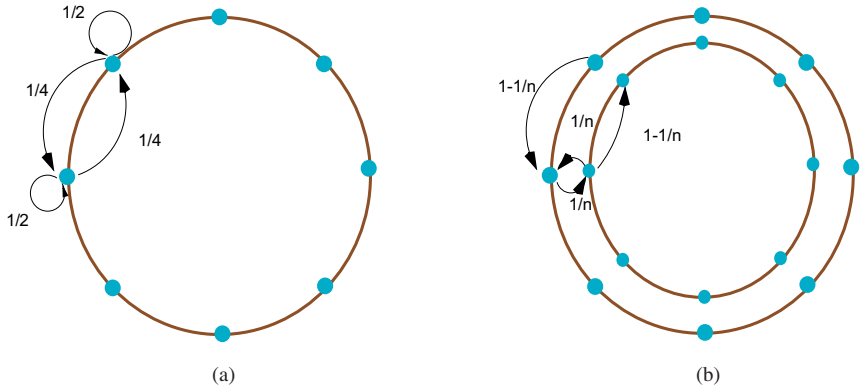

Fig. 4.2 (a) Symmetric P on the ring graph G_1. (b) Non-reversible P' on the lifted ring graph G_2.

surprisingly, they showed that this non-reversible random walk has linear mixing time $O^*(n)$.[1] Thus, effectively (i.e., upto $\log n$ factor) the mixing time scales like diameter n. It should be noted that because lifting preserves the graph topology and the stationary distribution, it is possible to simulate this lifted random walk on the original graph by expanding the *state* appropriately. Equivalently, if used for linear averaging it is possible to use lifted random walks by running iterations with extra states.

4.3.2 Non-Reversible Random Walks: General Graph

Given graph G and a symmetric doubly stochastic matrix P (say, obtained by Metropolis–Hasting method), here we describe a construction of a 'lifted graph' $\hat{G}$ and a non-reversible random walk with transition matrix $\hat{P}$ on $\hat{G}$ so that the mixing time of $\hat{P}$ is of the order of the diameter D of G. This construction will result in an increase in the size of $\hat{G}$, i.e., increase in $|\hat{P}|_o$ compared to $|P|_o$. Later we will improvise over this construction for graphs with 'geometry' to obtain $\hat{P}$ with $|\hat{P}|_o$ close to $|P|_o$. First, we formally define the notion of lifting, called 'Pseudo-lifting'.[2]

Definition 4.1(Pseudo-Lifting). A graph $\hat{G} = (\hat{V}, \hat{E})$ and a random walk $\hat{P}$ on it are called a pseudo-lifting of graph $G = (V, E)$ and random walk P if there exists a many-to-one function $f\colon \hat{V} \to V$, $T \subset \widehat{V}$ with $|T| = |V|$ such that the following holds: (a) for any $\hat{u}, \hat{v} \in \hat{V}$, $(\hat{u}, \hat{v}) \in \hat{E}$ only if $(f(\hat{u}), f(\hat{v})) \in E$, and (b) for any $u \in V$, $\hat{\pi}(f^{-1}(u) \cap T) = (1/2)\pi_u$,[3] where π and $\hat{\pi}$ are the stationary distributions of P and $\hat{P}$, respectively.

As we shall see later, due to property (a) in the definition it will be possible to simulate the pseudo-lifting $\widehat{P}$ in the original graph G in a distributed manner, i.e., run a linear dynamics based on $\hat{P}$ on G

[1] For a function $f\colon \mathbf{N} \to \mathbf{R}^+$, $O^*(f(n)) := O(f(n)\text{poly}(\log n))$.
[2] It is not called lifting because, the term lifting was used in [16] differently and Pseudo-lifting used here is in a sense a 'relaxation' of lifting.
[3] In fact, 1/2 can be replaced by δ for any constant $\delta \in (0, 1)$.

using message exchanges local to G. Furthermore, property (b) suggests that (by concentrating on set T), it is possible to simulate the uniform distribution exactly using the pseudo-lifting. Next, we present a construction of a pseudo-lifting with mixing time of order of D, the diameter of G.

4.3.2.1 Construction

For a given random walk P, we will construct the pseudo-lifted random walk $\widehat{P}$ of P. It may be assumed that P is given by the Metropolis–Hasting method. Without loss of generality (or rather to achieve generality), here we assume P to be the transition matrix of a random walk with an arbitrary stationary distribution $\pi = [\pi_i]$. Note that when specialized to $\pi = (1/n)\mathbf{1}$, we obtain the $\hat{P}$ of interest. In what follows, we will construct the pseudo-lifted graph $\widehat{G}$ by adding vertices and edges to G, and decide the values of the ergodic flows $\widehat{Q}$ on $\widehat{G}$, which defines its corresponding random walk $\widehat{P}$, since recall that ergodic flow along an edge $(\hat{u}, \hat{v})$ is $\hat{\pi}_u \hat{P}_{\hat{u}\hat{v}}$.

To this end, first select an arbitrary node v. Now, for each $w \in V$, there exist paths $\mathcal{P}_{wv}$ and $\mathcal{P}_{vw}$, from w to v and v to w, respectively. We will assume that all the paths are of length D: this can be achieved by repeating the same node or using self-loops. Now, we construct a pseudo-lifted graph $\hat{G}$ starting from G.

First, create a new node v' which is a copy of the chosen vertex v. Then, for every node w, add directed paths $\mathcal{P}'_{wv}$, a copy of $\mathcal{P}_{wv}$, from w to v'. Similarly, add $\mathcal{P}'_{vw}$ (a copy of $\mathcal{P}_{vw}$) from v' to w. Each addition creates $D-1$ new interior nodes. Thus, we have essentially created a virtual star topology using the paths of the old graph and added $O(nD)$ new nodes (note that, every new node is a copy of an old node).

Now, we define the ergodic flow $\widehat{Q}$ for this graph $\hat{G}$ as follows: for an edge (i,j),

$$\widehat{Q}_{ij} = \begin{cases} \frac{\delta}{2D}\pi_w, & \text{if } (i,j) \in E(\mathcal{P}'_{wv}) \quad \text{or} \quad E(\mathcal{P}'_{vw}) \\ (1-\delta)Q_{ij}, & \text{if } (i,j) \in E(G), \end{cases}$$

where $\delta \in (0,1)$ is a constant, that will be decided later. It is easy to check that $\sum_{ij} \widehat{Q}_{ij} = 1, \sum_j \widehat{Q}_{ij} = \sum_j \widehat{Q}_{ji}$. Hence it defines a a random

walk on $\hat{G}$. The stationary distribution of this pseudo-lifting is

$$\widehat{\pi}_i = \begin{cases} \frac{\delta}{2D}\pi_w, & \text{if } i \in (V(\mathcal{P}'_{wv}) \cup V(\mathcal{P}'_{vw}))\backslash\{w, v'\} \\ \left(1 - \delta + \frac{\varepsilon}{2D}\right)\pi_i, & \text{if } i \in V(G) \\ \frac{\delta}{2D}, & \text{if } i = v' \end{cases} \tag{4.19}$$

Given the above definition of $\hat{Q}$ and corresponding stationary distribution $\hat{\pi}$, it satisfies the pseudo-lifting definition if we choose ε such that $1/2 = \varepsilon(1 - (1/2D))$ and set $T = V(G)$, *i.e.*, T is the set of old nodes.

4.3.2.2 Mixing time

We claim the following bound on the mixing time of the pseudo-lifting we constructed.

Theorem 4.4. The mixing time of the random walk $\widehat{P}$ (equivalently, defined by $\widehat{Q}$) is $\tau(\varepsilon, \widehat{P}) = O(D \log \varepsilon^{-1})$ for any $\varepsilon \in (0, 1)$.

Proof. We will design a stopping rule where the distribution of the stopping node is $\widehat{\pi}$, and analyze its expected length. Then, use of the results from *Preliminaries* will lead to the bound on $\tau(\varepsilon, \widehat{P})$. Refer to *Preliminaries* for details on the use of stopping rule for bounding mixing time.

Now the description of stopping rule. Starting from any node, let the random walk continue until it reaches v'. Then roll a (four sides) die X with the following probability.

$$X = \begin{cases} 0, & \text{with probability } \frac{\delta}{2D} \\ 1, & \text{with probability } \frac{\delta(D-1)}{2D} \\ 2, & \text{with probability } 1 - \delta + \frac{\delta}{2D} \\ 3, & \text{with probability } \frac{\delta(D-1)}{2D} \end{cases}$$

Here $\delta \in (0, 1)$ is a constant, whose value will be chosen later. Depending upon the value of X, the walk will be stopped at a node to be decided as follows.

- $X = 0$: *Stop at* v'. The probability for stopping at v' is $\Pr[X = 0] = \delta/2D$, which is exactly $\widehat{\pi}_{v'}$.

- $X = 1$: *Walk a directed path P'_{vw}, and choose an interior node of P'_{vw} uniformly at random, and stop there.* For a given w, it is easy to check that the probability for walking P'_{vw} is π_w. There are $D - 1$ many interior nodes, hence, for an interior node i of P'_{vw}, the probability for stopping at i is

$$\Pr[X = 1] \times \pi_w \times \frac{1}{D-1} = \frac{\delta}{2D}\pi_w = \widehat{\pi}_i.$$

- $X = 2$: *Stop at the end node w of P'_{vw}.* The probability for stopping at w is

$$\Pr[X = 2] \times \Pr[\text{walk } P'_{vw}] = \left(1 - \delta + \frac{\delta}{2D}\right) \times \pi_w = \widehat{\pi}_w.$$

- $X = 3$: *Walk until getting a directed path P'_{wv}, and choose an interior node of P'_{wv} uniformly at random, and stop there.* Until getting a directed path P'_{wv}, the pseudo-lifted random walk defined by $\widehat{Q}$ is same as the original random walk. Since the distribution $w \in V(G)$ of the walk at the end of the previous step is exactly π, it follows that the distribution π over the nodes of $V(G)$ is preserved under this walk till walking on P'_{wv}. Calculations similar to those done in the case $X = 1$, we find that the probability of stopping at the interior node i of P'_{wv} is $\widehat{\pi}_i$.

Therefore, we have established the existence of a stopping rule that takes an arbitrary starting distribution to the stationary distribution $\hat{\pi}$. Now, this stopping rule has average length $O(D/\delta)$: since the probability of getting on a directed path P'_{wv} at w is $\frac{\delta}{2D}/(1 - \delta + (\delta/2D)) = \Theta(\delta/D)$, the expected number of walks until visiting v' and getting a directed path when $X = 3$ are $O(D/\delta) = O(D)$ in both cases. Now, $\tau(\varepsilon, \hat{P}) = O(\mathcal{H}(\hat{P}) \log \varepsilon^{-1})$ (see *Preliminaries*). Therefore, it follows that $\tau(\varepsilon, \hat{P}) = O(D \log \varepsilon^{-1}/\delta)$. Since δ is parameter of choice, we obtain that $\tau(\varepsilon, \hat{P}) = O(D \log \varepsilon^{-1})$ (e.g., choose $\delta = 0.1$). This completes the proof. □

4.3.3 Non-Reversible Random Walks: The Use of Geometry

The graph topologies arising in practice, such as that of a wireless sensor network deployed in some geographic area or a nearest neighbor network of unmanned vehicle [63], possess *geometry* and are far from being expanders. A good model for graphs with geometry is a class of graphs with finite doubling dimension which is defined as follows.

Definition 4.2 (Doubling Dimension). Consider a metric space $\mathcal{M} = (\mathcal{X}, \mathbf{d})$, where $\mathcal{X}$ is the set of point endowed with a metric $\mathbf{d}$. Given $x \in \mathcal{X}$, define a ball of radius $r \in \mathbf{R}_+$ around x as $\mathbf{B}(x,r) = \{y \in \mathcal{X} \colon \mathbf{d}(x,y) < r\}$. Define

$$\rho(x,r) = \inf \left\{ K \in \mathbf{N} \colon \exists\ y_1, \ldots, y_K \in \mathcal{X}, \mathbf{B}(x,r) \subset \bigcup_{i=1}^{K} \mathbf{B}(y_i, r/2) \right\}.$$

Then, the $\rho(\mathcal{M}) = \sup_{x \in \mathcal{X}, r \in \mathbf{R}_+} \rho(x,r)$ is called the *doubling constant* of $\mathcal{M}$ and $\log_2 \rho(\mathcal{M})$ is called the *doubling dimension* of $\mathcal{M}$. The doubling dimension of a graph $G = (V,E)$ is defined with respect to the metric induced on V by the shortest path metric.

The ring graph has $O(1)$ doubling dimension. The construction of the previous section leads to $\hat{P}$ with $\tau(\varepsilon, \hat{P})$ scaling as D and $|\hat{P}|_o$ scaling as nD for a general graph. That is, for a ring graph it will lead to (explained in detail in the next section) a deterministic linear computation algorithm with $T_{\text{ave}}(\varepsilon)$ scaling as n but $C_{\text{ave}}(\varepsilon)$ scaling as n^3. That is, the $C_{\text{ave}}(\varepsilon)$ of this algorithm will be the same as that of randomized algorithm. Therefore, we wish to improve the construction of the previous section for graphs with geometry, like a ring, by utilizing their structure in terms of the size of $\hat{P}$, i.e., $|\hat{P}|_o$.

Here, we will design a pseudo-lifting with efficient size for graphs with finite doubling dimension. To this end, recall that the basic idea for the construction of the pseudo-lifting is creating a virtual star topology using paths from every node to a fixed root, and the length of paths grows the size of the pseudo-lifting. For example, a caricature of this in the context of ring graph is shown in Figure 4.3(a). To reduce the overall length of paths, we make clusters of nodes such that nodes

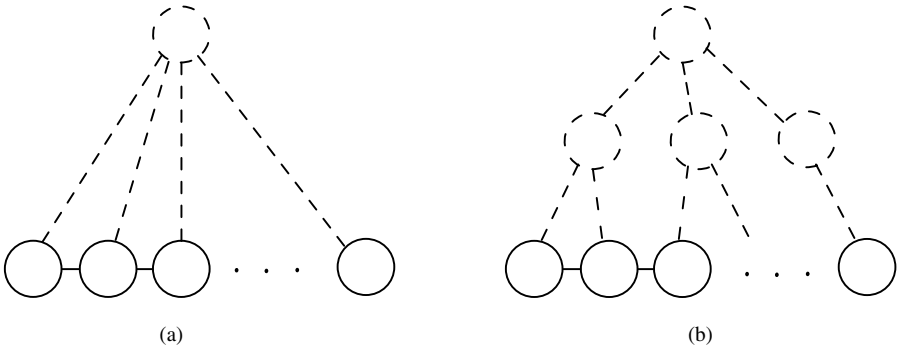

Fig. 4.3 For a given line graph with n nodes, (a) is the star topology which was used in the construction of the pseudo-lifted graph in Section 4.3.2.1, and (b) is the hierarchical star topology which will be used in this section for the new construction of the pseudo-lifting.

in each cluster are close to each other, and pick a sub-root node in each cluster. Then, we build a star topology in each cluster around its sub-root and connect every sub-root to the root. This creates a 'hierarchical' star topology (or 'tree' topology). A caricature example of such a construction is shown for a line graph in Figure 4.3(b). Since it needs short path lengths in each cluster, the overall length of paths decreases.

For a good clustering, we need to decide which nodes would become sub-roots. A natural candidate for them is the R-net $Y \subset V$ of graph G defined as follows.

Definition 4.3 (R-net). For a given graph $G = (V, E)$, $Y \subset V$ is an R-net if

(a) For every $v \in V$, there exists $u \in Y$ such that the shortest path distance between u and v is at most R.
(b) The distance between any two $y, z \in Y$ is more than R.

Such an R-net can be found in G greedily. As explained in the proof of Lemma 4.6, the small *doubling dimension* of G guarantees the existence of a good R-net for our purpose.

4.3.3.1 Construction

For a given random walk P, we will construct the pseudo-lifted random walk $\widehat{P}$ of P using the hierarchical star topology. As before, let π and $G = (V, E)$ be the stationary distribution and the underlying graph of P, respectively. Like the construction in Section 4.3.2.1, the pseudo-lifted graph $\widehat{G}$ is constructed by extending G; $\widehat{P}$ is defined by means of appropriate ergodic flows $\widehat{Q}$ on $\widehat{G}$.

Given an R-net Y, match each node w to the nearest $y \in Y$ (breaking ties arbitrarily). Let $C_y = \{w|\ w \text{ matched to } y\}$ for $y \in Y$. Clearly, $V = \cup_{y \in Y} C_y$. Finally, for each $y \in Y$ and for any $w \in C_y$ we have paths $\mathcal{P}_{wy}, \mathcal{P}_{yw}$ between w and y of length exactly R. Also, for each $y \in Y$, there exists $\mathcal{P}_{yv}, \mathcal{P}_{vy}$ between y and v of length exactly D (we allow the repetition of nodes to reach this length exactly).

Now, we construct the pseudo-lifted graph $\hat{G}$. As the construction in Section 4.3.2.1, select an arbitrary node $v \in V$ and create its copy v' again. Further, for each $y \in Y$ create two copies y'_1 and y'_2. Now, add directed paths $\mathcal{P}'_{wy}$, a copy of $\mathcal{P}_{wy}$, from w to y'_1 and add $\mathcal{P}'_{yv}$, a copy of $\mathcal{P}_{yv}$, from y'_1 to v'. Similarly, add $\mathcal{P}'_{vy}$ and $\mathcal{P}'_{yw}$ between v', y'_2 and y'_2, w. This construction adds $2D|Y| + 2Rn$ edges to G, giving $\hat{G}$. Now, the ergodic flow $\widehat{Q}$ on $\hat{G}$ is defined as follows: for any (i, j) of $\hat{G}$,

$$\widehat{Q}_{ij} = \begin{cases} \frac{\delta}{2(R+D)}\pi_w, & \text{if } (i,j) \in E(\mathcal{P}'_{wy}) \text{ or } E(\mathcal{P}'_{yw}), \\ \frac{\delta}{2(R+D)}\pi(C_y), & \text{if } (i,j) \in E(\mathcal{P}'_{yv}) \text{ or } E(\mathcal{P}'_{vy}), \\ (1-\delta)Q_{ij}, & \text{if } (i,j) \in E(G), \end{cases}$$

where $\pi(C_y) = \sum_{w \in C_y} \pi_w$ and $\delta \in (0, 1)$ is a constant to be decided later. It can be checked that $\sum_{ij} \widehat{Q}_{ij} = 1, \sum_j \widehat{Q}_{ij} = \sum_j \widehat{Q}_{ji}$. Hence it defines a random walk on $\hat{G}$. The stationary distribution of this pseudo-lifted chain is

$$\widehat{\pi}_i = \begin{cases} \frac{\delta}{2(R+D)}\pi_w, & \text{if } i \in (V(\mathcal{P}'_{wy}) \cup V(\mathcal{P}'_{yw}))\backslash\{w, y'_1, y'_2\}, \\ \frac{\delta}{2(R+D)}\pi(C_y), & \text{if } i \in (V(\mathcal{P}'_{yv}) \cup V(\mathcal{P}'_{vy}))\backslash\{v'\}, \\ \left(1 - \delta(1 - \frac{\delta}{2(R+D)})\right)\pi_i, & \text{if } i \in V(G), \\ \frac{\delta}{2(R+D)}, & \text{if } i = v'. \end{cases}$$

To establish this as the pseudo-lifting of the original random walk P, consider $T = V(G)$ and δ, where $1/2 = \delta(1 - 1/(2(R + D)))$. The $\widehat{G}$ has exactly $|E| + 2Rn + 2D|Y|$ edges.

