ABSTRACT

Title of thesis: DISTRIBUTED LOAD BALANCING ALGORITHM IN WIRELESS NETWORKS

Alireza Sheikhattar, Master of Science, 2014

Thesis directed by: Dr. Mehdi Kalantari
Department of Electrical and Computer Engineering

As communication networks scale up in size, complexity and demand, effective distribution of the traffic load throughout the network is a matter of great importance. Load balancing will enhance the network throughput and enables us to utilize both communication and energy resources more evenly through an efficient redistribution of traffic load across the network.

This thesis provides an algorithm for balancing the traffic load in a general network setting. Unlike most of state-of-the-art algorithms in load balancing context, the proposed method is fully distributed, eliminating the need to collect information at a central node and thereby improving network reliability. The effective distribution of load is realized through solving a convex optimization problem where the p-norm of network load is minimized subject to network physical constraints. The optimization solution relies on the Alternating Direction Method of Multipliers (ADMM), which is a powerful tool for solving distributed convex optimization problems. A three-step ADMM-based iterative scheme is derived from suitably reformulated form of p-norm problem. The distributed implementation of the proposed

algorithm is further elaborated by introducing a projection step and an initialization setup. The projection step involves an inner-loop iterative scheme to solve linear subproblems. In a distributed setting, each iteration step requires communication among all neighboring nodes. Due to high energy consumption of node-to-node communication, it is most appealing to devise a fast and computationally efficient iterative scheme which can converge to optimal solution within a desired accuracy by using as few iteration steps as possible. A fast convergence iterative scheme is presented which shows superior convergence performance compared to conventional methods. Inspired by fast propagation of waves in physical media, this iterative scheme is derived from partial differential equations for propagation of electrical voltages and currents in a transmission line. To perform these iterations, all nodes should have access to an acceleration parameter which relies on the network topology. The initialization stage is developed in order to overcome the last challenging obstacle toward achieving a fully distributed algorithm.

DISTRIBUTED LOAD BALANCING ALGORITHM IN WIRELESS NETWORKS

by

Alireza Sheikhattar

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2014

Advisory Committee:
Dr. Mehdi Kalantari, Chair/Advisor
Professor Mark Shayman
Professor Richard J. La

Acknowledgments

I would like to take this opportunity to express my gratitude to all the people who have made this thesis possible. First and foremost, I am deeply thankful to my advisor, Dr. Mehdi Kalantari for his supervision, encouragement, and strategic guidance which has greatly shaped this work from the very beginning. His wide knowledge and constructive comments have been a great value for me.

I would also like to thank Professor Richard La and Professor Mark Shayman for agreeing to serve on my thesis committee and generously sharing their valuable time and expertise to enrich this work.

I have been fortunate to have great friends at UMD over these years, who helped me to get through the tough times.

Lastly and most importantly, I owe my deepest thanks to my family - my mother, father and my sisters who have always stood by me in the worst of times and supported me in all manners during my entire academic process. They have lost a lot due to my studies abroad. To them I dedicate this thesis.

This work was partially supported by NSF under grant 0931957.

Table of Contents

List of Tables iv

List of Figures v

List of Abbreviations vii

1 Introduction 1

2 Fast Convergent Iterative Scheme Design 8
- 2.1 Motivation . . . 8
- 2.2 Accelerated Iterations Inspired by Wave Propagation . . . 9
  - 2.2.1 Numerical Analysis of Wave Propagation in Transmission Lines 10
  - 2.2.2 General Scheme . . . 13
- 2.3 Stability and Convergence Rate Analysis . . . 15
  - 2.3.1 Optimal Acceleration Parameter Derivation . . . 17
  - 2.3.2 Special Scenario: Simple Path Graph . . . 21
- 2.4 Numerical Examples . . . 23

3 Distributed Load Balancing Protocol 26
- 3.1 Network Model . . . 26
- 3.2 Problem Formulation . . . 28
- 3.3 ADMM Overview . . . 30
- 3.4 Algorithm Derivation . . . 33
  - 3.4.1 Application of ADMM to p-norm problem . . . 34
  - 3.4.2 Distributed Projection Subproblems . . . 38
  - 3.4.3 Initialization Stage . . . 42

4 Numerical Results 47
- 4.1 Convergence Performance . . . 47
- 4.2 initialization stage . . . 51

Bibliography 54

List of Tables

4.1 Dominant mode estimation within a specific accuracy for the generated network graph. 53

List of Figures

2.1 From top to bottom: 1) A continuous single stub transmission line discretized on indicated points $V_i = V(i\Delta x)$ with spacing Δx, 2) Equivalent circuit model consists of series of elementary components, each containing primary elements R, L, C, 3) Equivalent network graph model as a path containing $N + 1$ nodes with source and sink nodes at both ends. 10

2.2 The circuit interpretation of iterative equations: (a) the i-th node voltage V_i updated using the difference of input I_i and output I_{i+1} currents, (b) the branch current I_{i+1} updated using the difference of two adjacent node voltages V_i and V_{i+1}. 12

2.3 The Kirchhoff's nodal rule for i-th node in general setting as a junction of M branches. 14

2.4 A path network with source node as the first vertex (v_0) and sink node as the last vertex (v_N) and $(N-1)$ relay nodes. 21

2.5 A typical network with 12 nodes, where the first three nodes (v_1, v_2, v_3) are source nodes injecting $(b_1, b_2, b_3) = (1, 2, 1)$ units of traffic respectively, and node v_{12} is the sink node. 23

2.6 The convergence rate performance of the proposed iterative scheme for all node potentials. 24

2.7 Convergence rate comparison of potential levels for path network with $N = 20$ links and source and sink node at both end for two different iterative techniques: 1)Jacobi iterations with averaging policy, 2) proposed accelerated scheme with optimum stability factor ($\xi = \frac{\pi}{40}$). . 25

4.1 A complex network graph of $N = 100$ nodes and $M = 386$ edges randomly generated over a square of side 100 units, with five designated source nodes and a single sink node. The minimum cutset or namely *bottleneck* of network graph has been highlighted in dashed blue lines. 48

4.2 The convergence performance of the load-balanced network flow using the proposed ADMM-based algorithm applied to the generated network in figure 4.1 for the most significant network flows. 49

4.3 The convergence performance of primal residuals r_k showing the amount of violation from the flow conservation. 50

4.4 The convergence performance of dual residuals s_k. 51
4.5 Convergence performance of the inner-loop accelerated iterative scheme for the first four ADMM iteration steps at all nodes in network graph 4.1. 52
4.6 Impulse response of generated network graph at the location of all nodes. 53

List of Abbreviations

ADMM	Alternating Direction Method of Multipliers
WSN	Wireless Sensor Network
WMN	Wireless Mesh Network
MANET	Mobile Ad hoc Network
LTE	Long-Term Evolution
D2D	Device-to-Device
SPOF	Single Point of Failure
PDE	Partial Differential Equation
FDM	Finite Difference Method
MIMO	Multiple-input Multiple-output

Chapter 1: Introduction

A wireless multihop network is a collection of interconnected nodes communicating with each other over a shared wireless medium. Each network node has the capability of both generating information and relaying data belonging to other nodes. Thus, a typical route consists of several hops or intermediate nodes forwarding data to the destination. Information generated by the source nodes should be routed through specific nodes which possess higher energy supply and processing capabilities, termed as the *fusion* or *sink* nodes. One of the main goals of designing routing protocols in multihop wireless networks is to establish an effective traffic pattern from the source nodes to the sinks in the sense that it utilizes network resources in the most efficient way.

The majority of routing protocols in multihop networks use the shortest path or the minimum hop routing, where each source node transmits information via the shortest path to its corresponding destination. This routing policy concentrates the traffic load along certain paths leading to an imbalanced load distribution. As a result, intermediate nodes located on these paths have to deal with heavier traffic loads compared to their peers. These overloaded nodes may form routing bottlenecks which will reduce the networking performance through congestion, while rapidly

consuming energy resources at node level causing connectivity losses in network.

While several multi-path routing methods [1,2] have been proposed, they tend to be extension of the single shortest-path routing paradigm with use of additional alternate paths. Despite what is commonly believed in the networking community, [3] shows that using multiple paths for routing does not necessarily result in a better load balance compared to single-path routing, unless a huge number of paths is used between any source-destination pair of nodes. A load balancing scheme directly embedded in the network routing layer would resolve these problems through an efficient redistribution of traffic load across the network. Load balancing reduces the likelihood of congestion at the bottleneck links and improves the data delivery rate as the traffic load is dispersed uniformly along multiple paths. The balanced distribution of traffic load ensures effective use of links' capacities, and consequently, maximizes the network throughput. Load balancing plays a vital role in wireless sensor networks (WSNs), as it prolongs the expected lifetime of the sensor network by balancing the energy consumption of sensor nodes.

In this thesis, we will study the problem of load balancing in the general context of multihop networks. As defined in [4], the term *load* is generally defined as the quantity of traffic received and transmitted by a node per unit of time on behalf of other nodes in the network. Load balancing is defined as the uniform distribution of communication and processing operations among different entities of network to avoid overloading any one element. A large body of research has been conducted on load balancing in different types of networks, such as wireless mesh networks (WMNs) [5–7], mobile ad hoc networks (MANETs) [8–10], wireless access

networks [11], WiFi networks [12], and satellite networks [13]. Load balancing techniques have even found their way into the cellular networks [14–16] through several offloading techniques. In [14], a new wireless system architecture called *Integrated cellular and ad hoc relaying* (iCAR) systems is presented which can efficiently balance traffic loads between cells by using ad hoc relaying stations. Very recently in Long-Term Evolution Advanced (LTE-A) networks [17], a device-to-device (D2D) communication based technique for load balancing has been proposed which utilizes D2D communication to efficiently offload traffic among multi-tier cells according to their real-time traffic distributions.

Load balancing has received much attention recently by emerging large-scale wireless sensor networks (WSNs). Most of the previous works on load balancing in WSNs deal with the problem of maximizing the network lifetime [1, 18, 19]. In [1], an energy aware routing protocol has been developed for low energy networks with the aim of network survivability. This protocol use probabilistic forwarding to send traffic on different routes providing a simple multipath routing mechanism. In [18], routing has been formulated as a linear programming problem where the network lifetime is maximized constrained to flow conservation. As a result, a shortest cost path routing algorithm has been proposed, whose link cost is a combination of communication energy consumption and the residual energy levels. Several works in the context of WSNs tackle load balancing problem directly by distributing the traffic load across the network [20–23]. In [20], load balancing in dense wireless networks has been studied, where the authors consider a minimax optimization problem and develop lower bounds for the achievable minimal maximal traffic load.

In [21], the authors have proposed a novel cost-based routing protocol for WSNs called Arbutus, which tends to improve data delivery performance by balancing the workload through different routes.

Majority of load balancing techniques in the literature hold the notion of centralized processing in common. Load balancing strategies proposed in the MANETs literature require network topological information to be available at the node level. In the centralized scenario, all information is aggregated at a central or fusion node for processing. Data collection and transmission to a central processing node pose a major drain on communication and energy resources. Furthermore, the central node that performs load balancing introduces a single point of failure (SPOF) reducing the reliability of network. To overcome these shortcomings, it is desirable to replace the notion of centralized processing with decentralized cooperation of network nodes in a distributed scenario. By *distributed* we imply that a central node no longer exists and there is no global network information available at any location. Furthermore, each node only has one-hop access to the network information by exchanging data with neighbor nodes.

To the best of our knowledge, very few algorithms has been offered for load balancing in the distributed context. In [24], two algorithms have been proposed, one partially and one fully distributed, based on the subgradient algorithm to compute an optimal routing flow to maximize the network lifetime. These algorithms achieve the optimal load-balanced traffic pattern with multipath structure. However, the fully distributed algorithm shows slow convergence performance and the partially distributed algorithm requires global communication among all nodes. In

this thesis, we propose a fully distributed algorithm for load balancing in a general network setting. We consider the convergence rate issue as one of the determining factors in our distributed algorithm design. Hence, our proposed algorithm offers the advantage of requiring fewer iteration steps to balance the traffic load. The balanced distribution of load is achieved through solving a convex optimization problem where p-norm of network flow is minimized subject to flow conservation constraint. The p-norm flow optimization will result in a balanced flow assignment to the network links, as network flows with larger absolute values incur higher penalty costs. The parameter "p" serves as the balance factor; as it grows larger, network load will be distributed more evenly throughout the network. In order to develop a distributed load balancing algorithm, a fast iterative scheme is required to solve linear equations problem on the network graph. In the first part of this thesis, we propose a fast convergence iterative scheme inspired by wave propagation in a transmission line, whose accelerated iterations are derived from partial differential equations (PDE) for propagation of voltages and currents in a physical medium. We show that the proposed scheme outperforms the Jacobi method in the sense of convergence rate. Indeed, we show that the iterations required for convergence of our proposed method is reduced to $O(N)$ for a path network of N nodes.