4.3.3.2 Mixing time and size

We analyze performance of the above stated construction of $\hat{P}$ in terms of mixing time and size. First, we establish that the mixing time of $O(D)$ is preserved under this efficient construction.

Lemma 4.5. The mixing time of the random walk $\widehat{P}$ defined by $\widehat{Q}$ is $O(D)$.

Proof. Consider the following stopping rule. Walk until visiting v', and roll a (six sided) die X with the following probability.

$$X = \begin{cases} 0, & \text{with probability } \frac{\delta}{2(R+D)}, \\ 1, & \text{with probability } \frac{\delta D}{2(R+D)}, \\ 2, & \text{with probability } \frac{\delta(R-1)}{2(R+D)}, \\ 3, & \text{with probability } 1 - \delta(1 - \frac{\delta}{2(R+D)}), \\ 4, & \text{with probability } \frac{\delta(R-1)}{2(R+D)}, \\ 5, & \text{with probability } \frac{\delta D}{2(R+D)}. \end{cases}$$

Depending on the value of X,

- $X = 0$: Stop at v'.
- $X = 1$: Walk on a directed path $\mathcal{P}'_{vy}$, and choose its interior node uniformly at random, and stop there.
- $X = 2$: Walk until getting a directed path $\mathcal{P}'_{yw}$, and choose its interior node uniformly at random, and stop there.
- $X = 3$: Walk until getting to an old node in $V(G)$, and stop there.
- $X = 4$: Walk until getting a directed path $\mathcal{P}'_{wy}$, and choose its interior node uniformly at random, and stop there.
- $X = 5$: Walk until getting a directed path $\mathcal{P}'_{yv}$, and choose its interior node uniformly at random, and stop there.

It can be checked, using arguments similar to that in the proof of Theorem 4.4, that the distribution of the stopped node is precisely $\widehat{\pi}$. Also, we can show that the expected length of this stopping rule is $O((R+D)/\delta) = O(D/\delta) = O(D)$. This is primarily true because the probability of getting on a directed path $\mathcal{P}'_{wy}$ *at*w*is* $\Theta(\delta/(R+D))$. □

Now, we evaluate the size $|\hat{P}|_o$ of this construction for graphs with finite doubling dimension.

Lemma 4.6. Given a graph G with doubling dimension ρ and diameter D, the hierarchical construction gives a pseudo-lifted graph $\hat{G}$ with size $|\widehat{E}| = O(Dn^{1-\frac{1}{\rho+1}})$.

Proof. The property of the doubling dimension graph implies that there exists an R-net Y such that $|Y| \leq (2D/R)^\rho$ (cf. [2]). Consider $R = D2^{\frac{\rho}{\rho+1}} n^{-\frac{1}{\rho+1}}$. This is an appropriate choice because $R = D2^{\frac{\rho}{\rho+1}} n^{-\frac{1}{\rho+1}} > Dn^{-\frac{1}{\rho+1}} > n^{\frac{1}{\rho}-\frac{1}{\rho+1}} > 1$ (the second inequality is from $n \leq D^\rho$). Given this, the size of the pseudo-lifted graph $\hat{G}$ is

$$\begin{aligned}
|\widehat{E}| &= |E| + 2Rn + 2D|Y| \\
&\leq |E| + 2D\left(\frac{2^{\frac{\rho}{\rho+1}}}{n^{\frac{1}{\rho+1}}}\right) n + 2D\left(2\frac{n^{\frac{1}{\rho+1}}}{2^{\frac{\rho}{\rho+1}}}\right)^\rho \\
&= |E| + O(Dn^{1-\frac{1}{\rho+1}}).
\end{aligned}$$

Since $|E| = O(n)$ and $D = \Omega(n^{1/\rho})$, we have that $|\hat{E}| = O(Dn^{1-\frac{1}{\rho+1}})$. □

4.3.4 Back to Averaging: Deterministic Algorithm

We wish to compute $\mathbf{x}_{\text{ave}} = (\sum_i x_i)/n$ where $x_i \in \mathbf{R}$ is the value of node i on a given graph $G = (V, E)$. In what follows, we consider $x_i \geq 0$. This is without loss of generality since one may run an algorithm to compute the average of non-negative valued and negative valued numbers separately. Now let P be a doubly stochastic matrix over G of doubling dimension ρ such as the one obtained through the Metropolis–Hasting method. Let $\hat{P}$ be the efficient pseudo-lifting of P over the

lifted graph $\hat{G} = (\hat{V}, \hat{E})$ with stationary distribution $\hat{\pi}$ over $\hat{V}$. Recall that each node of V is part of $\hat{V}$. Let $V(G) \subset \hat{V}$ denote these nodes in what follows. Now consider the following deterministic linear algorithm over $\hat{V}$ based on $\hat{P}$. Let $\hat{\mathbf{y}}(0) = \hat{\mathbf{x}}$, with $\hat{\mathbf{x}} = [\hat{x}_i]$ where $\hat{x}_i = x_i$ for $i \in V(G)$ and $\hat{x}_i = 0$ otherwise. Now perform the following iterative linear computation over $\hat{G}$:

$$\hat{\mathbf{y}}(t+1) = \hat{P}\hat{\mathbf{y}}(t).$$

The above algorithm is described for graph $\hat{G}$, but our interest is in running the algorithm over G. Therefore, we describe an implementation of the above linear algorithm based on $\hat{P}$ on G.

As noted earlier, $V(G) \subset \hat{V}$ are nodes V of G. Now, under the above described linear algorithm, each node of $\hat{V}$ communicates with its neighbors connected via $\hat{E}$. But recall $\hat{u}$ and $\hat{v}$ are connected in $\hat{G}$ only if they are 'copies' of nodes u and v such that $(u, v) \in E$. Therefore, if each node $\hat{u} \in \hat{V}$ is 'simulated' by node u, where $\hat{u}$ is a copy of u, then all communications performed in each iteration based on $\hat{P}$ can be performed as local communications over the graph G. That is, G can indeed implement the linear dynamics based on $\hat{P}$ in a distributed manner. Finally, observe that each node in $V(G) \subset \hat{V}$ is hosted at its corresponding original node in V of G in the above implementation. Therefore, if each node $i \in V(G)$ learns the estimate $\mathbf{x}_{\text{ave}}$ then in the above implementation each node in V learns the estimate of $\mathbf{x}_{\text{ave}}$ as well. This completes the description of the algorithm. Now, we describe the convergence properties of the estimates of nodes in $V(G)$.

Lemma 4.7. Under the above algorithm, consider the estimates of nodes $V = V(G) \subset \hat{V}$. That is, for $i \in V(G)$ consider $\hat{y}_i(t)$. Then, $\hat{y}_i(t) \to \mathbf{x}_{\text{ave}}/2$ as $t \to \infty$. Further, for $t \geq \tau(\varepsilon/\sqrt{n}, \hat{P})$ and $i \in V(G)$,

$$|\hat{y}_i(t) - \mathbf{x}_{\text{ave}}/2| \leq \varepsilon \|\mathbf{x}\|_2.$$

Proof. Since $\hat{\pi}$ is the left eigenvector of $\hat{P}$ with the largest (unit) eigenvalue, i.e., $\hat{\pi}^T \hat{P} = \hat{\pi}^T$; we have

$$\lim_{t \to \infty} \mathbf{e}_i^T \hat{P}^t \to \hat{\pi}^T, \quad \forall\ i,$$

where $\mathbf{e}_i$ is the vector with its ith entry being 1 and the rest being 0. This implies that, for any i

$$\lim_{t\to\infty} \hat{P}^t_{ij} = \hat{\pi}_j, \quad \forall\ i. \tag{4.20}$$

Further, by the definition of the mixing time, it follows that for $t \geq \tau(\varepsilon, \hat{P})$,

$$\left|\hat{P}^t_{ji} - \hat{\pi}_i\right| \leq \varepsilon, \quad \forall\ j. \tag{4.21}$$

Now $\hat{\mathbf{y}}(t+1) = \hat{P}\hat{\mathbf{y}}(t)$. That is, $\hat{\mathbf{y}}(t) = \hat{P}^t\hat{\mathbf{y}}(0)$. Therefore, for any i

$$\hat{y}_i(t) = \sum_j \hat{P}^t_{ij}\hat{y}_j(0).$$

Also,

$$\begin{aligned}\sum_j \hat{y}_j(0)\hat{\pi}_j &= \sum_{j\in V(G)} x_j\hat{\pi}_j \\ &= \sum_{j\in V(G)} x_j\frac{1}{2n} \\ &= \mathbf{x}_{\text{ave}}/2.\end{aligned}$$

In the above equations, we have used the fact that $\hat{y}_j(0) = 0$ if $j \in \hat{V}\backslash V(G)$ and $\hat{y}_j(0) = x_j$ for $j \in V(G)$; $\hat{\pi}_j = 1/2n$ for $j \in V(G)$ by definition of pseudo-lifting. Putting the above together, for $i \in V(G)$

$$\begin{aligned}|\hat{y}_i(t) - \mathbf{x}_{\text{ave}}/2| &= \left|\sum_j \hat{P}^t_{ij}\hat{y}_j(0) - \sum_j \hat{\pi}_j\hat{y}_j(0)\right| \\ &\leq \sum_j \hat{y}_j(0)|\hat{P}^t_{ij} - \hat{\pi}_j|. \end{aligned}\tag{4.22}$$

From (4.20) to (4.22) it follows that for $t \geq \tau(\varepsilon, \hat{P})$ and for $i \in V(G)$

$$\begin{aligned}|\hat{y}_i(t) - \mathbf{x}_{\text{ave}}/2| &\leq \varepsilon\sum_j \hat{y}_j(0) \\ &= \varepsilon\|\mathbf{x}\|_1 = n\mathbf{x}_{\text{ave}} \\ &\leq \varepsilon\sqrt{n}\|\mathbf{x}\|_2. \end{aligned}\tag{4.23}$$

In above, we have used the fact that for any non-negative valued n-dimensional vector $\mathbf{x}$, $\sqrt{n}\|\mathbf{x}\|_2 \geq \|\mathbf{x}\|_1$. By choosing $\varepsilon/\sqrt{n}$ in place of ε we obtain the desired claim and we complete the proof. □

Table 4.1 Comparison of the determinstic linear algorithm based on pseudo-lifting with the randomized algorithm based on Metropolis–Hasting method.

	Rand. algo.	Det. algo.	Optimal
Running time: k-grid graph	$\Omega\left(\frac{1}{\Phi^2(P)}\right) : O^*(n^{\frac{2}{k}})$	$O(D) : O(n^{\frac{1}{k}})$	$D : n^{\frac{1}{k}}$
Size(dbl. dim.)ρ: k-grid graph	$\Theta(n) : \Theta(n)$	$O(n^{\frac{\rho}{\rho+1}} D) : O(n^{1+\frac{1}{k(k+1)}})$	$n : n$
Total # of operations: k-grid graph	$\Omega\left(\frac{n}{\Phi^2(P)}\right) : O^*(n^{1+\frac{2}{k}})$	$O(n^{\frac{\rho}{\rho+1}} D^2) : O^*(n^{1+\frac{k+2}{k(k+1)}})$	$nD : n^{1+\frac{1}{k}}$

Note: Optimal denotes lower bound for any distributed or message-passing algorithm.

The above suggests that, if we restrict our attention to nodes in $V(G)$, then their estimates converge to $\mathbf{x}_{\text{ave}}$. Specifically, let the n-dimensional vector $\tilde{\mathbf{y}}(t) = [\tilde{y}_i(t)] \in \mathbf{R}^n$ be defined as $\tilde{y}_i(t) = 2\hat{y}_i(t)$ for $i \in V(G)$. Then, from (4.23) it follows that for $t \geq \tau(\varepsilon/2n, \hat{P})$

$$\|\tilde{\mathbf{y}}(t) - \mathbf{x}_{\text{ave}}\mathbf{1}\|_2 \leq \varepsilon \|\mathbf{x}\|_2. \tag{4.24}$$

Thus, we have $T_{\text{ave}}(\varepsilon) \leq \tau(\varepsilon/2n, \hat{P})$. Therefore,

$$C_{\text{ave}}(\varepsilon) = O\big(|\hat{P}|_o T_{\text{ave}}(\varepsilon)\big) = O\big(|\hat{P}|_o \tau(\varepsilon/2n, \hat{P})\big).$$

This completes the description and analysis of the linear deterministic algorithm based on $\hat{P}$.

4.3.5 Application

Here, we describe application of the non-reversible random walk based determinstic algorithm described above for a class of graphs with geometry. As discussed above, for graphs with finite doubling dimension, our algorithm performs well. Specifically, for a graph with doubling dimension ρ and diameter D, the above results imply that the deterministic algorithm has $T_{\text{ave}}(\varepsilon)$ scaling as D and $C_{\text{ave}}(\varepsilon)$ scaling as D times $O^*(Dn^{1-1/(\rho+1)})$. Now any algorithm must take $O(D)$ as $T_{\text{ave}}(\varepsilon)$ and $C_{\text{ave}}(\varepsilon)$ as Dn. Thus, the effective 'loseness' in our algorithm compared to any other algorithm is no worse than $O(Dn^{-\frac{1}{1+\rho}})$.

Now the quintessential example of graphs with finite doubling dimension is the k-dimensional grid; one-dimensional grid being a line graph (or its symmetrized version is the ring graph). For k-dimensional

grid, the diameter D scales as $n^{1/k}$. Therefore, for our algorithm $T_{\text{ave}}(\varepsilon)$ scales as $\Theta(n^{1/k})$ and the total computation $C_{\text{ave}}(\varepsilon)$ scales as $O(n^{1+(1/k)+(1/(k(k+1)))})$. In contrast, for any algorithm $T_{\text{ave}}(\varepsilon)$ must scale at least as $n^{(1/k)}$ and $C_{\text{ave}}(\varepsilon)$ must scale at least as $n^{1+(1/k)}$. Thus, for a k-dimensional grid the possible loss in our algorithm is no more than $O(n^{1/(k(k+1))})$.

4.4 Summary

We considered linear dynamics based algorithms for the computing average of numbers in the network graph. First, we described a randomized algorithm in which a random permutation is chosen everytime so that the effective induced matrix corresponds to a certain symmetric doubly stochastic matrix on the graph. Equivalently, the algorithm corresponds to a certain randomized pair-wise 'averaging' based on a reversible random walk on the network graph. This algorithm performs minimal number of operations, i.e., on the order of the number of nodes n, per unit time in any network. The computation time scales as $\log n$ plus the mixing time of a related reversible random walk on the graph. Therefore, for graphs with 'good expansion' such as the complete graph or the constant degree expander, the computation time is essentially the minimal $O(\log n)$. However, this algorithm performs very poorly over graphs with 'geometry'. This is because the mixing time of a reversible random walk is inherently very large on graphs with geometry.

Motivated to improve performance, we considered algorithms based on non-reversible random walks. The performance of the randomized algorithm is inherently related to the reversible random walk, and hence we considered a deterministic algorithm. This comes at an additional cost of an increased number of operations per unit time. Therefore, we considered the question of designing non-reversible random walks on any graph with minimal mixing time and minimal 'size'. The notion of pseudo-lifting led to the design of non-reversible random walks with mixing times of the order of the diameter and small size. For graphs with 'geometry', i.e., graphs with finite doubling dimension, this was further improved in terms of size using the graph structure.

The averaging algorithms based on such non-reversible random walks take minimal number of iterations of the order of the diameter for any graph. For graphs with doubling dimension they have near optimal overall computation cost. In summary, the randomized algorithm is near optimal for graphs with good expansion, i.e., graphs without geometry; the deterministic algorithm based on a non-reversible random walk is near optimal for graphs with geometry.

We note that the deterministic construction based on pseudo-lifted Markov chain does incur a reasonable (polynomial in n) overhead. However, this cost can be thought of as amortized over time if the similar construction is used for many computations. Now, if topology is dynamic then it requires re-construction of pseudo-lifted chain often. Understanding the robustness of the pseudo-lifted chain and constructing pseudo-lifted chain with minimal overhead are natural questions of interest for future research.

4.5 Historical Notes

The recently emerging network paradigms such as sensor networks, peer-to-peer networks and surveilance networks of unmanned vehicles have led to the requirement of designing distributed, iterative and efficient algorithms for estimation, detection, optimization and control. Such algorithms provide scalability and robustness necessary for the operation of such highly distributed and dynamic networks. Motivated by applications of linear estimation in sensor networks [8, 37, 46, 69], information exchange in peer-to-peer networks [39, 55] and reaching consensus in a network of unmanned vehicles [6, 32, 63], we considered the problem of computing the average of numbers in a given network in a distributed manner. Specifically, we considered the class of algorithms for computing the average using distributed linear iterations. This approach was pioneered by Tsitsiklis et al. [69]. In the applications of interest, the rate of convergence of the algorithm strongly affect its performance. For example, the rate of convergence of the algorithm determines the agility of the distributed estimator to track the desired value [8] or the error in the distributed optimization algorithm [56].

This motivated us to consider the rate of convergence as the primary performance metric for such linear algorithms.

We considered two classes of algorithms: randomized and deterministic. The randomized algorithm reported here is based on work by Boyd et al. [8]. The deterministic algorithm reported here is based on work by Jung et al. [34].

5

Separable Function Computation

5.1 Setup

As usual, we are given an arbitrary connected network, represented by an undirected graph $G = (V, E)$, with $|V| = n$ nodes. Two nodes i and j can communicate with each other if (and only if) $(i, j) \in E$. We assume that, in a given time-slot each node can contact at most one other node. However, a node can be contacted by multiple nodes simultaneously.

Let 2^V denote the power set of the vertex set V (the set of all subsets of V). Let $\mathbf{x} = [x_i] \in \mathbf{R}^n$ denote a real valued n-dimensional vector. We call a function $f\colon \mathbf{R}^n \times 2^V \to \mathbf{R}$ *separable* if there exist functions $f_1, \ldots, f_n$ such that, for all $\mathbf{x} \in \mathbf{R}^n$ and $S \subseteq V$,

$$f(\mathbf{x}, S) = \sum_{i \in S} f_i(x_i). \tag{5.1}$$

Let $\mathcal{F}$ be the class of separable functions f for which $f_i(x) \geq 1$ for all $x \in \mathbf{R}$ and $i = 1, \ldots, n$. Here, the lower bound of 1 on value of $f_i(\cdot)$ is chosen for simplicity; in general it can be an arbitrary fixed positive constant.

Here, we wish to consider the question of designing a Gossip algorithm for the following separable function computation problem. Let x_i

denote the initial value of node $i \in V$ and $\mathbf{x} = [x_i]$ represent the vector of these initial values. Let $f \in \mathcal{F}$ be a given separable function. Then, all nodes in V wish to compute an estimate of $f(\mathbf{x}, V)$.

Clearly, the separable function computation stated above is equivalent to the following 'simpler' summation problem. Each node, say $i \in V$ has value $x_i \geq 1$. Let $\mathbf{x} = [x_i]$ denote the vector representation of these values. All nodes wish to compute an estimate of the summation $x_{\text{sum}} = \sum_i x_i$ using a Gossip algorithm as quickly as possible.

5.2 Algorithm

We will describe a Gossip algorithm for the summation problem to estimate x_{sum} at all nodes as quickly as possible. This algorithm will build on the information dissemination Gossip algorithm described earlier. We start with the description of a minimum computation algorithm that utilizes the information dissemination. Next, we state useful properties of the Exponential distribution. Finally, we will describe the summation algorithm that is obtained through a combination of computing the minimum and the properties of the Exponential distribution.

5.2.1 Minimum Computation

We start with the description of a Gossip algorithm for minimum computation based on the information dissemination algorithm. To this end, let each of the n nodes have distinct non-negative real values: let u_i be the value of node i. All nodes wish to compute $u_* = \min_i u_i$.

Let P be a doubly stochastic symmetric network graph G conformant irreducible matrix. Time is discrete and denoted by $t \in \mathbf{N}$. At each time t, each node i contacts its neighbor $j \in V$ with probability P_{ij}; it does not contact any node with probability P_{ii}. Upon making contact, both nodes i and j exchange all 'relevant' information. Now, we describe the information exchange aspect in the minimum computation algorithm.

Let $\hat{u}_i(t)$ be the estimate of u_* at node i at time t. Initially, $t = 0$ and $\hat{u}_i(0) = u_i$. As described above, at time t if a node $i \in V$ contacts another node j then i sends $\hat{u}_i(t)$ to j and j sends $\hat{u}_j(t)$ to i. Upon receiving various estimates, each node $i \in V$ sets its new estimate

$\hat{u}_i(t+1)$ as the minimum of its own value $\hat{u}_i(t)$ and all the received values.

We claim that, under the above described minimum computation algorithm, all nodes have the correct minimum by time $T_{\rm spr}^{\rm one}(\varepsilon)$ with probability at least $1-\varepsilon$. To see this, suppose that node $i \in V$ be such that $u_* = u_i$. Then, under the algorithm described the minimum 'spreads' in the same manner as in the setting of the single-piece dissemination. By definition, by time $T_{\rm spr}^{\rm one}(\varepsilon)$ all nodes will have the single-piece information with probability at least $1-\varepsilon$. Therefore, it follows that the minimum computation for all nodes under the above algorithm takes no more than $T_{\rm spr}^{\rm one}(\varepsilon)$ with probability at least $1-\varepsilon$. By Theorem 3.1, we have

$$T_{\rm spr}^{\rm one}(\varepsilon) = O\left(\frac{\log n + \log \varepsilon^{-1}}{\Phi(P)}\right).$$

Finally, consider the problem of computing r different minimums. Let each node $i \in V$ have r distinct values $u_i(1), \ldots, u_i(r)$. For $1 \leq \ell \leq r$, let

$$u_*(\ell) = \min_i u_i(\ell).$$

Then each node wishes to find out the r different minimums, $u_*(1), \ldots, u_*(r)$. It should be noted that by an application of the union bound and a 'round-robin' style parallel scheduling of r minimum computation algorithms as describe above, th computation time for r different minimums is bounded above by $rT_{\rm spr}^{\rm one}(\varepsilon)$ with probability at least $1 - r\varepsilon$. That is, r different minimums can be computed with probability at least $1-\varepsilon$ in time $\mathrm{T}_{\min}(r,\varepsilon)$, where

$$\mathrm{T}_{\min}(r,\varepsilon) = O\left(\frac{r(\log n + \log r + \log \varepsilon^{-1})}{\Phi(P)}\right). \tag{5.2}$$

5.2.2 A Useful Extremal Property

We describe an extremal property of Exponential random variables that will play a crucial role in designing Gossip algorithm for summation. Formally, we state it as follows.

Property 5.1. Let $W_1, \ldots, W_n$ be n independent random variables such that, for $i = 1, \ldots, n$, the distribution of W_i is Exponential with rate $\lambda_i > 0$, i.e., $\mathbf{E}[W_i] = 1/\lambda_i$. Let $\mathbf{W}$ be the minimum of $W_1, \ldots, W_n$. Then, $\mathbf{W}$ is an Exponential random variable of rate $\lambda = \sum_{i=1}^{n} \lambda_i$, i.e., $\mathbf{E}[\mathbf{W}] = 1/\lambda$.

Proof. For an Exponential random variable U with rate μ, for any $z \in \mathbf{R}_+$,

$$\Pr(U > z) = \exp(-\mu z).$$

Using this fact and the independence of the random variables W_i, we compute $\Pr(\mathbf{W} > z)$ for any $z \in \mathbf{R}_+$.

$$\begin{aligned}
\Pr(\mathbf{W} > z) &= \Pr\left(\cap_{i=1}^{n} \{W_i > z\}\right) \\
&= \prod_{i=1}^{n} \Pr(W_i > z) \\
&= \prod_{i=1}^{n} \exp(-\lambda_i z) \\
&= \exp\left(-z \sum_{i=1}^{n} \lambda_i\right).
\end{aligned}$$

This establishes the property stated above. □

5.2.3 Concentration of the Exponential Distribution

We state the concentration of the empirical mean of i.i.d. Exponential random variables. This will be useful in determining values of parameters in our summation algorithm.

Property 5.2. For any $k \geq 1$, let $Y_1, \ldots, Y_k$ be i.i.d. Exponential random variables with rate λ, i.e., $\mathbf{E}[Y_1] = 1/\lambda$. Let their empirical mean be denoted as R_k, i.e.,

$$R_k = \frac{1}{k} \sum_{i=1}^{k} Y_i.$$

Then, for any $\delta \in (0, 1/2)$,

$$\Pr\left(\left|R_k - \frac{1}{\lambda}\right| \geq \frac{\delta}{\lambda}\right) \leq 2\exp\left(-\frac{\delta^2 k}{3}\right). \tag{5.3}$$

Proof. By definition,

$$\mathbf{E}[R_k] = \frac{1}{k}\sum_{i=1}^{k} \lambda^{-1} = \lambda^{-1}.$$

Now the inequality in (5.3) follows directly from the well-known Large Deviation Principle implied by Cramér's Theorem (see [15], pp. 30, 35) for the empirical mean of i.i.d. random variables and the distributional property of Exponential random variables. □

5.2.4 Algorithm: Description and Performance

Now we are ready to describe the algorithm for summation computation. Recall that each node i has a positive real value $x_i \geq 1$. Each node i wishes to compute estimates of x_{sum}. Specifically, suppose all nodes wish to compute an estimate of x_{sum} within $[(1-\delta)x_{\text{sum}}, (1+\delta)x_{\text{sum}})]$ for some $\delta > 0$. We describe an algorithm that computes the estimation of x_{sum} at all nodes within this δ-accuracy with probability at least $1 - \varepsilon$ for a given $\varepsilon \in (0, 1/2)$.

To this end, let $r(\varepsilon, \delta) = 3\delta^{-2}\ln(4/\varepsilon)$. This selection is inspired by Property 5.2 for δ-accuracy with high enough probability. Inspired by Property 5.1, each node generates $r(\varepsilon, \delta)$ random numbers as follows: node i generates $r \stackrel{\Delta}{=} r(\varepsilon, \delta)$ random numbers $y_i(1), \ldots, y_i(r)$ by sampling Exponential distribution with parameter x_i. For $1 \leq \ell \leq r$,

$$y_*(\ell) = \min_i y_i(\ell).$$

All nodes compute these $r = r(\varepsilon, \delta)$ minimums using the minimum computation algorithm. Let $\hat{y}_i(\ell)$ be the estimate of the minimum $y_*(\ell)$ at node i (say after long enough time) for $1 \leq \ell \leq r(\varepsilon, \delta)$. Then node i generates an estimate of x_{sum} as $r(\sum_{\ell=1}^{r} \hat{y}_i(\ell))^{-1}$. Here, this choice of estimator is clearly inspired by Property 5.1 with the "confident accuracy" supplied by Property 5.2.

Now the algorithm for computing r minimums takes $O(r(\log n + \log r + \log \varepsilon^{-1})/\Phi(P))$ with probability at least $1-\varepsilon$, for $\varepsilon \in (0,1/2)$ as per (5.2). Therefore, for $r = r(\varepsilon,\delta)$ it takes time $O(\delta^{-2}\log(n\varepsilon^{-1}\delta^{-1})/\Phi(P))$ for all nodes to compute the minimum with probability at least $1-\varepsilon/2$. From Properties 5.1 and 5.2, after the minimum computation algorithm stops at all nodes, each node has an estimate of x_{sum} such that it is within $[(1-\delta)x_{\text{sum}}, (1+\delta)x_{\text{sum}})$ for all $\delta \in (0,1/2)$ with probability at least $1-\varepsilon/2$. Therefore, it follows by the union bound that under the above stated algorithm all nodes have an estimation of x_{sum} within 2δ-accuracy with probability at least $1-\varepsilon$ after time $O(\delta^{-2}\log(n\varepsilon^{-1}\delta^{-1})/\Phi(P))$. Therefore, we conclude the following result.

Theorem 5.1. Let P be an irreducible, doubly stochastic and symmetric matrix on graph G. Then, for any $\varepsilon,\delta \in (0,1/2)$ the summation computation algorithm described above based on P, leads to an estimation of x_{sum} within $[(1-\delta)x_{\text{sum}}, (1+\delta)x_{\text{sum}}]$ for all nodes in V with probability at least $1-\varepsilon$ within time $T_{\text{sum}}(\varepsilon,\delta)$ where

$$T_{\text{sum}}(\varepsilon,\delta) = O\left(\frac{\log n + \log \varepsilon^{-1} + \log \delta^{-1}}{\delta^2 \Phi(P)}\right).$$

5.2.5 Applications

Here, we will utilize Theorem 5.1 to evaluate the performance of the Gossip algorithm for summation or equivalently separable function computation on various graph models.

5.2.5.1 Complete graph

Recall that the complete graph represents situations when all nodes can communicate with each other. For a complete graph of n nodes with natural symmetric probability matrix $P = [1/n]$, as explained in *Preliminaries*, $\Phi(P) = O(1)$. Therefore, $T_{\text{sum}}(\varepsilon,\delta) = O(\delta^{-2}\log n)$ for $\varepsilon = \Omega(1/\text{poly}(n))$. That is, the algorithm performs computation essentially as fast as possible for any fixed δ.

5.2.5.2 Ring graph

Consider the ring graph of n nodes with a natural symmetric probability matrix P, obtained by the Metropolis–Hasting method, i.e., for each i, $P_{ii} = 1/2$, $P_{ii^+} = P_{ii^-} = 1/4$ where i^+ and i^- represent neighbors of i on either side. As established in *Preliminaries*, $\Phi(P) = O(1/n)$ for such P. Therefore, $T_{\text{sum}}(\varepsilon, \delta) = O(\delta^{-2} n \log n)$ for $\varepsilon = \Omega(1/\text{poly}(n))$. Since the diameter of a ring scales as n, this is again as fast as possible for any fixed δ.

5.2.5.3 Expander graph

For a d-regular expander with all nodes having degree d, the natural P has $\Phi(P) = O(1)$. Therefore, like the complete graph $T_{\text{sum}}(\varepsilon, \delta) = O(\delta^{-2} \log n)$ for $\varepsilon = \Omega(1/\text{poly}(n))$. That is, the algorithm performs computation essentially as fast as possible for any fixed δ.

5.2.5.4 Geometric random graph

For the geometric random graph over n nodes and connectivity radius $r = r(n)$ beyond the connectivity threshold, as established in *Preliminaries*, the natural probability matrix P on it has $\Phi(P)$ essentially scaling as r. Therefore, we will have $T_{\text{sum}}(\varepsilon, \delta) = O^*(\delta^{-2} r^{-1})$ for $\varepsilon = \Omega(1/\text{poly}(n))$. Again, since the diameter of $G(n, r)$ scales like $1/r$, the algorithm performs computation essentially as fast as possible for any fixed δ.

5.3 Summary

We described a Gossip algorithm for computing separable functions or equivalently computing the summation of distinct numbers in a network. The algorithm naturally builds over single-piece information dissemination by means of a natural minimum computation algorithm. This is made possible by a probabilistic transformation implied using an extremal property of the Exponential distribution: computing the summation can be performed by the computation of certain relevant minima. The computation time of the algorithm essentially scales as

that of the computation of the minimum or as single-piece information dissemination.

We remark on some features of the algorithm. The minimum computation algorithm can be implemented under a totally asychronous computational model. Therefore, the algorithm described here is quite robust with respect to the 'time-model'. The computation time of the algorithm is minimal for most of the reasonably regular graphs and scales like their diameter. The algorithm, as described, does not have a locally verifiable 'stopping condition'. However, a natural probabilistic stopping condition arises as an implication of Theorem 5.1 as follows. Since $\Phi(P)$ for most of the reasonable graphs is no smaller than $1/n$, if a node's estimate does not change for $O(n \log n)$ time slots (with large enough constant), then its estimate is correct with high enough probability as long as the number of nodes in the network is no larger than n.

Now, we remark on a weakness of the algorithm compared to the linear computation algorithm. Recall that the linear computation algorithm has time scaling proportional to $\log \varepsilon^{-1}$ for ε-accurate estimate with probability at least $1 - \varepsilon$. That is, it has scaling proportional to $\log \varepsilon^{-1}$ and $\log \delta^{-1}$ for δ-accurate estimate with probability $1 - \varepsilon$. In contrast, the algorithm describe here has scaling δ^{-2} and $\log \varepsilon^{-1}$ for δ-accurate estimate with probability $1 - \varepsilon$. Therefore, the algorithm decribed here performs rather poorly compared to the linear computaton for very small δ, equivalently very high accuracy.

Next, we remark on the 'implementation' of the algorithm. The algorithm described here requires exchange of 'real numbers'. However, any system implementation would require the exchange of bits and not real numbers. In a recent work by Ayaso et al. (see [3] for a detailed account of this work) a quantization of this algorithm is proposed. This leads to a slow-down of the computation time stated in Theorem 5.1 by a factor of $\log 1/\varepsilon$ for retaining accuracy within $1 \pm \varepsilon$. We also make a note of the following. The effective quantization of the linear dynamics based algorithms is far from clear and satisfactory analysis of the natural quantization of such algorithms seem non-trivial and is not known (to the best of author's knowledge). We refer an interested reader to a

recent work by Kashyap et al. [35] that proposes a mechanism to make the linear dynamics based quantized algorithm converge.

Finally, we remark on the 'optimality' of the algorithm. Under a natural probabilistic model, in the recent work Ayaso et al. [3] have established the optimality of the (quantized version of the) algorithm in terms of its depedence on the graph structure. Specifically, they established that any algorithm utilizing information spreading based on the probability matrix P cannot have a computation time scaling faster than $1/\Phi(P)$.

5.4 Historical Notes

The key concept behind the algorithm described here is the use of probabilistic extremal property for separable function. The concentration of the maximums of independent Geometric random variables was first used by Flajolet and Martin [23] to count distinct elements in a database. This idea was further improved by various authors over the past 25 years. For example, recent works by Considine et al. [12] and Enachescu et al. [21] build on [23] for approximate computation in sensor databases and sensor networks. The algorithm presented here is based on work by Mosk-Aoyama and Shah [55].

6

Network Scheduling

Wireless networks are becoming the architecture of choice for designing ad-hoc networks, metro-area networks and mesh-networks. The tasks of resource allocation and scheduling are essential for good network utilization. The multi-access capability of the wireless medium makes algorithm design for such networks intrinsically different and more challenging than its wireline counter-part. Further, wireless architectures require that algorithm be distributed and simple.

Here, we consider a Gossip based algorithm for scheduling nodes that wish to access a common wireless medium. The algorithm builds upon the information dissemination and the separable function computation algorithm along with an additional simple randomized sampling mechanism. Somewhat surprisingly, this algorithm utilizes the network capacity to the fullest despite its simplicity.

6.1 Setup

We consider an abstract model of a wireless network represented by a graph $G = (V, E)$ with $|V| = n$ wireless nodes and edges represented by E. As before, let $\mathcal{N}(i) = \{j \in V : (i, j) \in E\}$ denote the set of neighbors

of $i \in V$. We assume that network operates under the classical *indepedent set* interference model. That is, if a node v is transmitting then all of its neighbors in $\mathcal{N}(i)$ must not transmit at the same time. It should be noted that other popular combinatorial interference models are (computationally) equivalent to the independent set model. Therefore, even though the treatment here seems restricted to wireless networks, it naturally extends to other communication models.

The network operates in discrete time, i.e., time is slotted and $t \in \mathbf{N}$ denotes the time. Each node $i \in V$, capable of wireless transmission, can transmit at the unit rate to any of its neighbors. We ignore the power control for simplicity, but as an informed reader may notice, it can be easily included in the model. At each node, packets (of unit size) are arriving according to an external arrival process. Let $\bar{A}(t) = [\bar{A}_i(t)]$ denote the cumulative arrival process until time $t \in \mathbf{N}$, i.e., $\bar{A}_i(t)$ is the total number of packet that have arrived at node i in the time interval $[0,t]$; $\bar{A}(0) = \mathbf{0}$. Let $A_i(t) = \bar{A}_i(t) - \bar{A}_i(t-1)$ be the number of packets arriving at node i in time slot t. We assume that at most one packet can arrive at a node i in a time slot, i.e., $A_i(t) \in \{0,1\}$. We assume that arrivals happen in the middle of the time-slot. Finally, we assume that $A_i(\cdot)$ are Bernoulli i.i.d. random variable with $\Pr(A_i(t) = 1) = \lambda_i$. Let $\lambda = [\lambda_i] \in \mathbf{R}_+^n$ denote the arrival rate vector.