The key contribution of this thesis is to design a distributed scheme to solve the *p-norm* optimization in a network setting. Such distributed scheme follows an iterative procedure where the iterations converge to the optimal solution of the *p-norm* problem. Our solution for this problem relies on the Alternating Direction Method of Multipliers (ADMM), which is a powerful algorithm for solving convex

optimization problems [25]. ADMM is an effective tool suitable for distributed convex optimization problems, originally proposed in 1970's [26, 27], studied and developed in earlier literature with focus on convex programming and variational inequalities [28, 29]. It has received more attention recently by emerging large-scale distributed frameworks, and has found many applications in diverse areas [30–38]. A considerable body of work show the close relevance of ADMM, or equivalence in some cases, to a class of well-known algorithms, such as Douglas-Rachford splitting, and Bregman iterative algorithms [39–41]. Generally, ADMM is the joint process of decomposition-coordination where a large problem is decomposed to small local subproblems and then the solution to the global large problem found by coordination of local subproblems. A comprehensive survey on ADMM and its applications can be found in [42].

ADMM provides a three-step iterative scheme when applied to the suitably reformulated form of p-norm problem. These three iteration steps are as follows: Distributed projection, parallel flow minimization, and dual update. The distributed implementation of these three steps are discussed thoroughly in chapter 3. We present a distributed procedure for the projection step by reducing it into solving system of linear equations. A significant desirable feature pertaining distributed iterative schemes is their fast convergence rate. Hence, the accelerated second-order iterative scheme presented in chapter 2, is employed to solve linear subproblems in an efficient distributed way. To perform these iterations, all nodes should have access to a certain parameter which relies on the network topology. To find this parameter, we use a distributed initialization and network identification, which is based on novel

idea of distributed estimation by modeling network as a multi-input multi-output (MIMO) dynamical system. Using this initialization setup, each node can make an accurate evaluation of the desired parameter only by exchanging information with its neighboring nodes. Simulation results show that our proposed algorithm requires very few (typically, 3-5) ADMM iterations to achieve a load-balanced traffic on bottleneck links, even when used on randomly generated complex network graphs.

Chapter 2: Fast Convergent Iterative Scheme Design

2.1 Motivation

In this chapter, a fast iterative scheme is presented which can solve linear system of equations on a network graph in a distributed fashion. The proposed scheme has several applications in variety of network optimization problems such as load balancing, maximum utilization, optimal node deployment and sink placement. The proposed scheme is originally proposed in the context of potential-based routing [43], where a potential function is used as the basis for routing of information flow from sources to their corresponding destinations. This potential function determines a simple routing policy such that the input flow to each node is divided by amounts proportional to the potential decent to respective neighbor nodes. The potential-based approach is used for load balancing in [44]. In a potential-based routing, it is essential to compute the potential function at the location of nodes. Generally, it is desirable to calculate potential values in a distributed fashion (i.e., nodes compute their potential values through a sequence of distributed iterations with their neighbors). The best existing method for distributed potential calculation makes use of Jacobi iterations [45]. These iterations are well-known for their slow convergence which makes them useless for practical applications.

In this chapter, we first present an iterative scheme inspired by fast transient behavior of voltages and currents in transmission line. The proposed scheme is extended to make it compatible with the general network setting. Based upon the derived iterations, a second order iterative scheme is proposed in network domain. In the sequel, the stability and convergence properties of the proposed scheme is further analyzed for general network topology. The optimal rate of convergence is attained by optimally choosing an acceleration parameter. It has shown that the proposed scheme enhances the convergence rate by one order of magnitude, as it requires $O(N)$ number of iterations compared to $O(N^2)$ for Jacobi iterations. In the last section, convergence performance has been shown using numerical examples and simulations.

2.2 Accelerated Iterations Inspired by Wave Propagation

In this section, we use a numerical approximation scheme to discretize the wave propagation differential equation in transmission lines. Then, we introduce a new iterative method for distributed calculation of potential values in communication networks by drawing analogy between the transient behavior of voltages and currents in transmission line, and potential iterations in a path network. Later in this section, we will explain how the new scheme can be generalized to all network topologies.

2.2.1 Numerical Analysis of Wave Propagation in Transmission Lines

To analyze the transient behavior of voltages and currents, a continuous transmission line is modeled as a sequence of two-port elementary components, each representing an infinitesimally short segment of transmission line shown in Figure 2.1.

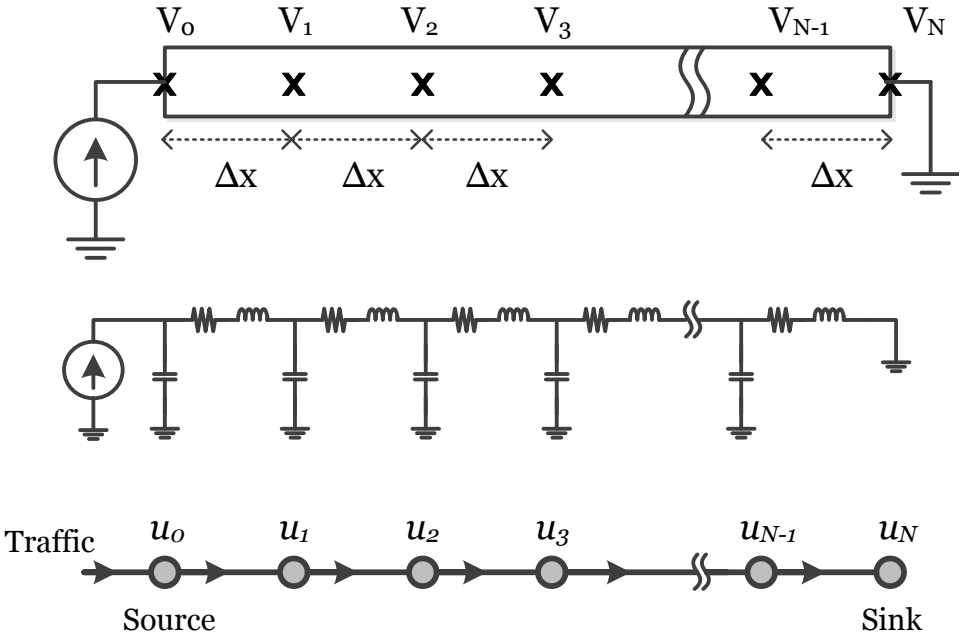

Figure 2.1: From top to bottom: 1) A continuous single stub transmission line discretized on indicated points $V_i = V(i\Delta x)$ with spacing Δx, 2) Equivalent circuit model consists of series of elementary components, each containing primary elements R, L, C, 3) Equivalent network graph model as a path containing $N + 1$ nodes with source and sink nodes at both ends.

A pair of first-order partial differential equations (PDE), known in classical physics as telegrapher's equations, describes wave propagation in the transmission line

$$\begin{aligned} \frac{\partial}{\partial x} I(x,t) &= -C\frac{\partial}{\partial t} V(x,t) - GV(x,t) \\ \frac{\partial}{\partial x} V(x,t) &= -L\frac{\partial}{\partial t} I(x,t) - RI(x,t) \end{aligned} \tag{2.1}$$

where the parameters R, L, C and G are distributed elements of transmission line per unit length known as primary line constants. Inspired by fast transient behavior of voltages and currents, our goal is to model wave propagation as an effective iterative process. An accurate numerical approximation scheme is required to make use of the fast transient behavior. For this purpose, we use the finite difference method (FDM) to discretize telegrapher's equations (2.1) and to convert them into difference equations by sampling in discrete space. We use $V_i^n = V(i\Delta x, n\Delta t)$ and $I_i^n = I(i\Delta x, n\Delta t)$, $i \in \{0, 1, 2, \ldots, N\}$, $n \in \{0, 1, 2, \ldots\}$, to denote the numerical approximation for voltages and currents at time sample n and location i. In other words, voltages and currents are sampled at discrete points with uniform spacing on both dimensions $(\Delta x, \Delta t)$. Replacing the derivatives in equations (2.1) by their finite difference approximations results in the following discrete equations:

$$\begin{aligned} \frac{I_{i+1}^n - I_i^n}{\Delta x} &\approx -C\frac{V_i^{n+1} - V_i^n}{\Delta t} - GV_i^n \\ \frac{V_{i+1}^{n+1} - V_i^{n+1}}{\Delta x} &\approx -L\frac{I_{i+1}^{n+1} - I_{i+1}^n}{\Delta t} - RI_{i+1}^{n+1} \end{aligned} \tag{2.2}$$

Given the initial conditions for voltages and currents along the line, they can both be obtained for every time index n. In order to achieve a time balance in (2.2), the loss terms GV_i^n and RI_{i+1}^{n+1} can be replaced by their time-averaged version, $(V_i^n + V_i^{n+1})/2$ and $(I_{i+1}^n + I_{i+1}^{n+1})/2$, respectively. This also helps get more accurate approximations and reduced discretization error. By applying this change and re-arranging the terms in (2.2), we find the following iterative system for voltages and

currents:

$$V_i^{n+1} = \frac{1-\frac{G\Delta t}{2C}}{1+\frac{G\Delta t}{2C}} V_i^n - \frac{\frac{\Delta t}{C}}{1+\frac{G\Delta t}{2C}} (I_{i+1}^n - I_i^n) \tag{2.3}$$

$$I_{i+1}^{n+1} = \frac{1-\frac{R\Delta t}{2L}}{1+\frac{R\Delta t}{2L}} I_{i+1}^n - \frac{\frac{\Delta t}{L}}{1+\frac{R\Delta t}{2L}} (V_{i+1}^{n+1} - V_i^{n+1}) \tag{2.4}$$

where the first equation updates the voltage at each node and the second equation updates the current in each branch, as illustrated in Figure 2.2.

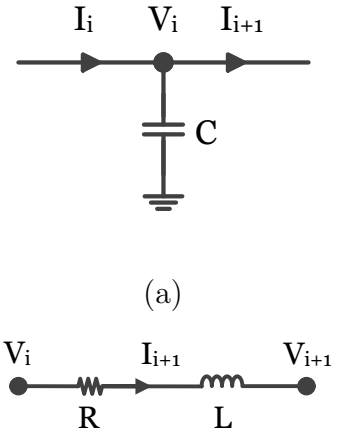

Figure 2.2: The circuit interpretation of iterative equations: (a) the i-th node voltage V_i updated using the difference of input I_i and output I_{i+1} currents, (b) the branch current I_{i+1} updated using the difference of two adjacent node voltages V_i and V_{i+1}.

The steady state behavior of voltages and currents in single stub transmission line resembles the final solution for network parameters in a path network as shown in Figure 2.1, where voltages are analogous to node potentials and so are currents to information flows. Furthermore, the information source and sink nodes are equivalent to current sources and short circuits to ground, respectively. In order to develop

a new scheme compatible with our desired network behavior, we assume G is zero in our scheme (note that a nonzero G implies leaking information at a node, which violates flow conservation law). However, it is desirable to have a nonzero R, because otherwise, we will have a lossless transmission line, which causes instability. This circuit model with zero G and nonzero R leads to steady state response consistent with our desired network behavior, as we will further discuss shortly.

2.2.2 General Scheme

So far, we have used the analogy between a discretized single stub transmission line and a path network. In the next step, we present a general scheme applicable to all possible network topologies. Using the transmission line model representation for a network, nodes and communication links are modeled as capacitors connected to ground and RL-series branches, respectively. Unlike the path network where each relay node connects only two communication links, a node in general network topology can be at junction of M branches. In both cases, there is only one link between two neighboring nodes. Therefore, the iterative equation (2.4) for updating currents between two adjacent nodes remains the same as before, while the paired equation (2.3) for updating voltages at each node requires modification.