For simplicity we assume that the network is single-hop,[1] i.e., data arriving at a node i will depart the network after its transmission. Let $Q_i(t)$ denote the queue-size, or number of packets waiting, at node i at the end of the time-slot t with $Q(t) = [Q_i(t)]$. We assume the system starts empty, i.e., $Q(0) = \mathbf{0}$. Let $\bar{D}(t) = [\bar{D}_i(t)]$ denotes the cumulative departure process from $Q(t)$; $D(t) = \bar{D}(t) - \bar{D}(t-1) = [D_i(t)]$ denote the number of departures in time slot t. We assume that departures happen in the beginning of a time slot. Then,

$$\begin{aligned} Q(t) &= Q(0) + \bar{A}(t) - \bar{D}(t) \\ &= \bar{A}(t) - \bar{D}(t) \\ &= Q(t-1) + A(t) - D(t). \end{aligned} \tag{6.1}$$

[1] The model ignores multi-hop situation. However, as explained in [67], the algorithm presented can be easily extended for the case of multi-hop situation.

Now departures happen according to a scheduling algorithm which satisfies the interference constraint that no two neighboring nodes are transmitting data in the same time slot. To this end, let $\mathcal{I}$ denote the set of all independent sets of G. Formally define

$$\mathcal{I} \triangleq \{S \subset V \colon S = \emptyset \text{ or, if } i,j \in S \text{ then } (i,j) \notin E\}.$$

And each time slot the scheduling algorithm schedules nodes of an independent set $I \in \mathcal{I}$ to transmit packets. In what follows, we will denote independent set I as vector $I = [I_i]$ with $I_i \in \{0,1\}$ and $I_i = 1$ indicates that node i is in I. Let $I(t) \in \mathcal{I}$ be the schedule chosen by algorithm in the beginning of the time slot t. Then,

$$Q(t) = Q(t-1) + A(t) - I(t)\mathbf{1}_{\{Q(t-1)>0\}}, \tag{6.2}$$

where $\mathbf{1}_{\{Q(t-1)>0\}}$ is to make sure that if $I_i(t) = 1$ but $Q_i(t-1) = 0$ then there can be no departure.

6.1.1 Performance Metric and a Desirable Algorithm

We say that a network is *stable* for a given $\lambda \in \mathbf{R}_+^n$ under a particular scheduling algorithm if

$$\limsup_{t\to\infty} \mathbf{E}[Q_i(t)] < \infty, \quad \forall\ i \in V.$$

Such an algorithm will be said to provide 100% throughput or be throughput optimal.

From [67], it is clear that the set of all $\lambda \in \mathbf{R}_+^n$ for which there exists a scheduling policy so that the system stablity is given by $\Lambda = \mathsf{Co}(\mathcal{I})$, where $\mathsf{Co}(\mathcal{I})$ is the convex hull of $\mathcal{I}$ in $\mathbf{R}_+^n$. Hence, we call $\mathsf{Co}(\mathcal{I})$ the *throughput region* of the system.

In the work by Tassiulas and Ephremides [67], it was shown that a 'maximum weight independent set' scheduling algorithm is stable for all $\lambda \in \mathsf{Co}(\mathcal{I})$, where the schedule or independent set $I(t)$ chosen at time t is such that

$$I(t) \in \arg\max_{I\in\mathcal{I}} \langle I, Q(t-1)\rangle,$$

with the notation that $\langle A, B\rangle \triangleq \sum_{i\in V} A_i B_i$. A striking property of this 'maximum weight' algorithm is its ability to utilize network resources

to the fullest without actually 'learning' or 'knowing' the arrival rates or any other network parameters. In that sense, the maximum weight algorithm is 'universal'.

6.1.2 The Question

Now the maximum weight independent set algorithm is desirable in terms of performance. However, it requires solving the maximum weight independent set problem in the network graph every time. However, finding a maximum weight independent set in general is NP-hard [25] and even hard to approximate within $n^{1-o(1)}$ ($B/2^{O(\sqrt{\log B})}$ for a graph with node degree B) factor [68]. This brings us to the following challenging question: is it even possible to have any throughput optimal, polynomial (in n) time distributed algorithm?

6.2 Scheduling Algorithm

In what follows, we shall address the above mentioned question by designing a Gossip algorithm for scheduling in the network. That is, the algorithm by design is extremely simple and totally distributed. Somewhat surprisingly, it turns out to be throughput optimal. The algorithm we describe will be denoted by SCH. It is based on two distributed sub-routines, SAMP and COMP, which we shall describe before the description of SCH. It should be noted that COMP directly utilizes the summation algorithm from *Separable function computation*.

6.2.1 Random Sampler: SAMP

We describe a simple, distributed sampling algorithm SAMP to sample an independent set from $\mathcal{I}$. This algorithm may not sample independent sets uniformly from $\mathcal{I}$, but it samples each of them with strictly positive probability.

SAMP

- Each node $i \in V$ chooses $I_i = 0$ or 1 with probability 1/2 independently.

- If node i finds any $j \in \mathcal{N}(i)$ such that $I_j = 1$, it immediately sets $I_i = 0$.
- Now, output $I = [I_i]$ as a sampled independent set.

Now we state the main property of the sampling distribution induced by algorithm SAMP.

Property 6.1. Algorithm SAMP samples independent sets of graph G in distributed manner so that each independent set is sampled with probability at least 2^{-n}. Node i performs $O(|\mathcal{N}(i)|)$ operations, and the algorithm in total performs $O(|E| + n) \leq O(n^2)$ operations.

Proof. Since each node selects 0 or 1 independently with probability $1/2$, each one of the 2^n assignment of $\{0,1\}^n$ is equally likely (i.e., probability 2^{-n}). Each independent set corresponds to one such assignment in $\{0,1\}^n$. As part of the algorithm, if random node assignment is by itself an independent set, then it can be easily checked that the final output is that particular independent set only. Thus, each independent set has at least 2^{-n} probability of being selected. It can be easily checked that the output is always an independent set of G including $\emptyset$.

Now, the number of operations done by the algorithm are n random coin tosses. Each node $i \in V$ performs $O(|\mathcal{N}(i)|)$ comparisons. Equivalently, there are at most $O(|E|)$ operations for all nodes. These are all extremely simple distributed operations. Now for any graph, $|E| = O(n^2)$. Hence, the total number of operations is $O(n^2)$. □

6.2.2 Comparator: COMP

The purpose of the algorithm COMP is to compute the summation of node weights (approximately) for a given independent set. A useful property of this algorithm is that all nodes obtain the same estimate, and hence it allows for distributed decision in SCH.

Formally, given an independent set $I = [I_i]$ and node weights $W = [W_i]$, we wish to estimate $\langle I, W \rangle$. Equivalently, each node $i \in V$ has a number $x_i = I_i W_i$ and it wishes to estimate $x_{\text{sum}} = \sum_i x_i$. The weights

correspond to the queue-size. Therefore, $x_i \in \mathbf{N}$. By requiring that nodes with $x_i = 0$ do not participate (but help in computation), we will have that for all participating nodes $x_i \geq 1$. Thus, in effect we have nodes with $x_i \geq 1$ and we wish to compute $x_{\text{sum}} = \sum_i x_i$. This is precisely the question considered in *Separable function computation* (SFC).

The summation algorithm designed in (SFC) computes an estimate, $\hat{w}(I)$ of $\langle I, W\rangle$ in time $O(\delta^{-2}n/\Phi(P))$ over graph G using Gossip information exchange based on the probability matrix P (see Theorem 5.1, with choice of $\varepsilon = 3^{-n}$) with the following property:

$$\Pr\left(\hat{w}(I) \notin ((1-\delta)\langle I, W\rangle, (1+\delta)\langle I, W\rangle)\right) \leq 3^{-n}. \tag{6.3}$$

We summarize this as the following formal property.

Property 6.2. The algorithm COMP with parameter $\delta > 0$ produces an estimate $\hat{w}(I)$ of the weight of an independent set I, $w(I) \stackrel{\Delta}{=} \langle I, W\rangle$ so that $\hat{w}(I) \in ((1-\delta)w(I), (1+\delta)w(I))$ with probability at least $1 - 3^{-n}$ in time $O(n\delta^{-2}/\Phi(P))$ over G based on probability matrix P.

6.2.3 Scheduling Algorithm SCH

Now, we describe the scheduling algorithm SCH. As described below, it utilizes the distributed algorithms SAMP and COMP.

SCH

- The algorithm will use parameter $\delta > 0$.
- Let $I(t)$ be the independent set schedule chosen by the algorithm at time t.
- At time $t+1$, choose schedule $I(t+1)$ as follows:
  - — Generate a random independent set $R(t+1)$ using SAMP.
  - — Obtain esimates $\hat{w}(I(t)), \hat{w}(R(t+1))$ of weights $\langle I(t), Q(t)\rangle$ and $\langle R(t+1), Q(t)\rangle$, respectively within accuracy $(1 \pm \delta/8)$ using COMP with parameter $\delta/8$.

— If $\hat{w}(R(t+1)) > (1+\delta/8)/(1-\delta/8)\hat{w}(I(t))$, then set $I(t+1) = R(t+1)$. Else, set $I(t+1) = I(t)$.

- Repeat the above algorithm every time.

Some remarks are in order. The algorithm SCH takes $O(n\delta^{-2}/\Phi(P))$ time-steps or equivalently, a total of $O(|E| + n^2\delta^{-1}/\Phi(P))$ distributed operations to compute a new schedule. In general, for any reasonable graph, $\Phi(P) = \Omega(1/n)$ and $|E| \leq n^2$. Therefore, SCH computes a new schedule in $O(n^2\delta^{-2})$ time-steps or total of $O(n^3\delta^{-2})$ operations for any network graph G.

The algorithm SCH can be further 'slowed' down by running it once every T steps. For any finite T, including $T = O(n^3\delta^{-2})$, the resulting algorithm will remain stable. Thus, in effect it leads to a stable algorithm that performs $O(1)$ overall operations for hard constraints like the independent set. It should be noted that this does not come for *free*: there is an increase in the average queue-sizes upon 'slowing down' the algorithm.

6.3 Performance of Scheduling Algorithm

Here, we shall establish the stability or throughput optimality property of SCH. We start with necessary technical preliminaries and then present the formal theorem followed by its proof.

6.3.1 Technical Preliminaries

We present useful technical preliminaries here. Consider a discrete time Markov chain on a countable state space $S = \mathbf{N}^M$ for some finite integer M. Let $X(t)$ denote the random state of the Markov chain at time $t \in \mathbf{N}$. Let $X(0) = \mathbf{0}$ (can be any arbitrary 'good' state). Let $L\colon S \to [0,\infty)$ and $f\colon S \to [0,\infty)$ be any non-negative valued functions with $L(\mathbf{0}) = 0$. The following is a well-known result and can be found in the book by Meyn and Tweedie [50].

Proposition 6.1. Let a Markov chain be aperiod and irreducible. Let there exists a finite set $C \subset S$ such that Markov chain satisfies the

following condition: $\forall\ t \in \mathbf{N}$,

$$\mathbf{E}[L(X(t+1))|X(t)] \leq L(X(t)) - f(X(t)) + B\mathbf{1}_{\{X(t)\in C\}},$$

with $B > 0$ a constant and $\sup_{x\in C} L(x) < \infty$. Then,

(a) Markov chain is positive recurrent with a unique stationary distribution $\pi = [\pi(x)]_{x\in S}$ such that

$$\pi(f) = \sum_{x\in S} \pi(x) f(x) < \infty.$$

(b) Further,

$$\lim_{t\to\infty} \mathbf{E}[f(X(t)] \to \pi(f).$$

Such results are popularly known as Foster–Lyapunov criteria for establishing positive recurrence of Markov chains.

6.3.2 Stability: Theorem and Proof

Here, we state the formal result establishing the throughput optimality of SCH and its proof in detail.

Theorem 6.2. The algorithm SCH described above using parameter $\delta > 0$ is stable as long as $\lambda \in (1-\delta)\mathsf{Co}(\mathcal{I})$. Further,

$$\lim_{t\to\infty} \mathbf{E}[\langle Q(t), \mathbf{1}\rangle] = O(6^n).$$

Proof. At time t, define Lyapunov function

$$L(t) = \langle Q(t), Q(t)\rangle = \sum_i Q_i^2(t).$$

We will study the 'average drift' in $L(\cdot)$ at time slots $t_k = kT$ for large enough T (will be 2.2^n) so that for $\lambda \in (1-\delta)\mathsf{Co}(\mathcal{I})$

$$\mathbf{E}[L(t_{k+1})|Q(t_k)] \leq L(t_k) - \frac{0.4\delta}{n}\langle Q(t_k), \mathbf{1}\rangle + B, \tag{6.4}$$

for some large enough (exponentially dependent on n) B. This will immediately imply that

$$\mathbf{E}[L(t_{k+1})|Q(t_k)] \leq L(t_k) - \phi\langle Q(t_k), \mathbf{1}\rangle + B\mathbf{1}_{\{Q(t_k)\in C\}}, \quad (6.5)$$

for some finite set C, constant B and $\phi > 0$. By Proposition 6.1 and the fact that the number of arrivals in a time interval of length T is at most nT, we will obtain the desired conclusion that

$$\limsup_{t\to\infty} \mathbf{E}[\langle Q(t), \mathbf{1}\rangle] < \infty.$$

Next, we proceed towards proving (6.4). Given $Q(t_k) = Q(kT)$, we wish to study the average drift, $L(t_{k+1}) - L(t_k)$: let $I(t)$ be the independent set schedule chosen by SCH at time t. Define

$$\Delta_i(t+1) = A_i(t+1) - D_i(t+1).$$

From the queueing dynamics in (6.1),

$$\begin{aligned} L(t+1) - L(t) &= \langle Q(t+1), Q(t+1)\rangle - \langle Q(t), Q(t)\rangle \\ &= \sum_i (Q_i(t+1) - Q_i(t))(Q_i(t+1) + Q_i(t)) \\ &= \sum_i \Delta_i(t+1)(2Q_i(t) + \Delta_i(t+1)) \\ &= \sum_i \Delta_i^2(t+1) + 2Q_i(t)\Delta_i(t+1). \end{aligned} \quad (6.6)$$

We will use the following facts: for all t, (1) $Q_i(t)D_i(t+1) = Q_i(t)I_i(t+1)$, (2) $\Delta_i^2(t+1) \leq 1$. By telescopic summation of (6.6) for $t = t_k, \ldots, t_{k+1} - 1$, we obtain

$$L(t_{k+1}) - L(t_k) \leq nT + 2\sum_{t=t_k}^{t_{k+1}-1} \langle Q(t), \Delta(t+1)\rangle. \quad (6.7)$$

Since the arrival process is Bernoulli i.i.d. with arrival rate vector λ,

$$\begin{aligned} &\mathbf{E}[L(t_{k+1}) - L(t_k)\,|\,Q(t_k)] \\ &\quad \leq nT + 2\sum_{t=t_k}^{t_{k+1}-1} \mathbf{E}[\langle Q(t), \lambda - I(t+1)\rangle | Q(t_k)]. \end{aligned} \quad (6.8)$$

We know that $\lambda \in (1-\delta)\mathsf{Co}(\mathcal{I})$, i.e.,

$$\lambda \leq \sum_j \alpha_j I_j, \quad \alpha_j \geq 0, \quad I_j \in \mathcal{I}, \quad \sum_j \alpha_j = 1-\delta.$$

Define

$$I^*(t) = \arg\max_{I\in\mathcal{I}} \langle I, Q(t-1)\rangle, \quad W^*(t) = \langle I^*(t), Q(t-1)\rangle,$$
$$W(t) = \langle I(t), Q(t-1)\rangle, \quad \boldsymbol{\Delta}(t) = W^*(t) - W(t).$$

Now, since at most n arrival and n departures can happen in a time-slot, we have $|W^*(t+s) - W^*(t)| \leq 2ns$ for every t, s. Some rearrangements, the above discussions and definitions yield the following:

$$\begin{aligned}
&\mathbf{E}[L(t_{k+1}) - L(t_k)\,|\,Q(t_k)] \\
&\quad \leq nT + 2\sum_{t=t_k}^{t_{k+1}-1} \mathbf{E}[\boldsymbol{\Delta}(t+1) - \delta W^*(t+1)|Q(t_k)] \\
&\quad \leq nT + 4nT^2 - \delta T W^*(t_k) + 2\sum_{t=t_k}^{t_{k+1}-1} \mathbf{E}[\boldsymbol{\Delta}(t)|Q(t_k)]. \qquad (6.9)
\end{aligned}$$

Note that so far, the derivation has been independent of the algorithm. Now, we bound the term $\sum_t \mathbf{E}[\boldsymbol{\Delta}(t)|Q(t_k)]$ using the property of SCH. To this end, first some useful definitions and facts. Define

$$Z = \inf_{m\geq 1}\{R(t_k+m) = I^*(t_k+m)\},$$

$$Z_1 = \inf_{m\geq 0}\{\text{Guarantee of (6.3) due to COMP does not hold at } t_k + Z + m\}.$$

By Property 6.1, we know that

$$\mathbf{E}[\min\{Z,T\}\,|\,Q(t_k)] \leq \mathbf{E}[Z\,|\,Q(t_k)] = \mathbf{E}[Z] \leq 2^n.$$

Define $\hat{T} = T - \min\{T, Z_1\}$. By Property 6.2 of COMPand an application of the union bound, we have $Z_1 \leq T$ with probability at most $T3^{-n}$. Hence,

$$\mathbf{E}[\hat{T}\,|\,Q(t_k)] \leq \mathbf{E}[\hat{T}] \leq T^2 3^{-n}.$$

Now, we are ready to bound $\sum_t \mathbf{E}[\mathbf{\Delta}(t)\,|\,Q(t_k)]$: define

$$A = [t_k + Z, t_k + Z + Z_1] \cap [t_k + 1, t_{k+1}],$$

and $B = [t_k + 1, t_{k+1}] \backslash A$. On the starting time of A, SAMP picks the maximum weight independent set of that time, i.e., $I^*(\tau_k + Z)$. If $Z_1 > 0$ (i.e., $A \neq \emptyset$), then by the property of COMPand SCH, we will have a schedule $I(t_k + Z)$ such that

$$W(t_k + Z) \geq \left(\frac{1 - \delta/8}{1 + \delta/8}\right)^2 W^*(t_k + Z) \approx (1 - \delta/2) W^*(t_k + Z).$$

Now, for $t > t_k + Z$ and $t \in A$, by definition of Z_1 and property of SCH, we have that

$$W(t) \geq W(t - 1) - n.$$

Putting the above discussion together with some re-arrangement and some bounds discussed above give:

$$\sum_{t \in A} \mathbf{E}[\mathbf{\Delta}(t) | Q(t_k)] \leq 2nT^2 + \frac{\delta T}{2} W^*(t_k). \tag{6.10}$$

For $t \in B$, note that the length of B is upper bounded by $\min\{Z, T\} + \hat{T}$. Using the bounds as discussed above and the obvious bound $\mathbf{\Delta}(t) \leq W^*(t)$ we have

$$\sum_{t \in B} \mathbf{E}[\mathbf{\Delta}(t) | Q(t_k)] \leq 2nT^2 + (2^n + T^2 3^{-n}) W^*(t_k). \tag{6.11}$$

For $T = 2.2^n$ and n large enough, it is clear that

$$(2^n + T^2 3^{-n}) \leq 0.1\delta T. \tag{6.12}$$

Replacing (6.10)–(6.12) in (6.9), we have

$$\begin{aligned} \mathbf{E}[L(t_{k+1}) - L(t_k) | Q(t_k)] &\leq (nT + 8nT^2) - 0.4\delta W^*(t_k) \\ &= -\frac{0.4\delta}{n} \langle Q(t_k), \mathbf{1} \rangle + O(5^n), \end{aligned} \tag{6.13}$$

since $T = 2.2^n$ and $W^*(t_k) \geq \langle Q(t_k), \mathbf{1} \rangle / n$ for any graph G. Inequality (6.13) is the same as (6.4), hence we have proved the desired property to establish positive recurrence.

Next, we prove the claim for the average queue-size, $\mathbf{E}[\langle Q(t), \mathbf{1}\rangle]$ as $t \to \infty$. Taking the expectation w.r.t. $Q(t_k)$ in (6.13),

$$\mathbf{E}[L(t_{k+1}) - L(t_k)] \leq -\frac{0.4\delta}{n}\mathbf{E}[\langle Q(t_k), \mathbf{1}\rangle] + O(5^n). \tag{6.14}$$

Telescopically, sum (6.14) for $k = 0, \ldots, K-1$; using the fact that $L(\cdot) \geq 0$ along with some re-arrangement yields

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbf{E}[\langle Q(t_k), \mathbf{1}\rangle] \leq O\left(\frac{n5^n}{\delta}\right).$$

Using the fact that for $t \in (t_k, t_{k+1}]$, $\langle Q(t), \mathbf{1}\rangle \leq \langle Q(t_k), \mathbf{1}\rangle + nT$, taking $K \to \infty$, and using the above inequality gives us

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{s=0}^{t-1}\mathbf{E}[\langle Q(s), \mathbf{1}\rangle] = O(6^n). \tag{6.15}$$

Given (6.13), the implication of Proposition 6.1(b) and (6.15), and the relation between the cesaro limit and the limit of a sequence implies that

$$\lim_{t\to\infty}\mathbf{E}[\langle Q(t), \mathbf{1}\rangle] = O(6^n). \tag{6.16}$$

This completes the proof of Theorem 6.2. □

6.4 Relation to Other Models

The above described model ignores the multi-hop setup. However, we have done so to keep the exposition simple. The scheduling algorithm of interest with the independent set interference constraint remain the same as maximum weight independent set but with weights being somewhat different. To explain this, we give example of two such well-known scenarios as follows.

1. *Multi-hop queuing network.* This network was considered in [67]. For a given network G, let S be the set of data-flows with arrival rate λ_s for flow $s \in S$. Let f_s and d_s denote source and destination node respectively for flow $s \in S$. The routing is assumed to be pre-determined in the network. If s passes through $v \in V$ then let $h(v, s) \in V$ denote

its next hop unless $v = d_s$ in which case its data departs from G. Let $Q_{vs}(t)$ denote the queue-size of flow s at node v at time t. Define

$$W_{vs}(t) = \begin{cases} Q_{vs}(t) - Q_{h(v,s)s}(t), & \text{if } v \neq d_s, \\ 0, & \text{if } v = d_s. \end{cases}$$

Define $W_v(t) = \max_{s \in S} W_{vs}(t)$ and $W(t) = [W_v(t)]$. Then the throughput optimal (stable) algorithm of [67] chooses $I^*(t)$ as the schedule which is a maximum weight independent set with respect to $W(t-1)$, i.e.,

$$I^*(t) = \arg\max_{I \in \mathcal{I}} \langle I, W(t-1) \rangle.$$

2. *Joint resource allocation & scheduling.* In [40], it is very well-explained that the problem of congestion control and scheduling decomposes into two weakly coupled sub-problems: (i) congestion control, and (ii) scheduling. We describe the link-level scheduling problem. We urge an interested reader to go through [40] for details. The setup of the problem is the same as in the previous example with difference that routing is not pre-determined. The coupling of congestion control and scheduling happens via Lagrange multipliers $\mathbf{q}(t) = [q_e(t)]_{e \in E}$. With the interference model of this paper, the scheduling problem boils down to the selection of maximum weight independent set $I^*(t)$ with respect to weight $W(t-1) = [W_v(t-1)]$, where $W_v(t-1) = \max_{e: e=(u,v) \in E} q_e(t-1)$.

6.5 Summary

Here, we presented a Gossip algorithm for multi-access scheduling in a wireless network under the independent set interference constraint. The algorithm presented builds upon the separable function computation algorithm and utilized a distributed sampler. Somewhat surprisingly, this rather simple algorithm turns out to be throughput optimal. The algorithm computes a new schedule in time that is inversely proportional to $1/\Phi(P)$, where $\Phi(P)$ is the conductance of the Gossip probability matrix. Thus the algorithm's computation time depends on the underlying network graph structure through this dependence on the reciprocal of $\Phi(P)$.

Now some remarks are in order about the practicality of this algorithm. In author's opinion, the algorithm as stated is unlikely to be useful for practice due to amount of the overhead involved in terms of control information that is exchanged in the network for finding a new schedule every time. However, the algorithm presented here provides a *proof-of-concept* for existence of simple, distributed throughput optimal algorithm. Better implementable gossip style throughput optimal scheduling algorithm remain of interest for future research. We take note of an exciting recent progress towards this goal by Rajagopalan et al. [60].

6.6 Historical Notes

We present a brief summary of previous work on network scheduling algorithms. The result by Tassiulas and Ephremides [67] established that the 'max-weight scheduling' policy is throughput optimal for a large class of scheduling problems. This result has been very influential in the design of scheduling algorithms since then. Application to input-queued switches led to an excellent development of theory and practice of algorithms for scheduling under matching constraints: notably, the results of [13, 27, 48, 49, 64, 65, 68]. A recent interest in wireless networks has led to proposal of distributed scheduling algorithms under matching constraints [10, 11, 29, 41]. Most of these algorithms, based on finding the maximal matching, guarantee only a constant fraction of throughput. Recently, Modiano et al. [51] exhibited a throughput optimal, extremely simple distributed scheduling algorithm with matching constraints. This algorithm, as discussed in [40], easily extends to provide a throughput optimal algorithm for the resource allocation and scheduling problem under matching constraints. The algorithm presented here is a natural generalization of [51] and was discussed in work by Jung and Shah [33]. The application of this algorithm to the joint scheduling and congestion control problem is described in a recent work by Eryilmaz et al. [22].

7

Network Convex Optimization

The efficient utilization of network resources requires a good resource allocation algorithm. Usually, such an algorithm is required to solve a network-wide global optimization problem. Many of the important network resource allocation problems can be viewed as convex optimization (minimization) problems with linear constraints. In the context of next generation networks, such as P2P or sensor networks, these need to be implemented through Gossip mechanism. This motivates the study of Gossip algorithms for global optimization problems without relying on any form of network infrastructure.

In the classical literature, convex minimization problems with linear constraints are known to be solvable through iterative algorithms by means of the dual decomposition or primal–dual algorithms. However, in most of the network resource allocation problems, these are not 'truly distributed' or 'local' or in our terminology, Gossip algorithms. This is because algorithms known in the classical literature are 'distributed' with respect to the "constraint graph" of the problem and not the 'network graph'.

To explain this subtle but important difference, as before consider a connected network of n nodes with *network graph* $G = (V, E)$ with $V = \{1, \ldots, n\}$ and E representing edges along which communication is feasible. Now suppose each node $i \in V$ has a non-negative variable x_i associated with it. The goal is to assign values for the x_i's to optimize a global network objective function under network resource constraints. We assume that the global objective function $f \colon \mathbf{R}_+^n \to \mathbf{R}$ is separable in the sense that $f(\mathbf{x}) = \sum_{i=1}^n f_i(x_i)$ for any variable assignment $\mathbf{x} \in \mathbf{R}_+^n$. The feasible region is described by a set of nonnegative linear constraints. Let us consider a specific example.

Example 7.1 (Network resource allocation). Given a connected edge capacitated network $G = (V, E)$, with each edge in E having non-negative capacity, each user $i \in V$ wishes to transfer data to a specific destination along a particular path in the network, and has a utility function that depends on the rate x_i that the user is allocated. The goal is to maximize the global network utility, which is the sum of the utilities of individual users. The rate allocation $\mathbf{x} = (x_i)$ must satisfy capacity constraints, which are linear.

Now the *constraint graph* of the above optimization problem, denoted by $G_C = (V_C, E_C)$ where V_C corresponds to variables and edges in E_C correspond to pairs of variables that participate in a common constraint. For example, in the network resource allocation problem above, the constraint graph G_C contains an edge between two users if and only if their paths intersect. Operationally, Gossip algorithm for rate allocation must be local with respect to the network graph G. Note that typically $G_C \not\subseteq G$ (i.e., $E_C \not\subseteq E$), and hence a fully distributed algorithm with respect to G_C is *not* fully distributed with respect to G; hence can satisfy 'Gossip' properties.

In summary, most of the classical algorithms from optimization theory (see Historical notes for details and references therein) provide distributed algorithms with respect to the constraint graph G_C, and not with respect to the underlying network graph G. Here, we shall describe Gossip algorithm that is distributed with respect to G and

naturally builds on the summation algorithm described in *Separable function computation*.

7.1 Setup

Here, we consider the problem of minimizing a convex separable function over linear inequalities. As noted earlier, let $G = (V, E)$ be the connected network graph with $V = \{1, \ldots, n\}$. Each node $i \in V$ has non-negative decision variable $x_i \in \mathbf{R}_+$.

We consider convex minimization problems of the following general form.

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \triangleq \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \mathbf{R}_+^n. \end{aligned} \tag{P}$$

Here, we assume that the objective function is separable, i.e., $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$, and each $f_i \colon \mathbf{R}_+ \to \mathbf{R}$ is twice differentiable and strictly convex, with $\lim_{x_i \downarrow 0} f_i'(x_i) < \infty$ and $\lim_{x_i \uparrow \infty} f_i'(x_i) = \infty$. We will call this optimization problem as the primal problem (P).

The constraints are linear equality constraints of the form $A\mathbf{x} = \mathbf{b}$ with matrix $A \in \mathbf{R}_+^{m \times n}$ and a vector $\mathbf{b} \in \mathbf{R}_{++}^m$, and non-negativity constraints $x_i \geq 0$ on the variables. We assume that $m \leq n$, and that the matrix A has linearly independent rows. For $i = 1, \ldots, n$, let $a_i = [A_{1i} \cdots A_{mi}]^T$ denote the ith column of the matrix A. In this distributed setting, we assume that node i is given the vectors b and a_i, but not the other columns of the matrix A.

The goal is to design a Gossip algorithm that produces an ε-approximately feasible solution with objective function value close to that of an optimal feasible solution for a given error parameter $\varepsilon > 0$. We would like the running time of the algorithm to be polynomial in $1/\varepsilon$, the number of constraints m, the inverse of the conductance, $\Phi(P)$ of the Gossip based information exchange probability matrix P graph. Also we shall allow reasonable dependence of running time on the property of the objective function.

7.2 Algorithm: Description and Performance Analysis

We describe a Gossip algorithm for solving the optimization problem with the above stated desired properties. The algorithm is based on the Lagrange dual problem. Due to the separable objective function, its dual problem can be decomposed so that an individual node can recover the value of its variable in a primal solution from a dual feasible solution. The dual problem is solved via a dual ascent algorithm. The standard approach for designing such an algorithm only leads to a distributed algorithm with respect to the constraint graph of the problem. To design a Gossip algorithm there are two main challenges: (a) making the algorithm distributed with respect to the network graph G, and (b) respecting the non-negativity constraints on the variables.

The first challenge is resolved by utilizing the Gossip summation algorithm from *Separable function computation* as a subroutine. In a sense, the summation algorithm provides 'a distributed layer' for solving the optimization problem. The second challenge is resolved by use of a barrier function that is inspired by (centralized) interior-point mathematical programming algorithms.

In what follows, we start with some necessary notations and preliminaries. Then, we shall describe the algorithm and its properties. Finally, we shall provide proof of its properties in detail.

7.2.1 Notations and Preliminaries

For a vector $\mathbf{x} \in \mathbf{R}^n$, by $\|\mathbf{x}\|$ we denote the ℓ_2-norm of the vector. The ball of radius r around $\mathbf{x}$ is defined as $B(\mathbf{x}, r) = \{\mathbf{y} : \|\mathbf{y} - \mathbf{x}\| \leq r\}$. For a real matrix M, we write $\sigma_{\min}(M)$ and $\sigma_{\max}(M)$ to denote the smallest and largest singular values, respectively, of M, so that $\sigma_{\min}(M)^2$ and $\sigma_{\max}(M)^2$ are the smallest and largest eigenvalues of $M^T M$. Note that $\sigma_{\min}(M) = \min\{\|Mz\| \mid \|z\| = 1\}$ and $\sigma_{\max}(M) = \max\{\|Mz\| \mid \|z\| = 1\}$. If M is symmetric, then the singular values and the eigenvalues of M coincide, so $\sigma_{\min}(M)$ and $\sigma_{\max}(M)$ are the smallest and largest eigenvalues of M.

Associated with the primal problem (P) is the Lagrangian function $L(\mathbf{x}, \lambda, \nu) = f(\mathbf{x}) + \lambda^T(A\mathbf{x} - \mathbf{b}) - \nu^T\mathbf{x}$, which is defined for $\lambda \in \mathbf{R}^m$

and $\nu \in \mathbf{R}^n$, and the Lagrange dual function

$$g(\lambda,\nu) = \inf_{\mathbf{x}\in\mathbf{R}^n_+} L(\mathbf{x},\lambda,\nu) = -\mathbf{b}^T\lambda + \sum_{i=1}^n \inf_{x_i\in\mathbf{R}_+} \left(f_i(x_i) + \left(a_i^T\lambda - \nu_i\right)x_i\right).$$

The following problem is the Lagrange dual problem to (P).

$$\begin{array}{ll} \text{maximize} & g(\lambda,\nu) \\ \text{subject to} & \nu_i \geq 0, \quad i = 1,\ldots,n \end{array} \tag{D}$$

Although, we seek a solution to the primal problem (P), to avoid directly enforcing the non-negativity constraints, we introduce a logarithmic barrier. For a parameter $\theta > 0$, we consider the following primal barrier problem.

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) - \theta\displaystyle\sum_{i=1}^n \ln x_i \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \end{array} \tag{P_θ}$$

The Lagrange dual function corresponding to (P_θ) is

$$g_\theta(\lambda) = -\mathbf{b}^T\lambda + \sum_{i=1}^n \inf_{x_i\in\mathbf{R}_{++}} \left(f_i(x_i) - \theta\ln x_i + a_i^T\lambda x_i\right),$$

and the associated Lagrange dual problem is the following unconstrained optimization problem.

$$\begin{array}{ll} \text{maximize} & g_\theta(\lambda) \\ \text{over} & \lambda \in \mathbf{R}^m. \end{array} \tag{D_θ}$$

We assume that the primal barrier problem (P_θ) is feasible; that is, there exists a vector $\mathbf{x} \in \mathbf{R}^n_+$ such that $A\mathbf{x} = \mathbf{b}$. Under this assumption, the optimal value of (P_θ) is finite, and Slater's condition implies that the dual problem (D_θ) has the same optimal value, and there exists a dual solution λ^* that achieves this optimal value [9]. Furthermore, because (D_θ) is an unconstrained maximization problem with a strictly concave objective function, the optimal solution λ^* is unique.

7.2.1.1 Some useful properties

For a vector of dual variables $\lambda \in \mathbf{R}^m$, let $\mathbf{x}(\lambda) \in \mathbf{R}^n_{++}$ denote the corresponding primal minimizer in the the Lagrange dual function: for $i = 1, \ldots, n$,

$$x_i(\lambda) = \arg \inf_{x_i \in \mathbf{R}_{++}} \left(f_i(x_i) - \theta \ln x_i + a_i^T \lambda x_i \right). \tag{7.1}$$

For this primal $\mathbf{x}(\lambda)$ based on dual λ, we denote the 'violation of equality' $\|A\mathbf{x}(\lambda) - \mathbf{b}\|$, as $p(\lambda)$. Now, we can solve for each $x_i(\lambda)$ explicitly. As $f_i(x_i) - \theta \ln x_i + a_i^T \lambda x_i$ is convex in x_i,

$$f_i'(x_i(\lambda)) - \frac{\theta}{x_i(\lambda)} + a_i^T \lambda = 0. \tag{7.2}$$

Define $h_i \colon \mathbf{R}_{++} \to \mathbf{R}$ by $h_i(x_i) = f_i'(x_i) - \theta / x_i$; since f_i is convex, h_i is strictly increasing and hence has a well-defined and strictly increasing inverse. We then have

$$x_i(\lambda) = h_i^{-1} \left(-a_i^T \lambda \right).$$

Also, we assume that, given a vector λ, a node i can compute $x_i(\lambda)$. This is reasonable since computing $x_i(\lambda)$ is simply an unconstrained convex optimization problem in a single variable (7.1), which can be done by several methods, such as Newton's method.

Next, in our convergence analysis, we will argue about the gradient of the Lagrange dual function g_θ. A calculation shows that

$$\nabla g_\theta(\lambda) = -\mathbf{b} + \sum_{i=1}^n a_i x_i(\lambda) = A\mathbf{x}(\lambda) - \mathbf{b}. \tag{7.3}$$

We will use $p(\lambda)$ to denote $\|\nabla g_\theta(\lambda)\| = \|A\mathbf{x}(\lambda) - \mathbf{b}\|$ for a vector $\lambda \in \mathbf{R}^m$. We note that at the optimal dual solution λ^*, we have $p(\lambda^*) = 0$ and $A\mathbf{x}(\lambda^*) = \mathbf{b}$.