For the general case, we derive this iterative equation directly from the Kirchhoff's Current Law (nodal rule) as shown in Figure 2.3. The updating equation for voltage V_i can be represented in term of the input current to the node's capacitor,

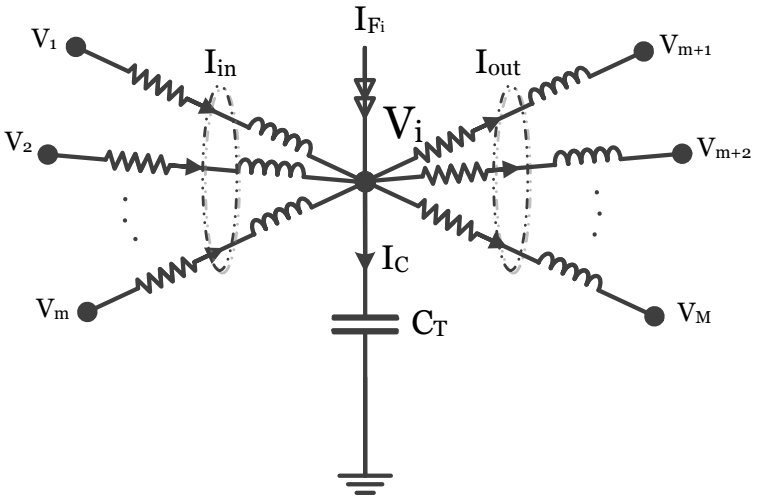

Figure 2.3: The Kirchhoff's nodal rule for i-th node in general setting as a junction of M branches.

I_{C_i}, as below:

$$V_i^{n+1} = V_i^n + \frac{\Delta t}{C_T} I_{C_i}^n \tag{2.5}$$

where the product term $I_C^n \Delta t$ can be interpreted as the amount of new electric charge ΔQ^n deposited in node's capacitor, and if divided by total capacitance C_T, the voltage difference ΔV^n is obtained. Moreover, according to Figure 2.3, I_{C_i} can be calculated as:

$$I_{C_i}^n = \sum_{j=1}^{m} I_{j,i}^n - \sum_{j=m+1}^{M} I_{i,j}^n + I_{F_i}^n \tag{2.6}$$

where the $I_{F_i}^n$ is the injected current at node i, which has a nonzero value for source nodes. The input and output currents are indexed by $j = 1, \ldots, m$ and $j = m+1, \ldots, M$, respectively, where $I_{i,j}$ is the current flowing from node i to node j. The total capacitance at each node C_T is proportional to the node degree M, which is consistent with the parallel connection of M capacitors.

After designing the iterative scheme for general network topology, we still

need to transform it to the network domain with equivalences in mind. In the next section, we will analyze the stability and convergence rate of the proposed scheme to desired solution.

2.3 Stability and Convergence Rate Analysis

In this section, we will analyze the stability and convergence properties of the proposed iterative scheme for general network topology. The convergence rate is enhanced by choosing optimal iteration parameters derived from the spectral analysis. We will prove the superior convergence performance of the proposed accelerated scheme compared to classical Jacobi iterations for the simple path graph.

Based on the aforementioned equivalences between network and circuit models, the proposed iterative scheme can be rewritten in network components as follows

$$u_i^{n+1} = u_i^n - \sigma_i \Big(\sum_{i \to k} f_{i,k}^n - \sum_{j \to i} f_{j,i}^n - f_{b_i}^n \Big) \tag{2.7}$$

$$f_{i,j}^{n+1} = \alpha f_{i,j}^n - \beta (u_j^{n+1} - u_i^{n+1}) \tag{2.8}$$

where $u_i^n, i \in \mathcal{V}$ is the potential of node i, and the flow from node i to node j is denoted by $f_{i,j}^n$, both at time iteration n. The first equation (2.7) represents the nodal equation (2.5) in network domain, where $\sigma_i = \sigma / n_i$ is the network factor for i-th node which is inversely proportional to node degree n_i. The notations $(i \to k)$ and $(j \to i)$ are used to denote the neighbor nodes with output flow from i-th node and input flow to i-th node, respectively. The paired equation (2.8) is the transformed version of currents update equation in (2.4). Similar to partial differential equations, these coupled difference equations require boundary and initial conditions to lead

to a unique solution. The boundary conditions for the general case is determined by the location of sources and sinks in the network graph; with the i-th node as a source or a sink, the boundary condition appears in the form of $f_{b_i}^n = r_i$ or $u_i^n = 0$ for all times n, respectively, where r_i is the generated traffic rate at node i. In the sequel, we will show how choosing network parameters (α, β, σ) ensures the stability condition and accelerates the convergence rate.

The pair of equations (2.7) and (2.8) can be combined into a single equation for node potentials.

$$u_i^{n+1} = u_i^n - \sigma_i \Big(\sum_{i \to k} \Big[\alpha f_{i,k}^{n-1} - \beta (u_k^n - u_i^n) \Big] - \sum_{j \to i} \Big[\alpha f_{j,i}^{n-1} - \beta (u_i^n - u_j^n) \Big] \Big) \quad (2.9)$$

By two consecutive substitutions of (2.8) into (2.7) and rearranging terms, we find the following single iterative equation performed at each node

$$u_i^{n+1} = (\alpha + 1) u_i^n - \alpha u_i^{n-1} + \beta \sigma_i (\sum_{j \sim i} u_j^n - n_i u_i^n) + c_i^n \quad (2.10)$$

where notation $(j \sim i)$ is used to denote the neighborhood of node j and node i, and the bias term $c_i^n = \sigma(1 - \alpha) r_i^n / n_i$. According to the normalized Laplacian operator matrix $\Delta = D^{-1} L$ defined as $[\Delta u]_i := u_i - \frac{1}{n_i} \sum_{j \sim i} u_j^n$ in [46], this linear iterative equation can be rewritten in matrix format as follows

$$u^{(n+1)} = 2Au^{(n)} - \alpha u^{(n-1)} + b^{(n)} \quad (2.11)$$

where

$$2A = \delta I + \gamma S, \tag{2.12}$$

$$S = I - \Delta, \tag{2.13}$$

$$\delta = (\alpha + 1) - \beta\sigma, \tag{2.14}$$

$$\gamma = \beta\sigma \tag{2.15}$$

where $u^{(n)} = [u_1^n, u_2^n, \ldots, u_N^n]^T$ is the potential vector of size N at time n containing node potentials and $b^{(n)} = (1-\alpha)D^{-1}r^n$ is the bias vector containing contributions from boundary conditions. The spectrum of normalized Laplacian matrix of graphs is extensively studied in spectral graph theory [47]. In the next section, we will show the derivation of optimum factors $(\alpha_{opt}, \delta_{opt}, \gamma_{opt})$ for the general network graph scenario based on the spectral analysis of iteration matrices.

2.3.1 Optimal Acceleration Parameter Derivation

By carefully choosing the network parameters, an accelerated rate of convergence would be achieved. The derived iteration (2.11) describes a second order iterative scheme, since computing the next state $u^{(n+1)}$ requires the value of the preceding two states, $u^{(n)}$ and $u^{(n-1)}$. To analyze the stability and convergence rate, the second order iterative scheme (2.11) can be reduced to first order matrix form as follows,

$$\widehat{u}^{(n+1)} = M\widehat{u}^{(n)} \tag{2.16}$$

where $\widehat{u}^{(n+1)}$ is a column vector of $2N$ elements and M is matrix of size $2N \times 2N$ defined as,

$$M = \begin{pmatrix} 2A & -\alpha I_N \\ I_N & 0_N \end{pmatrix}, \quad \widehat{u}^{(n+1)} = \begin{pmatrix} u^{(n+1)} \\ u^{(n)} \end{pmatrix} \tag{2.17}$$

where I_N and 0_N are identity matrix and square zero matrix of size N. Since each iteration step in (2.16) involves a matrix multiplication, the stability condition is satisfied when all eigenvalues of M are within the unit circle in the complex plane. Therefore, it is required that $|\mu_1| \leq 1$, where $\mu_1 = \rho(M)$ is the spectral radius of M, which has a key role in both stability and convergence rate of iterative scheme. In order to achieve a fast convergent scheme, the absolute value $|\mu_1|$ should be minimized, while satisfying the stability conditions. In other words, the spectral gap, $d = 1 - |\mu_1|$ should be maximized. The spectral radius $|\mu_1|$ relies on α and the spectrum of matrix A. Assume A is a diagonalizable matrix and can be factorized by eigenvalue decomposition to $A = Q\Lambda Q^{-1}$, where Q is square matrix with eigenvectors as columns and Λ is the diagonal matrix with real eigenvalues on the diagonal, $\Lambda_{ii} = \lambda_i$. By this decomposition, matrix M can be factorized to,

$$M = \begin{pmatrix} Q & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} 2\Lambda & -\alpha I_N \\ I_N & 0_N \end{pmatrix} \begin{pmatrix} Q^{-1} & 0 \\ 0 & Q^{-1} \end{pmatrix} \tag{2.18}$$

Let $\tilde{M}$ be defined as,

$$\tilde{M} = \begin{pmatrix} 2\Lambda & -\alpha I_N \\ I_N & 0_N \end{pmatrix} \tag{2.19}$$

Since $M \sim \tilde{M}$ are similar matrices [1], they share the same set of eigenvalues. Moreover, $\tilde{M}$ can be decomposed to 2×2 matrices for each eigenvalue λ_i as below,

$$\begin{pmatrix} 2\lambda_i & -\alpha \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \mu_i \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.20}$$

where each eigenvalue μ_i can be simply found as root of the characteristic polynomial,

$$\mu_i^2 - 2\lambda_i \mu_i + \alpha = 0 \tag{2.21}$$

Note that each λ_i is mapped to a pair of eigenvalues $\mu_i^{\pm} = \lambda_i \pm \sqrt{\lambda_i^2 - \alpha}$. To achieve optimum convergence rate, the parameter α should be chosen such that $|\mu_1|$ is minimized. The spectral radius of A is denoted as λ_1. For the case when $\lambda_1^2 > \alpha$, there exists $|\mu_1| > |\lambda_1|$. In the other case, when $\lambda_1^2 \leq \alpha$, it will lead to complex conjugate pair, $\mu_1^{\pm} = \lambda_1 \pm j\sqrt{\alpha - \lambda_i^2}$ where $|\mu_1^{\pm}| = \sqrt{\alpha}$. Hence, the lower bound for $|\mu_1|$ is achieved by choosing α to be equal to

$$\alpha_{opt} = \lambda_1^2 \tag{2.22}$$

By choosing the above α_{opt}, all eigenvalues on the spectrum λ_i will be mapped into complex conjugates $\mu_i^{\pm}$ on a circle of radius $\sqrt{\alpha} = |\lambda_1|$.

To further analyze stability, let us define the *stability factor* parameter, ξ, as below:

$$\xi = \frac{1-\alpha}{1+\alpha} \tag{2.23}$$

where this transformation leads to the same inverse form, $\alpha = \frac{1-\xi}{1+\xi}$. Now we investigate how changing the parameters (α, β, σ) will affect the spectrum of matrix A.

[1] Matrix B is similar to matrix A denoted as $A \sim B$, if there is an invertible matrix P such that $B = P^{-1}AP$, or in other words if they represent the same transformation up to a change of basis.

From the spectral perspective, the coefficient matrix in (2.12) is structured on the basis of the *translation factor* δ in (2.14) and the *scale factor* γ in (2.15). These factors (δ, γ) translate and scale the spectrum domain of matrix S and affect all its eigenvalues in the following way

$$2\lambda_i = \delta + \gamma\nu_i \tag{2.24}$$

where $\{\nu_i\}_{i=1}^N$ denotes the spectrum of matrix S. The necessary condition to obtain stability is to satisfy $|\lambda_1| \leq 1$. Furthermore, the fastest convergence rate would be achieved through minimizing the $|\lambda_1|$ magnitude, which consequently leads to minimal spectral radius $|\mu_1| = \sqrt{\alpha_{opt}}$. Since the spectrum of $S = I - \Delta$ is normalized $\{\nu_i\}_{i=1}^N \in [-1, +1]$, the optimal spectrum domain will be achieved when the spectrum is centered around the origin, $\delta = 0$, which yields the optimal scaling factor

$$\gamma_{opt} = \alpha + 1 \tag{2.25}$$

Hence, the second order iterative scheme takes the following form in the optimal convergence rate scenario by choosing the optimal factors $(\delta_{opt}, \gamma_{opt}) = (0, \alpha + 1)$

$$u^{(n+1)} = (\alpha + 1)Su^{(n)} - \alpha u^{(n-1)} + b^{(n)} \tag{2.26}$$

It is easy to observe that the matrix equality $2A = (\alpha+1)S$ will result in the spectral equality

$$\lambda_1 = \frac{\alpha + 1}{2}\nu_1 \tag{2.27}$$

where the spectral radius of matrix S is denoted as ν_1. Substituting this result in

(2.22), we will have

$$\nu_1^2 = \frac{4\alpha_{opt}}{(\alpha_{opt}+1)^2} \tag{2.28}$$

Based on definition (2.23), the optimal stability factor can be computed for the general network graph as follows

$$\xi_{opt}^2 = 1 - \nu_1^2 \tag{2.29}$$

Hence, the optimum parameter α_{opt} which attains optimal convergence performance is equal to,

$$\alpha_{opt} = \frac{1-\xi_{opt}}{1+\xi_{opt}} = \frac{1-\sqrt{1-\nu_1^2}}{1+\sqrt{1-\nu_1^2}} \tag{2.30}$$

2.3.2 Special Scenario: Simple Path Graph

In this section, we choose the simple path graph as our network configuration. As shown in figure 2.4, one unit of information is generated at the first node, going through $N-1$ relay nodes to route through the sink node at the other end. The boundary conditions are determined as $f_0^n = 1$ and $u_N^n = 0$ for all n. For initial condition, we assume $u_i^0 = 0$ for all i and $f_i^0 = 0$ for all $i \neq 0$. Applying the flow injection $f_0^n = 1$ to source node, all potentials and flows take values iteratively until finally reaching a steady state value.