To control the rate of decrease in the gradient norm $p(\lambda)$, we must understand the Hessian of g_θ. For $j_1, j_2 \in \{1, \ldots, m\}$, component (j_1, j_2) of the Hessian $\nabla^2 g_\theta(\lambda)$ of g_θ at a point λ is

$$\frac{\partial g_\theta(\lambda)}{\partial \lambda_{j_1} \partial \lambda_{j_2}} = \sum_{i=1}^n A_{j_1 i} \frac{\partial x_i(\lambda)}{\partial \lambda_{j_2}} = -\sum_{i=1}^n A_{j_1 i} A_{j_2 i} \left(h_i^{-1} \right)' \left(-a_i^T \lambda \right). \tag{7.4}$$

As the functions h_i^{-1} are strictly increasing,

$$\min_{\ell=1,\ldots,n} \left(\left(h_\ell^{-1}\right)' \left(-a_\ell^T \lambda\right) \right) > 0.$$

Hence, for any $\mu \in \mathbf{R}^m$,

$$\begin{aligned}
\mu^T \nabla^2 g_\theta(\lambda) \mu &= \sum_{j_1=1}^m \mu_{j_1} \sum_{j_2=1}^m \frac{\partial g_\theta(\lambda)}{\partial \lambda_{j_1} \partial \lambda_{j_2}} \mu_{j_2} \\
&= -\sum_{j_1=1}^m \mu_{j_1} \sum_{j_2=1}^m \sum_{i=1}^n A_{j_1 i} A_{j_2 i} \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right) \mu_{j_2} \\
&= -\sum_{i=1}^n \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right) \sum_{j_1=1}^m A_{j_1 i} \mu_{j_1} \sum_{j_2=1}^m A_{j_2 i} \mu_{j_2} \\
&= -\sum_{i=1}^n \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right) \left(a_i^T \mu\right)^2 \\
&\leq -\min_{\ell=1,\ldots,n} \left(\left(h_\ell^{-1}\right)' \left(-a_\ell^T \lambda\right) \right) \left(A^T \mu\right)^T \left(A^T \mu\right) < 0, \quad (7.5)
\end{aligned}$$

and g_θ is a strictly a concave function.

7.2.2 Description of Algorithm

7.2.2.1 Basic algorithm

We consider an iterative algorithm for obtaining an approximate solution to (P), which uses gradient ascent for the dual barrier problem (D_θ). The algorithm generates a sequence of feasible solutions λ^0, λ^1, λ^2, ... for (D_θ), where λ^0 is the initial vector. To update λ^{k-1} to λ^k in an iteration k, the algorithm uses the gradient $\nabla g_\theta\left(\lambda^{k-1}\right)$ to determine the direction of the difference $\lambda^k - \lambda^{k-1}$. We assume that the algorithm is given as inputs to the initial point λ^0, and an accuracy parameter ε, such that $\varepsilon \in (0,1)$. The goal of the algorithm is to find a point $\mathbf{x} \in \mathbf{R}_+^n$ that is nearly feasible in the sense that $\|A\mathbf{x} - \mathbf{b}\| \leq \varepsilon \|\mathbf{b}\|$, and that has objective function value close to that of an optimal feasible point.

In this section, we describe the operation of the algorithm under the assumption that the algorithm has knowledge of certain parameters that affect its execution and performance. We refer to an execution of

the algorithm with a particular set of parameters as an *inner run* of the algorithm. To address the fact that these parameters would not be available to the algorithm at the outset, we add an *outer loop* to the algorithm. The outer loop uses binary search to find appropriate values for the parameters, and performs an inner run for each set of parameters encountered during the search. Section 7.2.2.3 discusses the operation of the outer loop of the algorithm. The choice of these parameters is inspired by convergence analysis of the algorithm and that is the reason for defering the discussion of setting parameters later.

An inner run of the algorithm consists of a sequence of iterations. Iteration k, for $k = 1, 2, \ldots$, begins with a current vector of dual variables λ^{k-1}, from which each node i computes $x_i\left(\lambda^{k-1}\right)$. Let $s^{k-1} = Ax\left(\lambda^{k-1}\right)$, so that by (7.3) $\nabla g_\theta\left(\lambda^{k-1}\right) = s^{k-1} - b$.

In order for the algorithm to perform gradient ascent, each node must compute the vector s^{k-1}. A component $s_j^{k-1} = \sum_{i=1}^{n} A_{ji} x_i\left(\lambda^{k-1}\right)$ of s^{k-1} is the sum of the values $y_i = A_{ji} x_i\left(\lambda^{k-1}\right)$ for those nodes i such that $A_{ji} > 0$. This is where we need 'distributed layer' provided by the summation algorithm described in *Separable function computation.*

Specifically, nodes apply this summation Gossip algorithm (m times, one for each component) to compute a vector $\hat{s}^{k-1}$, where $\hat{s}_j^{k-1}$ is an estimate of s_j^{k-1} for $j = 1, \ldots, m$. Recall that the summation algorithm takes as input parameters an accuracy ε_1 and an error probability δ. They provide estimate $\hat{s}_j^{k-1}$ of s_j^{k-1} for a particular value of j so that

$$(1 - \varepsilon_1)\, s_j^{k-1} \leq \hat{s}_j^{k-1} \leq (1 + \varepsilon_1)\, s_j^{k-1} \tag{7.6}$$

with probability at least $1 - \delta$. This computation takes $O(\varepsilon_1^{-2} \log \delta^{-2} / \Phi(P))$ time for $\delta \leq 1/n$ (which will be the case here). Recall that P denotes the information exchange probability matrix utilized by the Gossip algorithm.

In the analysis of an inner run, we assume that each invocation of the summation routine succeeds, so that (7.6) is satisfied. Provided we choose δ sufficiently small (see Section 7.2.2.3), this assumption will hold with high probability.

A description of an iteration k of an inner run of the algorithm is shown in Figure 7.1. The values for the step size t and the error

Inner run: Iteration k

1. For $j = 1, \ldots, m$, the nodes compute an estimate $\hat{s}_j^{k-1}$ of $s_j^{k-1} = \sum_{i=1}^n A_{ji} x_i \left(\lambda^{k-1}\right)$.
2. The nodes check the following two stopping conditions.

$$(1-\varepsilon_1)\left(1-\frac{2}{3}\varepsilon\right)\|\mathbf{b}\| \leq \left\|\hat{s}^{k-1}\right\| \leq (1+\varepsilon_1)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\|. \tag{7.7}$$

$$\left\|\hat{s}^{k-1}-\mathbf{b}\right\| \leq \left(\frac{2}{3}\varepsilon+\varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\right)\|\mathbf{b}\|. \tag{7.8}$$

   If both conditions (7.7) and (7.8) are satisfied, the inner run terminates, producing as output the vector $\mathbf{x}\left(\lambda^{k-1}\right)$.
3. The nodes update the dual vector by setting $\Delta\lambda^{k-1} = \hat{s}^{k-1} - b$, and $\lambda^k = \lambda^{k-1} + t\Delta\lambda^{k-1}$.

Fig. 7.1 The kth iteration of an inner run.

tolerance ε_1 will follow next. An inner run is essentially standard gradient ascent, where the stopping criterion (sufficiently small gradient norm) is modified to reflect the potential error in nodes' estimates of the gradient.

7.2.2.2 Choosing the parameters

The step size t and the convergence rate of our algorithm are governed by the variation in curvature of the Lagrange dual function. Intuitively, regions of large curvature necessitate a small step size to guarantee convergence, and if small steps are taken in regions with small curvature, then progress toward an optimal solution is slow. Examining the Hessian of the Lagrange dual function (7.4), we see that curvature variation depends both on variation in $(h_i^{-1})'$, which roughly corresponds to variation in the curvature of the f_i's, and on the variation in the singular values of A^T. Precisely, note that

$$\left(h_i^{-1}\right)'\left(-a_i^T\lambda\right) = \frac{1}{h_i'\left(h_i^{-1}\left(-a_i^T\lambda\right)\right)} = \frac{1}{f_i''\left(h_i^{-1}\left(-a_i^T\lambda\right)\right) + \frac{\theta}{\left(h_i^{-1}\left(-a_i^T\lambda\right)\right)^2}},$$

and, for a distance $r > 0$, define

$$q(r) = \min_{\ell=1,\ldots,n} \min_{\lambda\in B(\lambda^*,r)} \left(h_i^{-1}\right)'\left(-a_i^T\lambda\right)$$

$$Q(r) = \max_{\ell=1,\ldots,n} \max_{\lambda\in B(\lambda^*,r)} \left(h_i^{-1}\right)'\left(-a_i^T\lambda\right)$$

Our step size and convergence rate will depend on a parameter $R \geq 1$, defined as

$$R = \frac{Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2}{q\left(\|\lambda^0 - \lambda^*\|\right)\sigma_{\min}\left(A^T\right)^2}.$$

The parameter R measures the maximum curvature variation of the Lagrange dual function only in a ball of radius $\left\|\lambda^0 - \lambda^*\right\|$ around the optimal dual solution λ^*; this is because the sequence of dual solutions generated by our algorithm grows monotonically closer to λ^*, and we are concerned only with variation in the region in which our algorithm executes (as opposed to the entire feasible region, which is all of $\mathbf{R}^m$). Thus a better initial estimate of the optimal dual solution yields a tighter bound on curvature variation and a better convergence result.

In the convergence analysis that will follow the algorithm description, first we shall assume that the inner run *knows* values of numerator and denominator of R. Later in the analysis, justification for this assumption will be provided by means of a binary search based algorithm that will estimate it.

Now, define $\alpha = 1/6R$. For the summation subroutine, nodes use the accuracy parameter $\varepsilon_1 = \varepsilon\alpha/3$, where ε is the error tolerance given to the distributed algorithm. For gradient ascent, nodes compute and employ the following step size:

$$t = \frac{\left(1 - \alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\right)^2 - \frac{1}{6}\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\left(1 + \alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\right)}{R\left(1 + \alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\right)^2\left(Q\left(\|\lambda^0 - \lambda^*\|\right)\sigma_{\max}\left(A^T\right)^2\right)}. \tag{7.9}$$

Here, $t > 0$ since $\alpha \leq 1/6$ and $\varepsilon \leq 1$. An inner run continues to execute iterations for increasing values of k until both stopping conditions are satisfied, or the outer loop of the algorithm terminates the inner run as described in Section 7.2.2.3.

7.2.2.3 Outer loop and stopping conditions

Here, we primarily describe the outer loop of the algorithm that leads to determination of parameters used by the inner loop. The termination or stopping conditions for the algorithm will be described as well.

First, we describe the outer loop of the algorithm. The purpose of the outer loop is to invoke inner runs with various parameter values, and to terminate runs if they do not end in the allotted number of iterations.

As the outer loop does not know the values $q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2$ and $Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2$, it uses binary search to choose the parameter values for the inner runs. The algorithm (and its analysis) remains valid if we replace the former product with a lower bound on it, and the latter product with an upper bound on it. Let $U > 0$ be an upper bound on the ratio between the largest and the smallest possible values of these two products.

The outer loop enumerates $\log U$ possible values $q_1, q_2, \ldots, q_{\log U}$ for $q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2$, with $q_{\ell+1} = 2q_\ell$ for each ℓ. Similarly, it considers values $Q_1, Q_2, \ldots, Q_{\log U}$ for $Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2$. For each pair of values (q_{ℓ_1}, Q_{ℓ_2}) such that $q_{\ell_1} \leq Q_{\ell_2}$, it computes an upper bound $T(q_{\ell_1}, Q_{\ell_2})$ on the number of iterations required for an inner run with these parameter values. As stated later in Theorem 7.8, this value is bounded above as

$$T(q_{\ell_1}, Q_{\ell_2}) = O\left(\frac{Q_{\ell_2}^2}{q_{\ell_1}^2}\log\left(\frac{p(\lambda^0)}{\varepsilon\|\mathbf{b}\|}\right)\right).$$

Recall that $p(\lambda) = \|A\mathbf{x}(\lambda) - \mathbf{b}\|$.

Now, the outer loop sorts the $T(q_{\ell_1}, Q_{\ell_2})$ values, and executes inner runs according to this sorted order. When an inner run is executed with parameter values (q_{ℓ_1}, Q_{ℓ_2}), the outer loop lets it execute for $T(q_{\ell_1}, Q_{\ell_2})$ iterations. If it terminates due to the stopping conditions being satisfied within this number of iterations, then by Theorem 7.8 the solution $x(\lambda)$ produced will satisfy

$$\|A\mathbf{x}(\lambda) - \mathbf{b}\| \leq \varepsilon\|\mathbf{b}\|,$$

and so the outer loop outputs this solution. On the other hand, if the stopping conditions for the inner run are not satisfied within the allotted number of iterations, the outer loop terminates the inner run, and then executes the next inner run with new parameter values according to the order induced by $T(q_{\ell_1}, Q_{\ell_2})$.

By the choice of q_{ℓ_1} and Q_{ℓ_2}, there exist $q_{\ell_1^*}$ and $Q_{\ell_2^*}$ such that $q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2/2 \leq q_{\ell_1^*} \leq q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2$ and $Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2 \leq Q_{\ell_2^*} \leq 2Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2$. For the parameter pair $(q_{\ell_1^*}, Q_{\ell_2^*})$, $T(q_{\ell_1^*}, Q_{\ell_2^*})$ is, up to constant factors, the bound in Theorem 7.8. As we shall see, the bound established in Theorem 7.8 in the future section, will imply that for given value of R, the inner loop algorithm must terminate within iterations $O(R^2 \log(p(\lambda^0)/\varepsilon\|\mathbf{b}\|))$. Therefore, when the outer loop reaches the pair $(q_{\ell_1^*}, Q_{\ell_2^*})$, the corresponding inner run will terminate with the stopping conditions satisfied in the number of iterations specified in Theorem 7.8. Since the inner runs executed prior to this one will also be terminated in at most this number of iterations, and there are at most $\log^2 U$ such runs, we obtain the following upper bound on the total number of iterations executed by the algorithm.

Lemma 7.1. The total number of iterations executed in all the inner runs initiated by the outer loop is

$$O\left(R^2 \log\left(\frac{p\left(\lambda^0\right)}{\varepsilon\|b\|}\right)\log^2 U\right).$$

Now, we describe the only remaining aspect of the termination condition. Recall that in an iteration k of an inner run, the nodes must compute an estimate $\hat{s}_j^{k-1}$ for each of the m components of the vector s_j^{k-1}. As such, the summation routine must be invoked m times in each iteration. Recall that the algorithm should terminate after $O(\varepsilon_1^{-2}\log^2\delta^{-1}/\Phi(P))$ iterations. Therefore, if nodes have estimate of upper bound on n (number of nodes) and lower bound on $\Phi(P)$ then they can determine when to stop. It should be noted that a trivial lower bound of $1/n$ on $\Phi(P)$ can be utilized for any reasonable graph; and hence only an upper bound on n is necessary information for determining stopping condition.

7.2.3 Performance: Convergence and Correctness

Here, we describe the correctness and convergence properties of the algorithm described above. To this end, we need to set the value of δ utilized in the summation algorithm for the failure probability. The Lemma 7.1 and union bound suggests that in order for all the summation computation to satisfy condition (7.6) with probability at least $1 - 1/n^2$, it is sufficient to set

$$\delta \leq \left(n^2 m R^2 \log \left(\frac{p\left(\lambda^0\right)}{\varepsilon \|b\|} \right) \log^2 U \right)^{-1}.$$

We will also set $\varepsilon_1 = \varepsilon\alpha/3$. This along with the above choice of δ leads to the following bound on each summation subroutine:

$$O\left(\frac{R^2}{\varepsilon^2 \Phi(P)} \left(\log \left(nmR \log \left(\frac{p\left(\lambda^0\right)}{\varepsilon \|b\|} \right) \log U \right) \right)^2 \right) \triangleq O^* \left(\frac{R^2}{\varepsilon^2 \Phi(P)} \right),$$

where we have ignored poly-logarithmic terms in the $O^*(\cdot)$ notation to highlight the key dependence of the running time. Here, note that the total number of operations performed by the algorithm is m times larger than the above stated bound since there are m summations we need to perform for each iteration of the inner run. Thus resulting algorithm (with all choices of parameters and stopping condition) has the following convergence and correctness property.

Theorem 7.2 (Main Theorem). The algorithm produces a solution $\mathbf{x}(\lambda)$ that satisfies the following properties with high probability (i.e., probability $\geq 1 - 1/n^2$) in time $O^*(m\varepsilon^{-2}R^2/\Phi(P))$:

(a) $\|A\mathbf{x}(\lambda) - b\| \leq \varepsilon\|\mathbf{b}\|$,
(b) $f(\mathbf{x}(\lambda)) \leq \text{OPT} + \varepsilon\|\mathbf{b}\|\|\lambda\| + n\theta$,

where OPT is the cost of optimal assignment of the primal optimization problem (P) and $\|\lambda\| \leq \|\lambda^0\| + 2\|\lambda^* - \lambda^0\|$.

7.2.4 Analysis of Algorithm

In this section, we shall establish the proof of Theorem 7.2. In order to establish the proof, we will state a sequence of results (in terms of Lemmas and a Theorem) whose proofs are defered to the next section. Using these results, we will conclude the proof.

To this end, first we wish to establish bound on the number of iterations required by the algorithm used for inner run to obtain a solution $\mathbf{x}\left(\lambda^k\right)$ such that $\left\|A\mathbf{x}\left(\lambda^k\right) - \mathbf{b}\right\| \leq \varepsilon\|\mathbf{b}\|$, and we also prove an approximation bound on the objective function value of the final solution. We assume in this analysis that the summation subroutine used by an inner run is always successful; that is, (7.6) holds for every sum computation. Furthermore, we assume that an inner run executes until both stopping conditions are satisfied.

The possibility of an inner run being terminated by the outer loop was addressed in Section 7.2.2.3. As explained there, in what follows we will establish bound on number of iterations taken by inner loop for given value of R in Theorem 7.8. And hence for a correct pair of values q_{ℓ_1}, Q_{ℓ_2} such that $R/4 \leq Q_{\ell_2}/q_{\ell_1} \leq 4R$, the outer loop will not terminate the inner loop before both stopping conditions are satisfied. Therefore, in the analysis that follows to establish the validity of this claim we can safely ignore the possibility of outer loop terminating inner loop. We shall also assume that the value of R is known in the analysis to follow.

To this end, first we consider the extent to which $\Delta\lambda^{k-1}$ deviates from the correct gradient $\nabla g_\theta\left(\lambda^{k-1}\right)$, provided that the inner run does not terminate in iteration k. To this end, let $u^{k-1} = \hat{s}^{k-1} - s^{k-1}$ be a vector representing the error in the computation of s^{k-1}. Note that $\Delta\lambda^{k-1} = \nabla g_\theta\left(\lambda^{k-1}\right) + u^{k-1}$.

Lemma 7.3. If the stopping conditions (7.7) and (7.8) are not both satisfied in iteration k, then

$$\left\|u^{k-1}\right\| \leq \alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \tag{7.10}$$

and

$$\left(1-\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \leq \left\|\Delta\lambda^{k-1}\right\| \leq \left(1+\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|. \quad (7.11)$$

Next, we develop some inequalities that will be useful in understanding the evolution of an inner run from one iteration to the next.