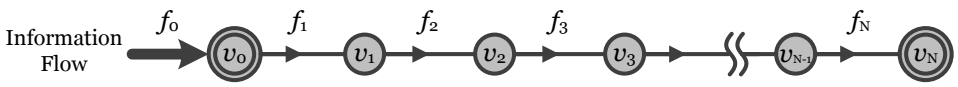

Figure 2.4: A path network with source node as the first vertex (v_0) and sink node as the last vertex (v_N) and $(N-1)$ relay nodes.

The spectrum of matrix S for the path graph has the following closed form:

$$\nu_i = \cos\left(\frac{2i\pi}{2N+1}\right), \quad 1 \leq i \leq N \tag{2.31}$$

where the stability condition is obviously satisfied as we have $|\nu_i| \leq 1$, and the spectral radius is equal to $|\nu_1| = \cos\left(\frac{\pi}{2N+1}\right)$. Iterative schemes converge to the optimal solution at a rate governed by the dominant mode of the iteration matrix. To analyze the convergence rate of Jacobi iterations, we compute the spectral gap of iteration matrix S as follow

$$d = 1 - |\nu_1| = 2\sin^2\left(\frac{\pi}{4N+2}\right) \tag{2.32}$$

where for large N, it can be approximated by $d \approx \frac{\pi^2}{8N^2}$. Hence, Jacobi iterations require order of $O(N^2)$ iterations to converge to the optimal solution. However, in the case of accelerated iterations, using the general form of optimal stability factor in (2.29), we find the following for the path graph

$$\xi_{opt} = \sin\left(\frac{\pi}{2N+1}\right) \tag{2.33}$$

If the length of the path, N, is large enough, we will have the approximation $\xi_{opt} \approx \frac{\pi}{2N}$. By substituting this approximation into $d_{max} = 1 - \sqrt{\alpha_{opt}}$, we find the following new approximation for maximum gap as a function of N:

$$d_{max} \approxeq \frac{\pi}{2N+\pi} \tag{2.34}$$

Hence, it can be seen that the rate of convergence for our proposed method is reduced to $O(N)$ iterations, compared to the Jacobi method whose convergence happens by $O(N^2)$ iterations. Therefore, the number of iterations required to converge to the

optimal solution within desired accuracy decreases significantly, especially when the network size grows.

2.4 Numerical Examples

In this section, we will show the convergence performance of the proposed fast convergent scheme and compare it to the well-known Jacobi iterations. Consider

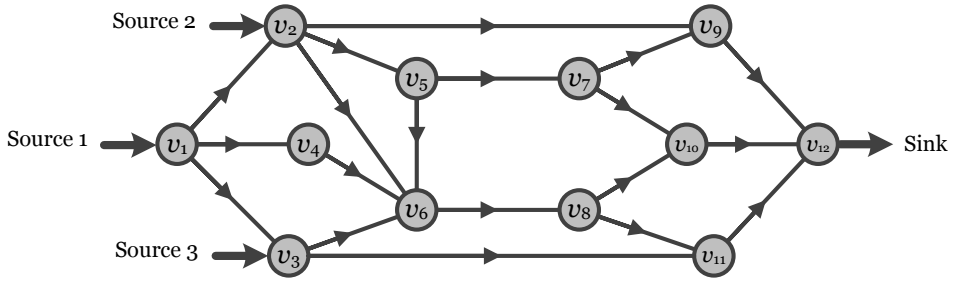

Figure 2.5: A typical network with 12 nodes, where the first three nodes (v_1, v_2, v_3) are source nodes injecting $(b_1, b_2, b_3) = (1, 2, 1)$ units of traffic respectively, and node v_{12} is the sink node.

a typical network in Figure 2.5, in order to evaluate the convergence rate of our proposed method. This network consists of 12 nodes with three nodes (v_1, v_2, v_3) as source nodes injecting $(1, 2, 1)$ units of flow, respectively, and v_{12} as sink node gathering all flow from network. In this example, the link capacity is assumed to be the same and equal to one for all links. Figure 2.6 shows performance of the proposed method in converging to final solution for all node potentials. For the first node potential v_1, our proposed method requires only 17 iterations to converge with error tolerance 1% to final solution. Figure 2.7 depicts the convergence rate of our proposed method for the path network with $N = 20$ compared to the Jacobi method

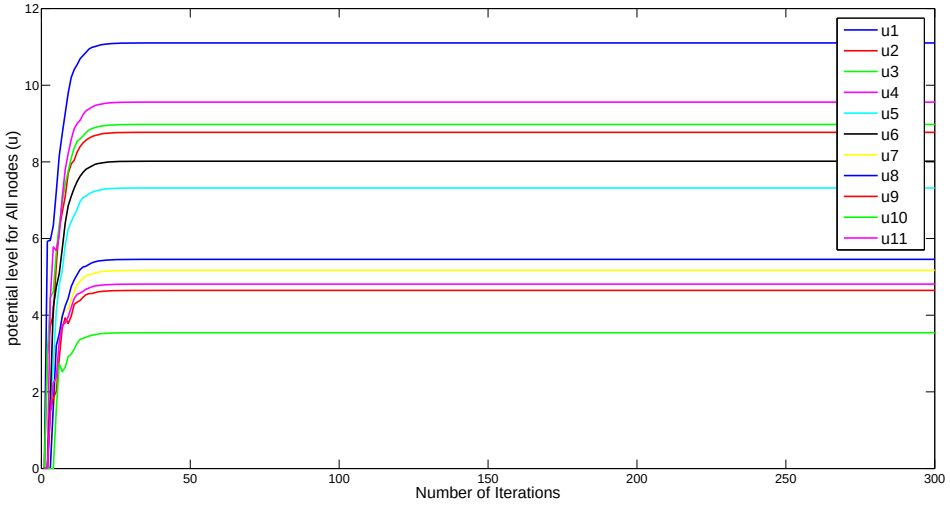

Figure 2.6: The convergence rate performance of the proposed iterative scheme for all node potentials.

based on averaging. As it can be seen, our proposed method requires considerably smaller number of iterations required compared to the Jacobi method. In particular, as the numerical results show, our proposed method requires 70 iterations to converge compared to 1200 iterations required for Jacobi method.

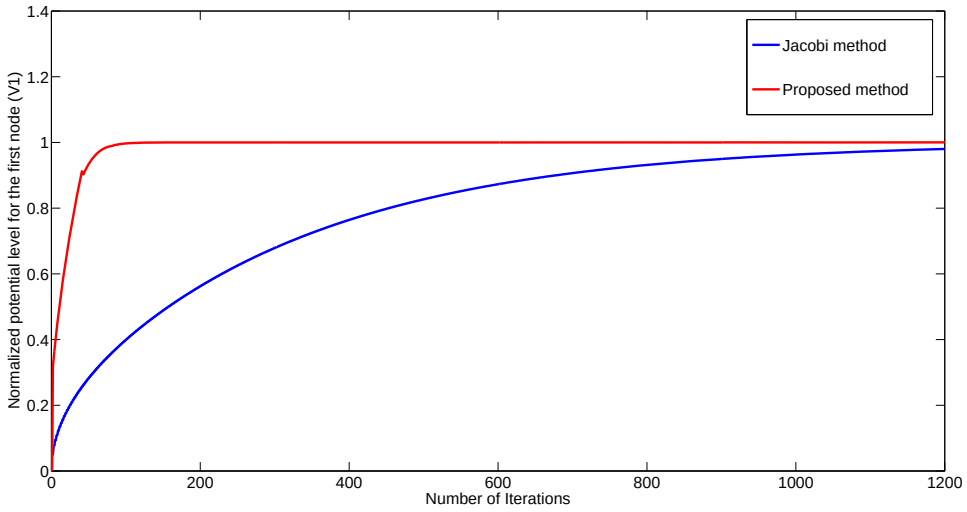

Figure 2.7: Convergence rate comparison of potential levels for path network with $N = 20$ links and source and sink node at both end for two different iterative techniques: 1)Jacobi iterations with averaging policy, 2) proposed accelerated scheme with optimum stability factor ($\xi = \frac{\pi}{40}$).

Chapter 3: Distributed Load Balancing Protocol

3.1 Network Model

We consider a network model based on a connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{1, 2, \ldots, N+1\}$ as the set of vertices and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ as the set of edges. In this model, network nodes and communication links are represented by vertices and edges, respectively. Each edge is denoted by $(i, j) \in \mathcal{E}$, with $i < j$, means that nodes i and j are neighbors which can exchange information directly. The set of one-hop neighbors of node i is denoted by $\mathcal{N}_i$; the cardinality of this set, $d_i = |\mathcal{N}_i|$, defined as the degree of node i. Assume there is M links available in the network graph. We assume that the network topology has no change or changes slowly giving enough time to optimally balance the traffic.

Let f be the flow vector containing network flows $f_m, m = 1, 2, \ldots, M$ streaming on the links. The sign of flow f_m corresponding to the link (i, j) determines the direction of flow from node i to j, with $f_m < 0$ meaning that flow is streaming in the opposite direction. The source vector $b = [b_1, b_2, \ldots, b_N]$ contains information rates generated at each node. The sign of component b_i determines the role of node in the network with positive sign as the source node, negative sign as the sink node and zero-valued for intermediate relay nodes. It is obvious that the total generated

flow would be routed through the sink node, $\sum_{i\in\mathcal{V}} b_i = 0$.

The basic feasibility condition for our network flow model is the flow conservation law. This implies that the sum of input flows to a node, including the generated flow at that node, is equal to zero. The flow conservation can be written in the compact form, $\tilde{K}f = \tilde{b}$, where matrix $\tilde{K}$ is the incidence matrix of size $(N+1)\times M$ for network graph. This matrix contains all the information about the node-edge relationship and show how node i is related to edge m by the $\tilde{K}_{im}$ element of matrix

$$\tilde{K}_{im} = \begin{cases} +1 & \text{if edge m leaves node i} \\ -1 & \text{if edge m enters node i} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

It is known that the rank of the incidence matrix $\tilde{K}_{(N+1)\times M}$ for a typical connected graph is equal to N [48]. Based on rank-nullity theorem, the nullity of $\tilde{K}$ is $M-N$, which is equal to the number of independent meshes in graph. Therefore, the set of linear equations for flow conservation law is linearly dependent. In order to obtain an independent set of equations, one node is specified as the reference node and the corresponding equation is eliminated to achieve

$$Kf = b \tag{3.2}$$

where $K_{N\times M}$ is the reduced incidence matrix of full-row rank. The reduced source vector b of size N specifies the generated flow rates at all nodes excluding the reference node. We assume that b is feasible in the sense that there exists at least one flow vector $f \in \mathcal{F}(b)$ satisfying the flow conservation and channel capacity

constraints. It means that the total generated flow at any arbitrary set of nodes should not exceed the total capacity constraint of their corresponding minimum cutset to the sink.

3.2 Problem Formulation

Our main goal is to devise a fully distributed routing protocol to balance the traffic load in a general network setting, in particular on the bottleneck links where congestion is more likely to occur. A load balancing scheme directly embedded at the network layer would improve the information delivery rate through an efficient redistribution of traffic load across the network. It would also alleviate the congestion problem at hot spots.

The balanced load distribution is achieved through solving a convex optimization problem where *p-norm* of network flow is minimized as cost function constrained to flow conservation law. Thus, the main problem can be formulated as an equality-constrained convex optimization problem as follows

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & J_p(f) = \|f\|_p \\ \text{subject to} \quad & Kf = b \end{aligned} \tag{3.3}$$

where the p-norm, $p \in [1, \infty)$, of vector $f \in \mathbb{R}^n$ can be defined as

$$\|f\|_p = \left(\sum_{m=1}^{M} (f_m)^p \right)^{\frac{1}{p}} \tag{3.4}$$

where f_m is the m-th component of f. In the p-norm problem, "p" determines the balance level of load distribution. The case $p = 1$ is equivalent to the shortest path routing where no load balancing occurs. By increasing the parameter p, the cost

function imposes higher penalties on network flows' absolute values and tries to redistribute flows more evenly throughout the network. In this case, each node distribute the incoming traffic (including the generated traffic) through multiple paths instead of forwarding to the node in the shortest path direction. The optimization solution will tend to minimax problem solution for the limiting case when $p \to \infty$. In most practical scenarios, the solution for the p-norm problem with sufficiently large p will be a good approximation of the maximally balanced flow.

Several convex optimization techniques have been proposed to solve (3.3) in a centralized way, which requires all information to be collected at a central node. This centralization makes these methods useless in a practical setting. In [45], a method based on Sequential Quadratic Programming (SQP) is proposed to optimize the p-norm of network flow. The SQP method solves the main problem (3.3) through solving subproblems of constrained quadratic programming. Each subproblem reduces to linear system of equations solved iteratively using Jacobi iterations, which suffers from slow convergence rate. Large number of iterations is required to achieve the optimal solution within acceptable accuracy. Another serious drawback is the instability caused by the singularity of weighted Laplacian matrices in the case when some flow iterates approach zero. We will come up with a distributed algorithm which overcomes these fundamental problems. In the sequel, we will first provide a brief overview of ADMM algorithm. Then, we will propose a distributed method based on ADMM iterations to solve the p-norm optimization problem.

3.3 ADMM Overview

Alternating Direction Method of Multipliers (ADMM) is an iterative approach to solve convex optimization problems. ADMM combines two methods of dual ascent and the method of multipliers [42]. Hence, it tends to gather the benefits of both methods together: the decomposability and decentralized nature of dual gradient ascent and improved convergence properties of the method of multipliers. In the standard ADMM form of problem, both the objective function and constraints are assumed to be separable across the splitting of the optimization variable x into two blocks of variables x_1 and x_2

$$\begin{aligned} &\text{minimize} && f_1(x_1) + f_2(x_2) \\ &\text{subject to} && A_1 x_1 + A_2 x_2 = c \end{aligned} \tag{3.5}$$

where f_1, f_2 are closed convex functions and A_1, A_2 are full-column rank matrices with the same number of rows. A wide range of important optimization problems can be reformulated in this form (3.5). In the extended version of ADMM, this form can be generalized to the case of multi-block convex optimization problem with objective function separable on $n > 2$ blocks of variables $\{x_i\}_{i=1}^{n}$.

The ADMM algorithm consists of sequential Gauss-Seidel based minimization steps on primal variables, and single dual variable update. In each minimization step, the augmented Lagrangian is minimized with respect to a single primal variable to update it to the next iteration. The set of ADMM iterations for (3.5) can be

expressed in the following form

$$\begin{aligned} x_1^{k+1} &= \underset{x_1}{\operatorname{argmin}} \quad L_\rho(x_1, x_2^k, y^k) \\ x_2^{k+1} &= \underset{x_2}{\operatorname{argmin}} \quad L_\rho(x_1^{k+1}, x_2, y^k) \\ y^{k+1} &= y^k + \rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} - c) \end{aligned} \tag{3.6}$$

where x_1^k, x_2^k and y^k are the k-th iteration updates of primal variables x_1, x_2 and dual variable y; additionally, ρ is the augmented Lagrangian parameter called the penalty factor for the augmented Lagrangian in the form

$$L_\rho(x_1, x_2, y) = f_1(x_1) + f_2(x_2) + y^T(A_1 x_1 + A_2 x_2 - c) + (\rho/2)\|A_1 x_1 + A_2 x_2 - c\|_2^2 \tag{3.7}$$

This Formulation for ADMM can be eased to scaled form [42], by scaling the dual variable $u = (1/\rho)y$, and combining linear and quadratic terms as follows

$$\begin{aligned} x_1^{k+1} &= \underset{x_1}{\operatorname{argmin}} \quad \left(f_1(x_1) + \frac{\rho}{2}\|A_1 x_1 + A_2 x_2^k - c + u^k\|_2^2\right) \\ x_2^{k+1} &= \underset{x_2}{\operatorname{argmin}} \quad \left(f_2(x_2) + \frac{\rho}{2}\|A_1 x_1^{k+1} + A_2 x_2 - c + u^k\|_2^2\right) \\ u^{k+1} &= u^k + A_1 x_1^{k+1} + A_2 x_2^{k+1} - c \end{aligned} \tag{3.8}$$

The convergence of standard ADMM is assured under two mild assumptions on functions and Lagrangian [42]. Functions are assumed to be closed[1] , proper[2] and convex leading to solvable primal updates. A wide range of functions including non-differentiable and unlimited range functions (e.g., indicator function) are covered

[1] A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *closed* if, the epigraph of f, $epif = \{(x,t) \in \mathbb{R}^n | x \in domf, f(x) \leq t\}$ is a closed set.

[2] A convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is *proper* if $f(x) < +\infty$ for at least one x and $f(x) > -\infty$ for all x.

under this general assumption. The unaugmented Lagrangian L_0 is assumed to have a saddle point.

While there is a body of research literature on convergence of ADMM [27, 49], studies on its convergence rate have been established recently for several different scenarios. In [50], the sublinear convergence rate of $O(k)$ for the Jacobi version of ADMM and $O(k^2)$ for the accelerated version has been shown under assumption of smooth objective functions with Lipschitz continuous gradients. The same rates have been proved in [51] for Gauss-Seidel version of ADMM when there is only one of the two objective functions required to be smooth with Lipschitz continuous gradient. [52] proves the convergence rate $O(k^2)$ of the dual objective value for modified version of ADMM and requires functions to be strongly convex and both subproblems to be solved exactly. Stronger result in [53] proves the linear rate of convergence $O(c^k)$ with $c > 1$, for several different scenarios of ADMM when at least one of the two objective functions is strictly convex and has Lipschitz continuous gradient.

In general, standard ADMM algorithm shows a good performance for modest accuracy, practically sufficient in most applications. Some variants and extensions of this algorithm show improved convergence performance compared to the standard version [52]. In most practical network load balancing scenarios, traffic patterns in which the load on bottleneck links is balanced within moderate accuracy (e.g. $\pm 5\%$) of the optimally balanced solution gives an excellent performance.

We will discuss a few variations of ADMM which would not violate our aim of rendering distributed iterations, in the meanwhile show good improvement in

convergence. One way to obtain several variations is to perform updates in different orders or even multiple times. Since the problem is symmetric with respect to the primal variables, their order could be reversed. The variant of algorithm with symmetric update order, where an extra dual update is performed between minimization steps is equivalent to *Peaceman-Rachford splitting* [25]. This variant bring the symmetry to the standard ADMM which will enhance the overall convergence performance. Another simple variation is the *over-relaxation*, where the primal iterate is substituted by relaxed version with parameter $\alpha \in (0, 1)$ in iteration updates. In [54], it has been shown that convergence can improve by choosing $\alpha \in (1.5, 1.8)$. Another variation, called *inexact minimization* or *early termination*, can be helpful in the case where at least one minimization step is carried out in an iterative way. Due to the resilience property of ADMM, the convergence still persists when iterative minimization steps are solved approximately in early ADMM iterations and more accurately for further iterations, provided that certain suboptimality conditions satisfied [40].

3.4 Algorithm Derivation

In this section, we will present a reformulation of our optimization problem and show how application of ADMM would yield distributed, fast, and numerically stable iterations.

3.4.1 Application of ADMM to p-norm problem

We will apply ADMM algorithm to the p-norm optimization problem in (3.3). Among all possible reformulations for ADMM, we choose the one which best match our problem formulation and our aim of distributed implementation. We choose the ADMM formulation proposed for constrained convex optimization in [42]. The equality-constrained convex optimization problem (3.3) can be reformulated in ADMM form as

$$\begin{aligned} &\text{minimize} \quad I_{\mathcal{C}}(x) + \|z\|_p^p \\ &\text{subject to} \quad x - z = 0 \end{aligned} \tag{3.9}$$

where $I_{\mathcal{C}}(x)$ is the indicator function of constraint set $\mathcal{C} = \{x \in \mathbb{R}^M | Kx = b\}$, which is equal to 0 when constraint is satisfied $x \in \mathcal{C}$, and takes value $+\infty$ if constraint is violated $x \notin \mathcal{C}$. By this formulation, the constraint appears in the objective term as the indicator function on the duplicate variable x enforcing the optimal solution to satisfy the constraint $x \in \mathcal{C}$, while minimizing the p-norm cost function.

For ease of formulation, ADMM algorithm for our reformulated problem (3.9) can be expressed in scaled form as follows

$$x^{k+1} = \underset{x}{\operatorname{argmin}} \Big(I_{\mathcal{C}}(x) + \frac{\rho}{2} \|x - z^k + u^k\|_2^2 \Big) \tag{3.10}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} \Big(\|z\|_p^p + \frac{\rho}{2} \|x^{k+1} - z + u^k\|_2^2 \Big) \tag{3.11}$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1} \tag{3.12}$$

where u^k is the scaled dual variable.

The minimization steps (3.10) and (3.11) can be be efficiently simplified ex-

ploiting the special function structures of indicator and p-norm form. The minimization on the indicator function (3.10) takes the form of projection on the constraint set

$$x^{k+1} = \quad \Pi_{\mathcal{C}}(z^k - u^k) \tag{3.13}$$

where $\Pi_{\mathcal{C}}$ is the projection operator on the constraint set $\mathcal{C}$. The projection onto set $\mathcal{C} = \{x \in \mathbb{R}^M | Kx = b\}$ can be casted as a minimum Euclidean norm problem of underdetermined set of linear equations, where the solution takes the following closed form

$$x^{k+1} = (I - K^\dagger K)(z^k - u^k) + K^\dagger b \tag{3.14}$$

where $K^\dagger = K^T(KK^T)^{-1}$ is the Moore-Penrose pseudo-inverse of K, and $(I - K^\dagger K)$ is the projection operator on the null space of K. Hence, the first term $x_H = (I - K^\dagger K)(z^k - u^k)$ lies in the nullspace of incidence matrix $Null(K)$, also known as *cycle space* [48]; and the second term $x_P = K^\dagger b$ is the least norm-2 solution to $Kx = b$. This closed form of solution can be interpreted in a different way from graph theory perspective. The iteration update (3.14) tends to balance the feasible flow x_P by further refining a circulation flow x_H from the cycle space. The cycle space is the set of all flow vectors with zero divergence at every node $Kx = 0$, forming circular flows which pass through the loops of network graph, therefore termed as *circulation*. As iterations proceed, the initial feasible flow will be balanced by the optimal circulation flow.

The minimization in z-update step (3.11) can be carried out in a parallel way through M separate scalar minimizations on components z_m. This is due to

the fact that the p-norm function has the component separability feature, where the objective function can be separated as summation of functions on individual components, $\|z\|_p^p = \sum_{m=1}^{M}(z_m)^p$. This separability property alongside the separable quadratic term in (3.11) will result in a separable $L_\rho(x^{k+1}, z, u^k)$ and the opportunity to update z^{k+1} in a completely parallel way

$$z^{k+1} = \sum_{m=1}^{M} \underset{z_m}{\operatorname{argmin}} \Big((z_m)^p + \frac{\rho}{2}(x_m^{k+1} - z_m + u_m^k)^2 \Big) \tag{3.15}$$

The parallelization of z-update step on network flow components perfectly matches up to the aim of decentralization and makes it suitable for a distributed algorithm design. By this parallel structure, each scalar minimization can be rewritten as a root-finding problem solved locally for m-th component as follows

$$z_m^{k+1} = Root\Big[p(z_m)^{p-1} + \rho z_m - \rho(x_m^{k+1} + u_m^k) = 0 \Big] \tag{3.16}$$

where the local minimization problem reduces to finding roots for a polynomial of degree $p-1$. In order to avoid the ambiguity of having multiple roots, let p to be even, leading to monotone behavior of polynomial in (3.16) where a single root can be found easily. In the final step of ADMM iteration (3.12), the dual variable is updated.