Lemma 7.4. For any two points $\rho^1, \rho^2 \in B\left(\lambda^*, \left\|\lambda^0-\lambda^*\right\|\right)$,

$$\left\|A\mathbf{x}\left(\rho^2\right)-A\mathbf{x}\left(\rho^1\right)\right\| \leq Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\left\|\rho^2-\rho^1\right\| \quad (7.12)$$

and

$$\left(\nabla g_\theta\left(\rho^2\right)-\nabla g_\theta\left(\rho^1\right)\right)^T\left(\rho^2-\rho^1\right) \leq -q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2\left\|\rho^2-\rho^1\right\|^2. \quad (7.13)$$

Corollary 7.5. For any $\lambda \in B\left(\lambda^*, \left\|\lambda^0-\lambda^*\right\|\right)$,

$$\left\|\nabla g_\theta(\lambda)\right\| \leq Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\left\|\lambda-\lambda^*\right\|,$$

and

$$\nabla g_\theta(\lambda)^T\left(\lambda-\lambda^*\right) \leq -q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2\left\|\lambda-\lambda^*\right\|^2.$$

We now show that all the dual vectors generated by an inner run are as close to the optimal solution λ^* as the initial point λ^0.

Lemma 7.6. For each iteration k executed by an inner run, $\lambda^{k-1} \in B\left(\lambda^*, \left\|\lambda^0-\lambda^*\right\|\right)$.

To establish that an inner run makes progress as it executes iterations, we show that the norm of the gradient of $g_\theta\left(\lambda^k\right)$, $p\left(\lambda^k\right) = \left\|A\mathbf{x}\left(\lambda^k\right)-\mathbf{b}\right\|$, decreases by a multiplicative factor in each iteration.

Lemma 7.7. For each iteration k executed by an inner run in which the stopping conditions are not satisfied,

$$\left\|\nabla g_\theta\left(\lambda^k\right)\right\| \leq \left(\sqrt{1-\frac{1}{4R^2}}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|.$$

Lemma 7.7 implies an upper bound on the number of iterations executed by an inner run.

Theorem 7.8. An inner run terminates after

$$O\left(R^2 \log\left(\frac{p\left(\lambda^0\right)}{\varepsilon\|b\|}\right)\right)$$

iterations with a solution $\mathbf{x}(\lambda)$ such that $\|A\mathbf{x}(\lambda) - \mathbf{b}\| \leq \varepsilon\|\mathbf{b}\|$.

Finally, we bound the difference between the objective function value of the solution produced by an inner run and the optimal value of the primal problem. Let OPT denote the optimal value of (P).

Corollary 7.9. The objective function value of the solution $\mathbf{x}(\lambda)$ produced by an inner run satisfies

$$f(\mathbf{x}(\lambda)) \leq \mathrm{OPT} + \varepsilon\|\mathbf{b}\|\|\lambda\| + n\theta.$$

Since the dual solution λ produced by the algorithm satisfies $\|\lambda\| \leq \|\lambda^0\| + 2\|\lambda^0 - \lambda^*\|$, by choosing the parameters ε and θ appropriately, the approximation error can be made as small as desired (though, of course, the convergence time increases as each of these parameters decreases).

Proof. [Theorem 7.2] In order to prove the claims of the Theorem, in light of Theorem 7.8 and Corollary 7.9, it is sufficient to establish that our algorithm indeed executes the inner loop with the a good estimate

of R upto $O^*(R^2)$ iterations with high probability. As explained in Section 7.2.2.3, the appropriate choice of δ along with binary search over proper set of $\log^2 U$, q_{ℓ_1}, Q_{ℓ_2} pairs the algorithm will indeed have this property. Therefore, the output of the algorithm is indeed satisfying (a) and (b) properties with high probability. The bound on computation time follows from the accounting performed in Section 7.2.3 along with Lemma 7.1. This completes the proof of Theorem 7.2. □

7.2.4.1 Remaining proofs

Here, we present the remaining proofs of the results stated above to establish Theorem 7.2.

Proof of Lemma 7.3: If (7.7) is not satisfied, then

$$\left\|\hat{s}^{k-1}\right\| < (1-\varepsilon_1)\left(1-\frac{2}{3}\varepsilon\right)\|\mathbf{b}\| \text{ or } \left\|\hat{s}^{k-1}\right\| > (1+\varepsilon_1)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\|,$$

and so, by (7.6),

$$\left\|s^{k-1}\right\| < \left(1-\frac{2}{3}\varepsilon\right)\|\mathbf{b}\| \text{ or } \left\|s^{k-1}\right\| > \left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\|.$$

By the triangle inequality, this implies that

$$\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| = \left\|s^{k-1}-\mathbf{b}\right\| \geq \left|\left\|s^{k-1}\right\| - \|\mathbf{b}\|\right| > \frac{2}{3}\varepsilon\|\mathbf{b}\|. \tag{7.14}$$

Suppose that (7.7) is satisfied and (7.8) is not satisfied. Note that (7.6) implies that $\left\|u^{k-1}\right\| \leq \varepsilon_1 \left\|s^{k-1}\right\|$, and so (7.7) and (7.6) yield

$$\left\|u^{k-1}\right\| \leq \varepsilon_1 \left\|s^{k-1}\right\| \leq \varepsilon_1 \left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\|. \tag{7.15}$$

By the triangle inequality and (7.15),

$$\begin{aligned}\left\|\Delta\lambda^{k-1}\right\| = \left\|\hat{s}^{k-1}-\mathbf{b}\right\| &= \left\|\nabla g_\theta\left(\lambda^{k-1}\right)+u^{k-1}\right\| \\ &\leq \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| + \left\|u^{k-1}\right\| \\ &\leq \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| + \varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\|,\end{aligned}$$

and so the fact that (7.8) is not satisfied implies that

$$\begin{aligned}\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| &\geq \left\|\hat{s}^{k-1}-\mathbf{b}\right\| - \varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\| \\ &> \left(\frac{2}{3}\varepsilon+\varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\right)\|\mathbf{b}\| \\ &\quad -\varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\| \\ &= \frac{2}{3}\varepsilon\|\mathbf{b}\|. \end{aligned} \tag{7.16}$$

Combining (7.14) and (7.16), it follows that if the two stopping conditions are not both satisfied, then

$$\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| > \frac{2}{3}\varepsilon\|\mathbf{b}\|.$$

Now, applying the triangle inequality yields

$$\begin{aligned}\left\|u^{k-1}\right\| &\leq \varepsilon_1\left\|s^{k-1}\right\| \\ &\leq \varepsilon_1\left(\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| + \|\mathbf{b}\|\right) \\ &\leq \varepsilon_1\left(1+\frac{3}{2\varepsilon}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \\ &= \alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|, \end{aligned}$$

where the last equality follows from the fact that $\varepsilon_1 = \varepsilon\alpha/3$. This proves inequality in (7.10), and the inequalities in (7.11) follow from (7.10) and the triangle inequality. □

Proof of Lemma 7.4: Let $\left[\rho^1,\rho^2\right]$ denote the line segment joining ρ^1 and ρ^2. Since $B\left(\lambda^*,\left\|\lambda^0-\lambda^*\right\|\right)$ is a convex set, for any $i=1,\ldots,n$ and any $\lambda\in\left[\rho^1,\rho^2\right]$, $\left(h_i^{-1}\right)'\left(-a_i^T\lambda\right) \leq Q\left(\left\|\lambda^0-\lambda^*\right\|\right)$. As a result,

$$\begin{aligned}\left|x_i\left(\rho^2\right)-x_i\left(\rho^1\right)\right| &= \left|h_i^{-1}\left(-a_i^T\rho^2\right)-h_i^{-1}\left(-a_i^T\rho^1\right)\right| \\ &\leq Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\left|a_i^T\left(\rho^2-\rho^1\right)\right| \\ &= Q\left(\left\|\lambda^0-\lambda^*\right\|\right)a_i^T\rho, \end{aligned}$$

where $\rho \in \mathbf{R}^m$ is defined by $\rho_j = |\rho_j^2 - \rho_j^1|$ for $j = 1, \ldots, m$. This implies that

$$\begin{aligned}\left\|A\mathbf{x}\left(\rho^2\right) - A\mathbf{x}\left(\rho^1\right)\right\| &= \left\|A\left(\mathbf{x}\left(\rho^2\right) - \mathbf{x}\left(\rho^1\right)\right)\right\| \\ &\leq Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\left\|AA^T\rho\right\| \\ &\leq Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\max}\left(AA^T\right)\|\rho\| \\ &= Q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\left\|\rho^2 - \rho^1\right\|,\end{aligned}$$

and the inequality in (7.12) is proved.

For any $\lambda \in \left[\rho^1, \rho^2\right]$ and any $\mu \in \mathbf{R}^m$, a calculation analogous to the one in (7.5) yields

$$\begin{aligned}\mu^T\nabla^2 g_\theta(\lambda)\mu &= -\sum_{j_1=1}^{m}\mu_{j_1}\sum_{j_2=1}^{m}\sum_{i=1}^{n}A_{j_1 i}A_{j_2 i}\left(h_i^{-1}\right)'\left(-a_i^T\lambda\right)\mu_{j_2} \\ &\leq -q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\mu^T AA^T\mu \\ &\leq -q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\min}\left(AA^T\right)\|\mu\| \\ &= -q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2\|\mu\| \qquad (7.17)\end{aligned}$$

From the second-order expansion of the function g_θ, there exist vectors $\mu^1, \mu^2 \in \left[\rho^1, \rho^2\right]$ such that

$$\begin{aligned}g_\theta\left(\rho^2\right) &= g_\theta\left(\rho^1\right) + \nabla g_\theta\left(\rho^1\right)^T\left(\rho^2 - \rho^1\right) \\ &\quad + \frac{1}{2}\left(\rho^2 - \rho^1\right)^T\nabla^2 g_\theta\left(\mu^1\right)\left(\rho^2 - \rho^1\right) \\ g_\theta\left(\rho^1\right) &= g_\theta\left(\rho^2\right) + \nabla g_\theta\left(\rho^2\right)^T\left(\rho^1 - \rho^2\right) \\ &\quad + \frac{1}{2}\left(\rho^1 - \rho^2\right)^T\nabla^2 g_\theta\left(\mu^2\right)\left(\rho^1 - \rho^2\right)\end{aligned}$$

Adding the two equations and applying (7.17) yields

$$\begin{aligned}&\left(\nabla g_\theta\left(\rho^2\right) - \nabla g_\theta\left(\rho^1\right)\right)^T\left(\rho^2 - \rho^1\right) \\ &\quad = \frac{1}{2}\left(\rho^2 - \rho^1\right)^T\nabla^2 g_\delta\left(\mu^1\right)\left(\rho^2 - \rho^1\right) \\ &\qquad + \frac{1}{2}\left(\rho^1 - \rho^2\right)^T\nabla^2 g_\delta\left(\mu^2\right)\left(\rho^1 - \rho^2\right) \\ &\quad \leq -q\left(\left\|\lambda^0 - \lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2\left\|\rho^2 - \rho^1\right\|^2\end{aligned}$$

This establishes the inequality in (7.13) and completes the proof of the lemma. □

Proof of Corollary 7.5: This follows from an application of Lemma 7.4 with $\rho^1 = \lambda^*$ and $\rho^2 = \lambda$, using the additional observations that $\nabla g_\theta(\lambda) = A\mathbf{x}(\lambda) - \mathbf{b} = A\mathbf{x}(\lambda) - A\mathbf{x}(\lambda^*)$, and $\nabla g_\theta(\lambda^*) = 0$ because λ^* is an optimal solution to (D_θ). □

Proof of Lemma 7.6: The proof is by induction on k, the iteration number. For the base case $k = 1$, $\left\|\lambda^{k-1} - \lambda^*\right\| = \left\|\lambda^0 - \lambda^*\right\|$.

In the inductive case, we assume that the statement is true for an iteration k, and we show that it then holds for iteration $k+1$, where $k \geq 1$. As such, the inductive hypothesis is that $\lambda^{k-1} \in B\left(\lambda^*, \left\|\lambda^0 - \lambda^*\right\|\right)$. If the algorithm executes iteration $k+1$, then it does not terminate in iteration k, and $\lambda^k - \lambda^{k-1} = t\Delta\lambda^{k-1}$. The squared distance from λ^k to λ^* can be expressed as follows.

$$\begin{aligned}
\left\|\lambda^k - \lambda^*\right\|^2 &= \left\|\left(\lambda^k - \lambda^{k-1}\right) + \left(\lambda^{k-1} - \lambda^*\right)\right\|^2 \\
&= \left\|\lambda^{k-1} - \lambda^*\right\|^2 + \left\|\lambda^k - \lambda^{k-1}\right\|^2 \\
&\quad + 2\left(\lambda^k - \lambda^{k-1}\right)^T \left(\lambda^{k-1} - \lambda^*\right) \\
&= \left\|\lambda^{k-1} - \lambda^*\right\|^2 + t^2 \left\|\Delta\lambda^{k-1}\right\|^2 \\
&\quad + 2t\left(\Delta\lambda^{k-1}\right)^T \left(\lambda^{k-1} - \lambda^*\right) \qquad (7.18)
\end{aligned}$$

The third term in the right-hand side of (7.18) can be bounded from above by applying the inductive hypothesis, Corollary 7.5, Lemma 7.3, and the Cauchy–Schwarz inequality.

$$\begin{aligned}
&\left(\Delta\lambda^{k-1}\right)^T \left(\lambda^{k-1} - \lambda^*\right) \\
&\quad = \left(\nabla g_\theta\left(\lambda^{k-1}\right) + u^{k-1}\right)^T \left(\lambda^{k-1} - \lambda^*\right) \\
&\quad \leq -q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A^T\right)^2 \left\|\lambda^{k-1} - \lambda^*\right\|^2 + \left\|u^{k-1}\right\| \left\|\lambda^{k-1} - \lambda^*\right\|
\end{aligned}$$

$$\leq \left\|\lambda^{k-1} - \lambda^*\right\|^2 \left(\alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right) Q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\max}\left(A^T\right)^2 \right.$$
$$\left. -q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A^T\right)^2\right)$$

Substituting this inequality into (7.18), and again applying the inductive hypothesis and Lemma 7.3, yields

$$\begin{aligned}\left\|\lambda^k - \lambda^*\right\|^2 \leq \left\|\lambda^{k-1} - \lambda^*\right\|^2 \Bigg(1 + t^2 \left(1 + \alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\right)^2 \\ \times \left(Q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\max}\left(A^T\right)^2\right)^2 \\ + 2t\left(\alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right) Q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\max}\left(A^T\right)^2 \right. \\ \left. -q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A^T\right)^2\right)\Bigg).\end{aligned}$$

As $\alpha Q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\max}\left(A^T\right)^2 = q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A^T\right)^2/6$, we will have the sequence of inequalities $\left\|\lambda^k - \lambda^*\right\| \leq \left\|\lambda^{k-1} - \lambda^*\right\| \leq \left\|\lambda^0 - \lambda^*\right\|$ provided that

$$t \leq \frac{\left(2 - \frac{1}{3}\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\right) q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A^T\right)^2}{\left(1 + \alpha\left(\frac{1}{2} + \frac{\varepsilon}{3}\right)\right)^2 \left(Q\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\max}\left(A^T\right)^2\right)^2}.$$

The step size in (7.9) used by an inner run satisfies this inequality because $\varepsilon \leq 1$. This completes the proof of the inductive case and of the lemma. □

Proof of Lemma 7.7: If the stopping conditions are not satisfied in iteration k, then Lemma 7.6 implies that $\lambda^{k-1}, \lambda^k \in B\left(\lambda^*, \left\|\lambda^0 - \lambda^*\right\|\right)$. The squared norm of the gradient of g_θ at λ^k can be expressed as

$$\begin{aligned}\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2 &= \left\|\left(\nabla g_\theta\left(\lambda^k\right) - \nabla g_\theta\left(\lambda^{k-1}\right)\right) + \nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \\ &= \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 + \left\|\nabla g_\theta\left(\lambda^k\right) - \nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \\ &\quad + 2\left(\nabla g_\theta\left(\lambda^k\right) - \nabla g_\theta\left(\lambda^{k-1}\right)\right)^T \nabla g_\theta\left(\lambda^{k-1}\right). \qquad (7.19)\end{aligned}$$

An upper bound on the second term in the right-hand side of (7.19) follows from Lemmas 7.3 and 7.4.

$$\begin{aligned}
&\left\|\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \\
&\quad=\left\|\left(A\mathbf{x}\left(\lambda^k\right)-\mathbf{b}\right)-\left(A\mathbf{x}\left(\lambda^{k-1}\right)-\mathbf{b}\right)\right\| \\
&\quad=\left\|A\mathbf{x}\left(\lambda^k\right)-A\mathbf{x}\left(\lambda^{k-1}\right)\right\| \\
&\quad\le Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\left\|\lambda^k-\lambda^{k-1}\right\| \\
&\quad=tQ\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\left\|\Delta\lambda^{k-1}\right\| \\
&\quad\le t\left(1+\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|
\end{aligned} \tag{7.20}$$

To bound the third term in the right-hand side of (7.19), we again apply Lemmas 7.3 and 7.4.

$$\begin{aligned}
&\left(\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right)^T\nabla g_\theta\left(\lambda^{k-1}\right) \\
&\quad=\left(\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right)^T\left(\Delta\lambda^{k-1}-u^{k-1}\right) \\
&\quad\le -tq\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2\left\|\Delta\lambda^{k-1}\right\|^2 \\
&\qquad+\left\|u^{k-1}\right\|\left\|\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \\
&\quad\le -t\left(1-\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)^2 q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \\
&\qquad+t\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\left(1+\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)Q\left(\left\|\lambda^0-\lambda^*\right\|\right) \\
&\qquad\times\sigma_{\max}\left(A^T\right)^2\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2.
\end{aligned} \tag{7.21}$$

Substituting (7.20) and (7.21) in (7.19) yields

$$\begin{aligned}
\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2\le\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2&\left(1+t^2\left(1+\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)^2\right. \\
&\left.\times\left(Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\right)^2\right.
\end{aligned}$$

$$+ 2t\Bigg(\frac{1}{6}\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\left(1+\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right) - \left(1-\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)^2\Bigg) q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2\Bigg),$$

where we have used the fact that $\alpha Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2 = q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2/6$. For the step size t in (7.9), we have

$$\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2 \le \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \times \zeta,$$

where

$$\zeta = \left(1 - \left(\frac{\left(\left(1-\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)^2 - \frac{1}{6}\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\left(1+\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)\right) \times q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2}{\left(1+\alpha\left(\frac{1}{2}+\frac{\varepsilon}{3}\right)\right)Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2}\right)^2\right).$$

Since $\alpha \le 1/6$ and $\varepsilon \le 1$, it follows that

$$\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2 \le \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \left(1-\left(\frac{q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2}{2Q\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2}\right)^2\right),$$

and the proof is complete. □

Proof of Theorem 7.8: If an inner run terminates with a solution $\mathbf{x}(\lambda)$, then the stopping conditions (7.7) and (7.8) are both satisfied for the estimate $\hat{s} = s + u$ of the vector $s = A\mathbf{x}(\lambda)$. Applying (7.6) and the triangle inequality yields

$$\begin{aligned}\|A\mathbf{x}(\lambda) - \mathbf{b}\| = \|s-\mathbf{b}\| &\le \|\hat{s}-\mathbf{b}\| + \|u\| \le \|\hat{s}-\mathbf{b}\| + \varepsilon_1\|s\| \\ &\le \left(\frac{2}{3}\varepsilon + \varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\right)\|\mathbf{b}\| \\ &\quad + \varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\| \\ &= \left(\frac{2}{3}\varepsilon + \frac{2\varepsilon\alpha}{3}\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\right)\|\mathbf{b}\|.\end{aligned}$$

Because $\varepsilon \leq 1$ and $\alpha \leq 1/6$, $\varepsilon_1 = \varepsilon\alpha/3 \leq 1/18$, and so we obtain $\|A\mathbf{x}(\lambda) - \mathbf{b}\| \leq \varepsilon\|\mathbf{b}\|$.