In summary, ADMM iterations for our case can be expressed in the following three steps: (1) Projection on the constraint set, (2) Parallel flow minimization, and

(3) Dual update

$$\begin{aligned} x^{k+1} &= (I - K^{\dagger}K)(z^k - u^k) + K^{\dagger}b \\ z^{k+1} &= \sum_{m=1}^{M} \underset{z_m}{\operatorname{argmin}} \Big((z_m)^p + (\rho/2)(x_m^{k+1} - z_m + u_m^k)^2 \Big) \\ u^{k+1} &= u^k + x^{k+1} - z^{k+1} \end{aligned} \tag{3.17}$$

For convergence analysis, we know that z^{k+1} minimizes scaled objective term in (3.11). By substituting the dual variable (3.12), we will find that

$$\nabla g(z^{k+1}) = \rho u^{k+1} \tag{3.18}$$

where $g(z) = \|z\|_p^p$ is the p-norm function powered to p. Since $g(z)$ is strictly convex with Lipschitz continuous gradient, we have the inequality

$$\|\nabla g(z^{k+1}) - \nabla g(z^k)\| \leq L_p \|z^{k+1} - z^k\| \tag{3.19}$$

where $L_p \in (0, \infty)$ is the Lipschitz constant. Using (3.18), this inequality can be simplified in the following form

$$\|r^{k+1}\| \leq \frac{L_p}{\rho^2} \|s^{k+1}\| \tag{3.20}$$

where $r^{k+1} = x^{k+1} - z^{k+1}$ and $s^{k+1} = \rho(z^{k+1} - z^k)$ are respectively the primal residual and dual residual terms at iteration $k+1$. As ADMM iterations proceed, these residuals $\{r^k\}$ and $\{s^k\}$ converge to zero [42]. The linear rate of convergence is guaranteed for our proposed algorithm based on the result in [53], since the p-norm function $g(z)$ satisfies the strong convexity and Lipschitz continuous gradient conditions.

It should be noted that the convergence of ADMM to the optimal solution is independent of initial values of variables; however, a good initial point can significantly reduce the number of iterations required to converge within a desired accuracy. The main rationale behind choosing the updating order in (3.17) stem from the "warm start" idea. If we perform the projection step firstly, the primal variable will update to $x^1 = K^\dagger b = K^T(KK^T)^{-1}b$ in the first step, assuming zero initials $z^0 = u^0 = 0$. As stated earlier, this term is the minimum norm-2 solution of $Kx = b$, denoted as $x^*_{\ell\text{-}2}$. It is evident that the roughly balanced flow $x^*_{\ell\text{-}2}$ is a fairly good approximation to the optimal norm-p flow $x^*_{\ell\text{-}p}$ leading to fewer iterations required for convergence.

In our proposed ADMM iterations, two last steps (the z-update and u-update) have straightforward computations involving only local processing such as additions, multiplications, and simple root-finding process, which can be carried out in a decentralized way without the need of exchanging information between neighbor nodes; however, the projection step has more complicated computations involving matrix inversion which requires some communication steps among nodes.

3.4.2 Distributed Projection Subproblems

In this section, we will show how the projection step needs to be specially designed to perfectly match the distributed setting. We will discuss how the projection problem will reduce to solving linear subproblems in an iterative approach. The accelerated second-order iterations is employed to solve subproblems in an efficient

distributed way.

In a distributed setting, nodes perform updates based on local variables available within their one-hop neighborhood and keep track of variables associated with their outgoing links. It is easy to verify that (3.14) can be rewritten in the following simplified format

$$x^{k+1} = w^k + K^T v^k \tag{3.21}$$

where $w^k = z^k - u^k$, $r^k = b - Kw^k$, and $v^k = L^{-1}r^k$ are new auxiliary variables and $L = KK^T$ is the Laplacian matrix of network graph $\mathcal{G}$. Given v^k, it is straightforward to show that the x-update can be implemented in a distributed way. The k-th iteration of (3.21) can be interpreted in the following way: the flow x_m^k corresponding to the link $(i, j) \in \mathcal{E}$, is updated by adding virtual flow w_m^k to the difference value $sgn(j - i)(v_i^k - v_j^k)$ of nodes at the both end, where $sgn(a) = 1$ for $a \geq 0$, and $sgn(a) = -1$, otherwise. By this interpretation, we only need to develop a distributed scheme to compute v^k. Hence, the projection step at k-th ADMM iteration is reduced to solving system of linear equations expressed as

$$r^k = Lv^k \tag{3.22}$$

where the direct computation of v^k requires inversion of Laplacian matrix. We know that the distributed implementation would require each node to have access to the corresponding row. Unlike the Laplacian matrix, its inverse L^{-1} has nonzero coefficients corresponding to non-neighbor nodes at each row. Hence, the direct inversion solution for (3.22) requires centralized calculation and therefore global communication of nodes together. Several centralized graph-based algorithms have

been proposed for solving symmetric linear equations [55], [56], [57] in an efficient way. In [56], an efficient graph-based algorithm has been proposed for solving linear systems in symmetric M-matrices which yields a nearly-linear time convergence performance.

Distributed implementation of projection step (3.21) requires an inner-loop iterative scheme in order to solve linear subproblems (3.22). Each iteration step involves node-to-node communication between neighboring nodes which is the main source of drain on energy and communication resources. It is more appealing to devise a fast and computationally efficient iterative scheme which can converge to desirable accuracy by far fewer iteration steps. A classical iterative scheme which can solve linear system of equations in a distributed fashion is the well-known Jacobi method [58]. However, Jacobi method suffers from slow convergence rate which makes it practically inefficient. The fast convergent iterative scheme proposed in our previous work [43] can be a suitable candidate. This iterative scheme originally inspired by wave propagation in physical media shows superior convergence performance compared to Jacobi method. Analysis results show that this scheme significantly improves the convergence rate by reducing the number of iterations from $O(N^2)$ to $O(N)$ for the path network of N nodes. Our proposed algorithm use a second order linear iterative scheme in the following matrix form

$$v^{(l+1)} = (\alpha + 1)Sv^{(l)} - \alpha v^{(l-1)} + c \tag{3.23}$$

where l is the inner-loop iteration index. $S = I - \Delta$ is the iteration matrix containing averaging weights, and $c = (1 - \alpha)D^{-1}r^k$ is the bias vector. $\Delta = D^{-1}L$ is the

normalized Laplacian matrix, and D is the degree matrix. The iteration parameter α plays a key role in acceleration of convergence rate. As shown in (2.30) in previous chapter, the optimum iteration parameter can be computed for the general network graph in the following form

$$\alpha_{opt} = \frac{1 - \sqrt{1 - \lambda_1^2}}{1 + \sqrt{1 - \lambda_1^2}} \tag{3.24}$$

where λ_1 is the spectral radius of matrix S. Every node in network need to compute the optimum parameter α_{opt} to perform the accelerated iterations in (3.23). Hence, all nodes should have the information about spectral radius λ_1, which depends on the network topology. This fact is in conflict with the aim of distributed implementation where each node just has a degree-1 access to the network structure. In the next section, we resolve this issue by proposing an initialization stage where each node can individually estimate the spectral radius λ_1 within desired accuracy.

Algorithm 1 gives a concise and elegant presentation of our proposed algorithm as executed at each node. Assuming zero initial values $x_m^0 = z_m^0 = u_m^0 = 0$, after sufficient number of inner-loop iterations $\tilde{L}$ at each node (line 4), the linear subproblem, $Lv^k = r^k$ is solved in a distributed manner. Then, each node updates the set of virtual flows $\{(x_m, z_m, u_m) | m \in \mathcal{F}_i\}$ connected to the corresponding node in a parallel way (lines 7 to 9), where $\mathcal{F}_i = \{m | K_{im} \neq 0\}$. The x-update step in line 7 is the nodal representation of (3.21), where the m-th virtual flow corresponds to the link (i, j). This iterative procedure is repeated until reaching some stopping criterion. Finally, the resulting flow vector x is the desired balanced network flow. It should be noted that the optimal parameter is assumed to be already estimated

at each node $\hat{\alpha}_i \cong \alpha_{opt}$ in algorithm 1. The distribution estimation of α_{opt} will be presented in algorithm 2.

Algorithm 1 Main Algorithm

Initialization: for all $i \in \mathcal{V}$, set $v_i^{(-1)} = v_i^{(0)} = 0$, and for all $m \in \mathcal{F}_i$ set $x_m^0 = z_m^0 = u_m^0 = 0$, and $k = 0$

repeat
  for all $i \in \mathcal{V}$ **do**
    for $l = 0, \ldots, \tilde{L} - 1$ **do**
      $v_i^{(l+1)} = \frac{(\hat{\alpha}_i + 1)}{d_i} \sum_{i \in \mathcal{N}_i} v_i^{(l)} - \hat{\alpha}_i v_i^{(l-1)} + c_i$
    end for
    for all $m \in \mathcal{F}_i$ **do in parallel**
      $x_m^{k+1} = z_m^k - u_m^k + sgn(j-i)(v_i^{(\tilde{L})} - v_j^{(\tilde{L})})$
      $z_m^{k+1} = \underset{z_m}{\operatorname{argmin}} \Big((z_m)^p + \frac{\rho}{2}(x_m^{k+1} - z_m + u_m^k)^2 \Big)$
      $u_m^{k+1} = u_m^k + x_m^{k+1} - z_m^{k+1}$
    end for
  end for
  $k \Leftarrow k + 1$
until some stopping criterion is satisfied

3.4.3 Initialization Stage

The global information about network spectral radius required at each node seems to be in conflict with the distributed nature of our algorithm. We design

the initialization stage to overcome this obstacle in the path of achieving a fully distributed algorithm. We claim that after sufficient number of iterations, each node can individually estimate λ_1 within desired accuracy. It is noteworthy to mention that the proposed initialization routine should be implementable in a distributed way by itself.

The initialization stage starts with the process of averaging iterations, where each node updates its value by averaging the neighbor nodes' values. These iterations can be easily formulated in the matrix form as follows

$$\bar{x}(n+1) = S\bar{x}(n) + \bar{u}(n) \tag{3.25}$$

where $\bar{x}(n)$ and $\bar{u}(n)$ are vectors of size N containing states and input values of all nodes at time n respectively, and S is the averaging matrix whose spectral radius we need to estimate. It should be noted that matrix $S = I - D^{-1}L$ has the same structure as the Laplacian matrix L where nonzero elements at each row corresponds only to the neighbor nodes.

From a purely different viewpoint, an N-node network can be considered as a MIMO system of size $N \times N$ with input vector $\bar{u}(n)$ excited at the location of all nodes and the state vector $\bar{x}(n)$ containing observed values at each node at time n. The averaging iterations in (3.25) can be equivalently regarded as the first order linear state-space model for the network system. The basic idea behind the initialization stage is that the dominant mode of system can be extracted from the impulse response. In the case of linear time-invariant system, the solution can be

simplified in the following form

$$\bar{x}(n+1) = S^{n+1}\bar{x}(0) + \sum_{l=0}^{n} S^{n-l}\bar{u}(l) \tag{3.26}$$

Assuming zero initial state at all nodes, the impulse response can be written in the following form

$$\bar{x}(n+1) = S^{n}\bar{u}(0) \tag{3.27}$$

where all nodes are excited by impulse-shaped signals, $\bar{u}_i(n) = \delta(n)$ for $i = 1, \ldots, N$. Since the averaging matrix S is diagonalizable, it can be factorized by eigenvalue decomposition to $S = Q\Lambda Q^{-1}$ where Q and Q^{-1} are square matrices formed of right and left eigenvectors as their columns and rows respectively, and $\Lambda = diag[\lambda_1, \ldots, \lambda_N]$ is the diagonal matrix with eigenvalues on the diagonal. Let q_i be the i-th column of Q and $\tilde{q}_i$ be the i-th row of Q^{-1}. The state vector can be expressed as linear combination of various modes of the system as follows

$$\bar{x}(n+1) = Q\Lambda^{n}Q^{-1}\bar{u}(0) = \sum_{i=1}^{N}(\lambda_i)^{n}\langle\tilde{q}_i|\bar{u}(0)\rangle q_i \tag{3.28}$$

where $\langle .|. \rangle$ is the inner product operation. Suppose the eigenvalues to be indexed in order $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_N|$. Since we have $|\frac{\lambda_i}{\lambda_1}|^{n} \ll 1$, $i \in \{2, \ldots, N\}$ for sufficiently large n, the output response will be dominated by the term associated with the largest eigenvalue. Hence, after sufficient number of iterations, the decay rate of output response is determined by the dominant mode

$$\bar{x}(n+1) \approx (\lambda_1)^{n}\alpha_1 q_1 \tag{3.29}$$

where $\alpha_1 = \langle \tilde{q}_1 | \bar{u}(0) \rangle$. It should be noted that the network graphs with the dominant mode ratio (λ_2/λ_1) close to one, require larger number of iterations for extraction of dominant mode within a desired accuracy.

The expression in (3.29) verifies the fact that the dominant mode can be extracted from output response $\bar{x}_i(n)$ of every node in network, if the corresponding $q_{1,i} \neq 0$. By further processing, each node can easily evaluate the dominant mode as follows

$$\hat{\lambda}_{1,i}^{(n)} = \frac{\bar{x}_i(n+1)}{\bar{x}_i(n)} \approx \lambda_1 \tag{3.30}$$

where $\hat{\lambda}_{1,i}^{(n)}$ is the estimated dominant mode value from the n-th iterate of node i. As n grows large, the dominant role of λ_1 increases which will result in more accurate estimates at all nodes. Furthermore, the accuracy of $\hat{\lambda}_{1,i}^{(n)}$ estimate can be increased by computing average of dominant modes extracted from a few recent output samples. In our proposed method, each node simply performs s iterations, then the dominant mode is estimated based on average of last r estimated values

$$\hat{\lambda}_1 = \frac{1}{r} \sum_{k=s-r+1}^{s} \hat{\lambda}_1^{(k)} \tag{3.31}$$

where $\hat{\lambda_1}$ is the vector of size N containing estimated dominant mode values at all nodes. It should be noted that the process of evaluating dominant mode at each node does not require any further communication between nodes. In other words, every node in network can individually compute an accurate estimate of the dominant mode of system or namely spectral radius of matrix S after sufficient number of iterations. Hence, we achieve an efficient distributed method to estimate the spectral radius of matrix S and consequently optimum parameter α_{opt}.

Algorithm 2 shows the initialization setup required for the accelerated iterations solving the projection subproblems. As we see, each node computes an estimate of optimum parameter, $\hat{\alpha}_i \cong \alpha_{opt}$, after s averaging iterations.

Algorithm 2 Initialization Stage

Initialization: for all $i \in \mathcal{V}$, set $\bar{x}_i(0) = 0$, and impulse input $\bar{u}_i(n) = \delta(n)$

1: **for all** $i \in \mathcal{V}$ **do**

2: **for** $n = 0, \ldots, s-1$ **do**

3: $\bar{x}_i(n+1) = \frac{1}{d_i} \sum_{i \in \mathcal{N}_i} \bar{x}_i(n) + \bar{u}_i(n)$

4: **end for**

5: $\hat{\lambda}_{1,i} = \frac{1}{r} \sum_{k=s-r}^{s-1} \frac{\bar{x}_i(k+1)}{\bar{x}_i(k)}$

6: $\hat{\alpha}_i = \frac{1-\sqrt{1-\hat{\lambda}_{1,i}^2}}{1+\sqrt{1-\hat{\lambda}_{1,i}^2}}$

7: **end for**

Chapter 4: Numerical Results

The proposed distributed load balancing protocol is simulated over a network of $N = 100$ nodes randomly distributed over a square of side 100 units. As shown in figure 4.1, network graph has a relatively complex topology inspired by sensor networks, where each node communicates with the neighbor nodes not farther than a specific coverage radius. Five nodes have been chosen arbitrarily as source nodes generating uneven $(0.5, 0.2, 0.1, 0.1, 0.1)$ units of information which would be routed through the sink node. The total generated traffic is normalized to one without loss of generality. The link capacity is assumed to be the same and normalized to one for all links. As highlighted in figure, the generated network graph turns out to have a bottleneck with 3 mainstream flows. The penalty factor is assumed to be equal to $\rho = 1$ for ADMM iterations.

4.1 Convergence Performance

Figure 4.2 shows the convergence performance of the proposed ADMM-based algorithm for the generated network graph. To highlight the significance of flows on bottleneck, this figure contains the convergence performance of most significant flows in network with larger absolute values. It is interesting to note that our proposed

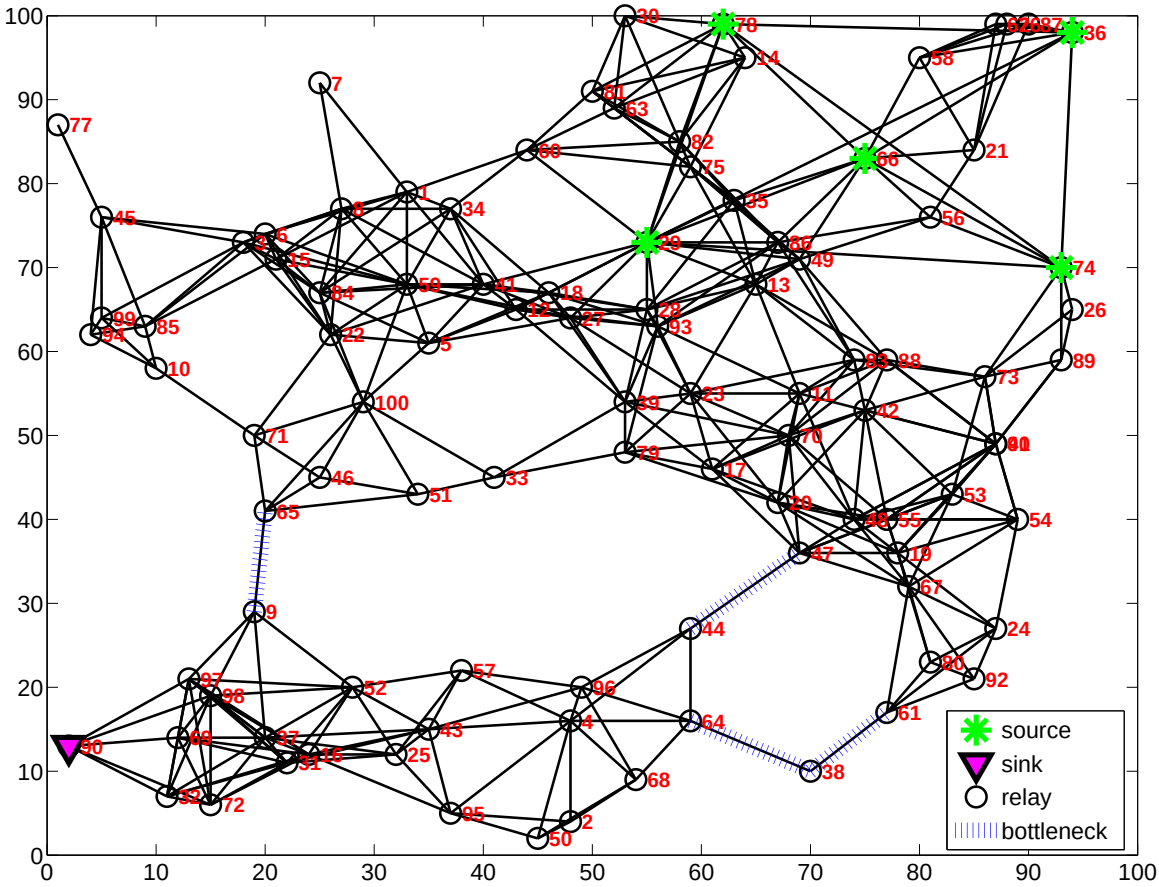

Figure 4.1: A complex network graph of $N = 100$ nodes and $M = 386$ edges randomly generated over a square of side 100 units, with five designated source nodes and a single sink node. The minimum cutset or namely *bottleneck* of network graph has been highlighted in dashed blue lines.

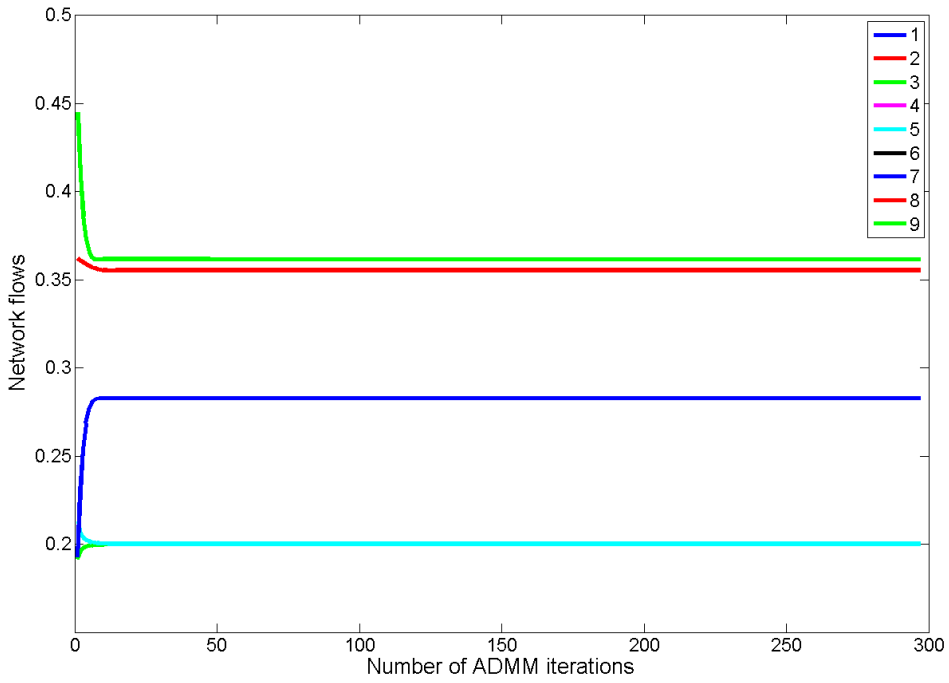

Figure 4.2: The convergence performance of the load-balanced network flow using the proposed ADMM-based algorithm applied to the generated network in figure 4.1 for the most significant network flows.

method requires only 4 ADMM iterations to balance network flows on bottleneck links within 5% accuracy to the optimal solution of p-norm problem $x^*_{\ell\text{-}p}$. Simulation results show that by further iteration steps, network flow will be redistributed more evenly throughout the network. Figures 4.3 and 4.4 confirm the linear convergence rate of the residuals validating the excellent convergence performance of the proposed algorithm. Finally, figure 4.5 shows the iterative solution of distributed projection subproblems for the first four ADMM-based iteration steps using the accelerated iterative scheme in (3.23). This figure plots the node values v^k versus number of inner-loop iterations at all nodes.

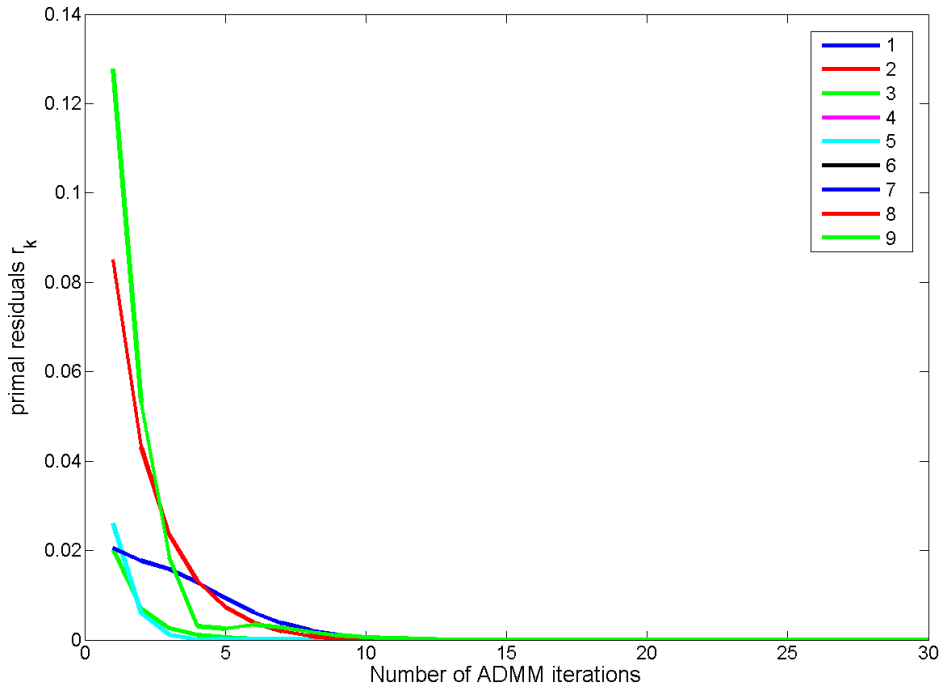

Figure 4.3: The convergence performance of primal residuals r_k showing the amount of violation from the flow conservation.