Now, consider an iteration k such that $\left\|s^{k-1} - \mathbf{b}\right\| \leq (2/3)\varepsilon\|\mathbf{b}\|$. Since $\left|\left\|s^{k-1}\right\| - \|\mathbf{b}\|\right| \leq \left\|s^{k-1} - \mathbf{b}\right\|$, (7.6) implies that

$$(1-\varepsilon_1)\left(1-\frac{2}{3}\varepsilon\right)\|\mathbf{b}\| \leq \left\|\hat{s}^{k-1}\right\| \leq (1+\varepsilon_1)\left(1+\frac{2}{3}\varepsilon\right)\|\mathbf{b}\|,$$

and (7.7) is satisfied. Moreover,

$$\begin{aligned}\left\|\hat{s}^{k-1} - \mathbf{b}\right\| &\leq \left\|s^{k-1} - \mathbf{b}\right\| + \left\|u^{k-1}\right\| \leq \frac{2}{3}\varepsilon\|\mathbf{b}\| + \varepsilon_1\left\|s^{k-1}\right\| \\ &\leq \left(\frac{2}{3}\varepsilon + \varepsilon_1\left(1+\frac{2}{3}\varepsilon\right)\right)\|\mathbf{b}\| \\ &\leq \left(\frac{2}{3}\varepsilon + \varepsilon_1\left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right)\left(1+\frac{2}{3}\varepsilon\right)\right)\|\mathbf{b}\|,\end{aligned}$$

and so (7.8) is satisfied as well. Thus, if $\left\|s^{k-1} - \mathbf{b}\right\| \leq (2/3)\varepsilon\|\mathbf{b}\|$, then the inner run will terminate in iteration k.

Repeated application of Lemma 7.7 implies that, if an inner run does not terminate in or before an iteration k, then

$$\left\|\nabla g_\theta\left(\lambda^k\right)\right\| \leq \left(1-\frac{1}{4R^2}\right)^{\frac{k}{2}} p\left(\lambda^0\right).$$

For

$$k \geq 8R^2 \ln\left(\frac{3p\left(\lambda^0\right)}{2\varepsilon\|\mathbf{b}\|}\right),$$

we have $\left\|\nabla g_\theta\left(\lambda^k\right)\right\| \leq (2/3)\varepsilon\|\mathbf{b}\|$, and hence the stopping conditions will be satisfied and an inner run will terminate in the claimed number of iterations. □

Proof of Corollary 7.9: Given the solution $\mathbf{x}(\lambda)$ produced by an inner run, define a vector $\nu(\lambda) \in \mathbf{R}^n_{++}$ by, for all $i = 1, \ldots, n$,

$$\nu_i(\lambda) = \frac{\theta}{x_i(\lambda)}.$$

The pair $(\lambda, \nu(\lambda))$ is a feasible solution to the dual problem (D) with objective function value

$$\begin{aligned} g(\lambda, \nu(\lambda)) &= \inf_{\mathbf{x} \in \mathbf{R}_+^n} L(\mathbf{x}, \lambda, \nu(\lambda)) \\ &= -\mathbf{b}^T \lambda + \sum_{i=1}^{n} \inf_{x_i \in \mathbf{R}_+} \left(f_i(x_i) + \left(a_i^T \lambda - \frac{\theta}{x_i(\lambda)} \right) x_i \right). \end{aligned}$$

As the components of the vector $\mathbf{x}(\lambda)$ satisfy (7.2), we have $L(\mathbf{x}(\lambda), \lambda, \nu(\lambda)) = g(\lambda, \nu(\lambda))$.

From the definition of the Lagrangian and the fact that $(\lambda, \nu(\lambda))$ is feasible for (D),

$$\begin{aligned} f(\mathbf{x}(\lambda)) + \lambda^T (A\mathbf{x}(\lambda) - \mathbf{b}) - \nu(\lambda)^T \mathbf{x}(\lambda) \\ = L(\mathbf{x}(\lambda), \lambda, \nu(\lambda)) = g(\lambda, \nu(\lambda)) \leq \mathrm{OPT}. \end{aligned}$$

Applying the Cauchy–Schwarz inequality and Theorem 7.8 yields the claimed upper bound on the objective function value of the vector $x(\lambda)$.

$$\begin{aligned} f(\mathbf{x}(\lambda)) &\leq \mathrm{OPT} - \lambda^T (A\mathbf{x}(\lambda) - \mathbf{b}) + \nu(\lambda)^T \mathbf{x}(\lambda) \\ &\leq \|\lambda\| \, \|A\mathbf{x}(\lambda) - \mathbf{b}\| + \sum_{i=1}^{n} \left(\frac{\theta}{x_i(\lambda)} \right) x_i(\lambda) \\ &\leq \mathrm{OPT} + \varepsilon \|\mathbf{b}\| \|\lambda\| + n\theta. \end{aligned}$$

□

7.3 Historical Notes

The design of distributed algorithms for convex minimization with linear constraints has been of interest since the early 1960s. The essence of the work before the mid-1980s is well documented in the book by Rockafellar [62]. Rockafellar [62] describes distributed algorithms for *monotropic programs*, which are separable convex minimization problems with linear constraints. These algorithms leverage the decomposable structure of the Lagrange dual problem arising from the separable primal objective. This structure has also been used to design parallel and asynchronous algorithms for monotropic programs; see the book by Bertsekas and Tsitsiklis [5] for further details. All of these algorithms are by design distributed with respect to an appropriate constraint graph G_C, as opposed to an underlying network G. For the special case

of a network routing problem, the distributed algorithm of Gallager [24] is intuitively "closer" to being distributed with respect to G; however, it still requires direct access to route information and hence is fully distributed with respect to the constraint graph G_C only.

The network resource allocation problems that motivate the algorithm described here are special cases of monotropic programs. Kelly et al. [36] used these known distributed algorithmic solutions to explain the congestion control protocols for the resource allocation problem. Moreover, they show that in an idealized model with perfect feedback (in the form of packet drops) by network queues, these algorithms can also be interpreted as distributed over G. See also Garg and Young [26] for similar results that emphasize the rate of convergence to an optimal solution. See the book by Srikanthan [66] for further work on congestion control.

Flow control also serves as the motivation for the work of Bartal et al. [4] on distributed algorithms for positive linear programming (building on earlier work by Papadimitriou and Yannakakis [57] and Luby and Nisan [44]). In this model, there is a primal agent for each primal variable and a dual agent for each dual variable (or primal constraint). In [4], direct communication is permitted between a dual agent and all of the primal agents appearing in the corresponding constraint; in this model, Bartal et al. [4] give a decentralized algorithm that achieves a $(1 + \varepsilon)$-approximation in a polylogarithmic number of rounds. We note that the results presented here are from work by Moskaoyama et al. [53]. Again, a remark about practicality of this algorithm is in order. Indeed, like most solutions listed above, the algorithm presented here is unlikely to be useful as is in practice. However, it is a proof-of-concept for existence of such distributed solution and variant of it is likely to be useful in practice. Better solutions with practical utility naturally form topic of future research.

8

Conclusions

We considered the question of designing Gossip algorithms motivated by applications to next generation networks such as sensor networks, peer-to-peer networks, mobile networks of vehicles, social networks, etc. These algrotihms are built upon a *gossip* or *rumor* style unreliable, asynchronous information exchange protocol. Due to immense simplicity and wide applicability, this class of algorithms have emerged as a canonical architectural solution for these next generation networks.

We started with the description of Gossip algorithm for information exchange. On this Gossip based information layer, we presented design of the linear dynamics based algorithm as well as the separable function computation algorithms. These algorithms were further utilized to design the network scheduling and the network convex optimization algorithm. Thus, in effect we described a whole 'Gossip based network algorithmic stack' here. An important conclusion is that the performance of Gossip algorithms is strongly dependent on the spectral properties of underlying network graph.

Acknowledgments

The author would like to acknowledge support of NSF projects CNS 0626764, CCF 0728554 and HSD 0729361 while this article was written. He wishes to thank Damon Mosk-Aoyama for various collaborations on the topic of Gossip algorithms that have provided a bulk of the material for this article. Author wishes to thank Jinwoo Shin for various useful discussions. Finally, he also wishes to thank Tauhid Zaman and an anonymous reviewer for suggestions to improve the readability of this article.

Notations and Acronyms

$\mathbf{Z}$, The set of all integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

$\mathbf{N}$, The non-negative integers.

$\mathbf{R}$, $\mathbf{R}_+$ and $\mathbf{R}_{++}$, Real numbers, non-negative real numbers and strictly positive real numbers respectively.

Bold small letters, (e.g., $\mathbf{x}$), a vector $[x_i] \in \mathbf{X}^n$ of real numbers ($\mathbf{X} = \mathbf{R}$), or integers ($\mathbf{X} = \mathbf{Z}$). Usually, vectors will be assumed to be represented in the 'column' form.

$\mathbf{0}$ and $\mathbf{1}$ vector of all zeros and all ones, respectively.

$\mathbf{1}_{\{\cdot\}}$, Indicator function for checking the 'truth' of the condition '$\cdot$', i.e., $\mathbf{1}_{\text{true}} = 1$ and $\mathbf{1}_{\text{false}} = 0$. $\mathbf{1}_{\{\mathbf{x}\}}$, A vector whose ith component is $\mathbf{1}_{\{x_i\}}$.

$\langle \mathbf{x}, \mathbf{y} \rangle$, $\sum_{i=1}^{n} x_i y_i$ (for n-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$).

References

[1] D. J. Aldous, "Some inequalities for reversible Markov chains," *Journal of the London Mathematical Society*, vol. 25, pp. 564–576, 1982.

[2] P. Assouad, "Plongements lipschitziens dans $\mathbf{R}^n$," *Bulletin de la Société Mathématique de France*, vol. 111, no. 4, pp. 429–448, 1983.

[3] O. Ayaso, "Information theoretic approaches to distributed function computation," PhD thesis, Massachusetts Institute of Technology, 2008.

[4] Y. Bartal, J. W. Byers, and D. Raz, "Fast, distributed approximation algorithms for positive linear programming with applications to flow control," *SIAM Journal on Computing*, vol. 33, no. 6, pp. 1261–1279, 2004.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[6] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, December 2005.

[7] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Review*, 2004.

[8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transaction on Networking*, vol. 14, no. SI, pp. 2508–2530, 2006.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[10] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," in *43rd Allerton Conference on Communication Control and Computing*, 2005.

[11] L. Chen, S. H. Low, M. Chang, and J. C. Doyle, "Optimal cross-layer congestion control, routing and scheduling design in ad-hoc wireless networks," in *IEEE INFOCOM*, 2006.

[12] J. Considine, F. Li, G. Kollios, and J. W. Byers, "Approximate aggregation techniques for sensor databases," in *20th IEEE International Conference on Data Engineering*, April 2004.

[13] J. Dai and B. Prabhakar, "The throughput of switches with and without speed-up," in *Proceedings of IEEE Infocom*, pp. 556–564, 2000.

[14] S. Deb, M. Médard, and C. Choute, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE/ACM Transactions on Networking*, vol. 14, 2006.

[15] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*. Springer, Second Edition, 1998.

[16] P. Diaconis, S. Holmes, and R. Neal, "Analysis of a non-reversible Markov chain sampler," *Annals of Applied Probability*, vol. 10, pp. 726–752, 2000.

[17] P. Diaconis and L. Saloff-Coste, "Moderate growth and random walk on finite groups," *Geometric and Functional Analysis*, vol. 4, no. 1, pp. 1–36, 1994.

[18] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient aggregation for sensor networks," in *5th International ACM/IEEE Symposium on Information Processing in Sensor Networks (IPSN '06)*, April 2006.

[19] A. El Gamal, J. Mammen, B. Prabhakar, and D. Shah, "Optimal throughput-delay scaling in wireless networks-part I: The fluid model," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2568–2592, 2006.

[20] L. Elsner, I. Koltracht, and M. Neumann, "On the convergence of asynchronous paracontractions with applications to tomographic reconstruction from incomplete data," *Linear Algebra and Its Applications*, no. 130, pp. 65–82, 1990.

[21] M. Enachescu, A. Goel, R. Govindan, and R. Motwani, "Scale free aggregation in sensor networks," in *International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[22] A. Eryilmaz, A. Ozdaglar, D. Shah, and E. Modiano, "Distributed cross-layer algorithms for the optimal control of multi-hop wireless networks," *IEEE/ACM Transactions on Networking* (accepted to appear), 2009.

[23] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computer and System Science*, vol. 31, no. 2, pp. 182–209, 1985.

[24] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Transactions on Communications*, vol. COM-25, no. l, pp. 73–85, 1977.

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*.

[26] N. Garg and N. E. Young, "On-line end-to-end congestion control," in *IEEE FOCS*, pp. 303–312, 2002.

[27] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE Journal of Select Areas*

Communication High-performance Electronic Switches/Routers for High-speed Internet, vol. 21, no. 4, pp. 546–559, 2003.

[28] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transaction on Information Theory*, vol. 46, no. 2, pp. 388–404, March 2000.

[29] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Transactions on Information Theory*, vol. 34, 1988.

[30] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.

[31] http://www.reuters.com/article/pressrelease/idUS55597+16-Jan-2008+BW 20080116.

[32] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[33] K. Jung and D. Shah, "Low delay scheduling in wireless network," in *IEEE ISIT*, 2007.

[34] K. Jung, D. Shah, and J. Shin, "Minimizing rate of convergence for iterative algorithms," *IEEE Transactions on Information Theory* (accepted to appear), 2009.

[35] A. Kashyap, T. Basar, and R. Srikant, "Quantized consensus," (in press).

[36] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, March 1998.

[37] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, p. 482, Washington, DC, USA: IEEE Computer Society, 2003.

[38] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analaysis," in *Symposium on Theory of Computing*, ACM, 2004.

[39] M. Koubarakis, C. Tryfonopoulos, S. Idreos, and Y. Drougas, "Selective information dissemination in P2P networks: Problems and solutions," *SIGMOD Record*, vol. 32, no. 3, pp. 71–76, 2003.

[40] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *Submitted,* Available Through csl.uiuc.edu/rsrikant, 2006.

[41] X. Lin and N. B. Shroff, "Impact of imperfect scheduling in wireless networks," in *IEEE INFOCOM*, 2005.

[42] N. Linial and A. Wigderson, "Lecture notes on Expander Graphs," http://www.math.ias.edu/∼avi/BOOKS/expanderbookr1.pdf.

[43] L. Lovasz and P. Winkler, "Mixing times," in *Microsurveys in Discrete Probability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, (D. Aldous and J. Propp, eds.), pp. 85–133, AMS, 1998.

[44] M. Luby and N. Nisan, "A parallel approximation algorithm for positive linear programming," in *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pp. 448–457, 1993.

[45] R. Madan, D. Shah, and O. Leveque, "Product multi-commodity flow in wireless networks," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1460–1476, 2008.

[46] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, 2002.

[47] L. Massoulié and M. Vojnovic, "Coupon replication systems," in *ACM SIGMETRICS/Performance*, 2005.

[48] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE Transactions on Networking*, vol. 7, no. 2, pp. 188–201, 1999.

[49] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings of IEEE Infocom*, pp. 296–302, 1996.

[50] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. London: Springer-Verlag, 1993.

[51] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless network via gossiping," in *ACM SIGMETRICS/Performance*, 2006.

[52] R. Montenegro and P. Tetali, "Mathematical aspects of mixing times in Markov chains," *Foundations and Trends in Theoretical Computer Science*, vol. 1, no. 3, pp. 237–354, 2006.

[53] D. Mosk-Aoyama, T. Roughgarden, and D. Shah, "Fully distributed algorithms for convex optimization problems," in *International Symposium on Distributed Computation (DISC)*, 2007.

[54] D. Mosk-Aoyama and D. Shah, "Information dissemination via network coding," in *IEEE ISIT*, 2006.

[55] D. Mosk-Aoyama and D. Shah, "Fast distributed algorithms for computing separable functions," *IEEE Transactions on Information Theory*, vol. 54, no. 7, pp. 2997–3007, 2008.

[56] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *LIDS Report 2755, to appear in IEEE Transactions on Automatic Control*, 2008.

[57] C. Papadimitriou and M. Yannakakis, "Linear programming without the matrix," in *Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, 1993.

[58] M. Penrose, *Random Geometric Graphs. Oxford Studies in Probability*, Oxford: Oxford University Press, 2003.

[59] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *ACM SIGCOMM*, pp. 367–378, 2004.

[60] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: An efficient randomized protocol for contention resolution," in *ACM SIGMETRICS/Performance*, 2009.

[61] O. Reingold, A. Wigderson, and S. Vadhan, "Entropy waves, The zig-zag graph product, and new constant-degree expanders and extractors," *Annals of Mathematics*, 2002.

[62] T. Rockafellar, *Network Flows and Monotropic Optimization*. Wiley-Interscience, (republished by Athena Scientific, 1998), 1984.

[63] K. Savla, F. Bullo, and E. Frazzoli, "On traveling salesperson problems for Dubins' vehicle: stochastic and dynamic environments," in *IEEE CDC-ECC*, pp. 4530–4535, Seville, Spain, December 2005.

[64] D. Shah, "Stable algorithms for input queued switches," in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2001.
[65] D. Shah and D. J. Wischik, "Optimal scheduling algorithm for input queued switch," in *IEEE INFOCOM*, 2006.
[66] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhäuser, 2004.
[67] 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*, vol. 37, pp. 1936–1948, 1992.
[68] L. Trevisan, "Non-approximability results for optimization problems on bounded degree instances," in *ACM STOC*, 2001.
[69] J. Tsitsiklis, "Problems in decentralized decision making and computation," PhD dissertation, Lab. Information and Decision Systems, MIT, Cambridge, MA, 1984.
[70] www.bbc.co.uk/iplayer.