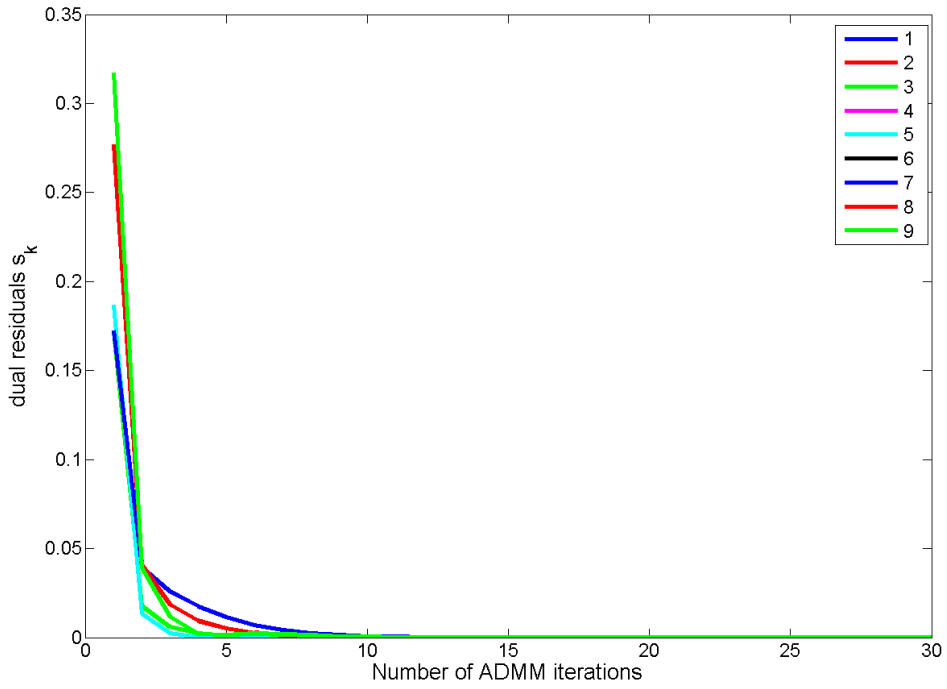

Figure 4.4: The convergence performance of dual residuals s_k.

4.2 initialization stage

The impulse response of the network system is depicted in figure 4.6 for all nodes. This figure illustrates the fact that dominant mode can be extracted from the impulse response at the location of all nodes, as the decay rates are roughly equal for all nodes after sufficient number of iterations. Hence, all nodes can estimate the dominant mode individually. The dominant mode true value for the generated network graph in (4.1) is equal to $\lambda_1 = 0.9984$. In table (4.1), we have shown the number of iterations required to estimate the dominant mode within a desired accuracy. The mean squared error $MSE = E[(\hat{\lambda_1} - \lambda_1)^2]$ is chosen as the estimation accuracy measure. For the case of our generated network, we chose the parameter

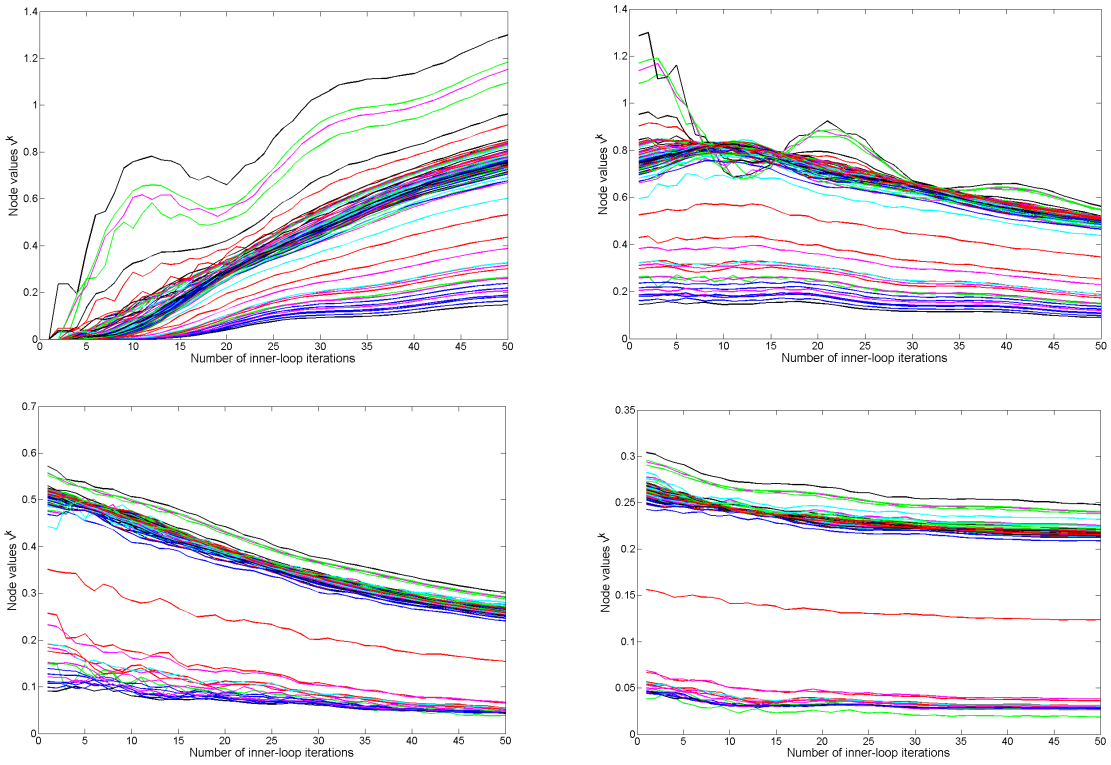

Figure 4.5: Convergence performance of the inner-loop accelerated iterative scheme for the first four ADMM iteration steps at all nodes in network graph 4.1.

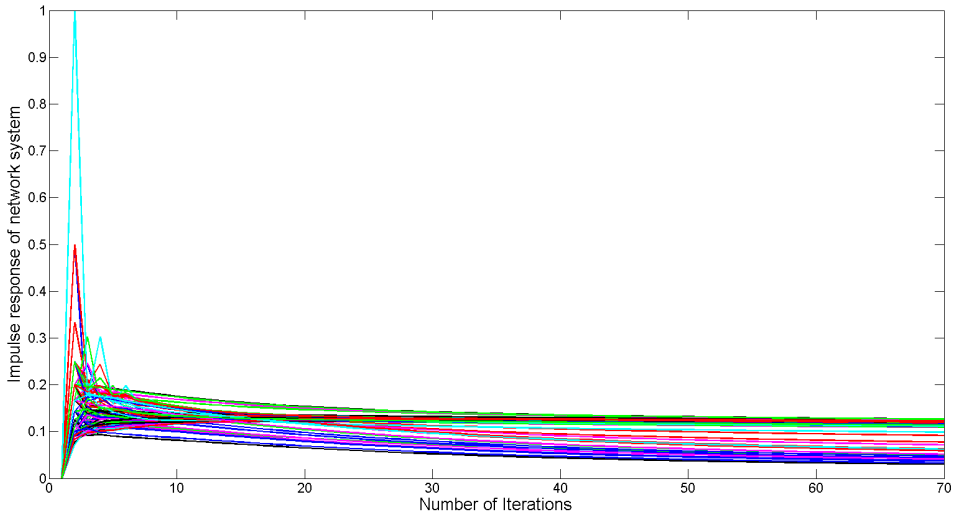

Figure 4.6: Impulse response of generated network graph at the location of all nodes.

$r = 10$, where every node perform total s averaging iterations and the dominant mode is estimated based on average of last 10 estimated values. Since the dominant mode ratio for the generated graph (4.1) is close to one, larger averaging operations is required to achieve a specific accuracy. It should be noted that these iterations are performed only once as the initialization setup and would not increase the total computational cost.

Accuracy measure (MSE)	Number of iterations required
10^{-4}	25
10^{-5}	70
10^{-6}	105

Table 4.1: Dominant mode estimation within a specific accuracy for the generated network graph.

Bibliography

[1] Rahul C Shah and Jan M Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 350–355. IEEE, 2002.

[2] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25, 2001.

[3] Yashar Ganjali and Abtin Keshavarzian. Load balancing in ad hoc networks: single-path routing vs. multi-path routing. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1120–1125. IEEE, 2004.

[4] Mounir Frikha. *Ad Hoc Networks: Routing, QoS and Optimization*. John Wiley & Sons, 2013.

[5] Banani Das and Sudipta Roy. Load balancing techniques for wireless mesh networks: A survey. In *Computational and Business Intelligence (ISCBI), 2013 International Symposium on*, pages 247–253. IEEE, 2013.

[6] Yigal Bejerano, Seung-Jae Han, and Amit Kumar. Efficient load-balancing routing for wireless mesh networks. *Computer Networks*, 51(10):2450–2466, 2007.

[7] Liang Ma and Mieso K Denko. A routing metric for load-balancing in wireless mesh networks. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 409–414. IEEE, 2007.

[8] Hossam Hassanein and Audrey Zhou. Routing with load balancing in wireless ad hoc networks. In *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 89–96. ACM, 2001.

[9] Chai Keong Toh, Anh-Ngoc Le, and You-Ze Cho. Load balanced routing protocols for ad hoc mobile wireless networks. *Communications Magazine, IEEE*, 47(8):78–84, 2009.

[10] Shouyi Yin and Xiaokang Lin. Malb: Manet adaptive load balancing. In *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, volume 4, pages 2843–2847. IEEE, 2004.

[11] Pai-Hsiang Hsiao, Adon Hwang, HT Kung, and Dario Vlah. Load-balancing routing for wireless access networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 986–995. IEEE, 2001.

[12] Huazhi Gong and JongWon Kim. Dynamic load balancing through association control of mobile users in wifi networks. *Consumer Electronics, IEEE Transactions on*, 54(2):342–348, 2008.

[13] Tarik Taleb, Daisuke Mashimo, Abbas Jamalipour, Nei Kato, and Yoshiaki Nemoto. Explicit load balancing technique for ngeo satellite ip networks with on-board processing capabilities. *Networking, IEEE/ACM Transactions on*, 17(1):281–293, 2009.

[14] Hongyi Wu, Chunming Qiao, Swades De, and Ozan Tonguz. Integrated cellular and ad hoc relaying systems: icar. *Selected Areas in Communications, IEEE Journal on*, 19(10):2105–2115, 2001.

[15] Wei Song, Weihua Zhuang, and Yu Cheng. Load balancing for cellular/wlan integrated networks. *Network, IEEE*, 21(1):27–33, 2007.

[16] Evsen Yanmaz and Ozan K Tonguz. Dynamic load balancing and sharing performance of integrated wireless networks. *Selected Areas in Communications, IEEE Journal on*, 22(5):862–872, 2004.

[17] Jiajia Liu, Yuichi Kawamoto, Hiroki Nishiyama, Nei Kato, and Naoto Kadowaki. Device-to-device communications achieve efficient load balancing in lte-advanced networks. *Wireless Communications, IEEE*, 21(2):57–65, 2014.

[18] Jae-Hwan Chang and Leandros Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 12(4):609–619, 2004.

[19] Ritesh Madan, Shuguang Cui, Sanjay Lall, and Andrea Goldsmith. Cross-layer design for lifetime maximization in interference-limited wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 5(11):3142–3152, 2006.

[20] Esa Hyytiä and Jorma Virtamo. On traffic load distribution and load balancing in dense wireless multihop networks. *EURASIP Journal on Wireless Communications and Networking*, 2007(1):21–21, 2007.

[21] Daniele Puccinelli and Martin Haenggi. Arbutus: Network-layer load balancing for wireless sensor networks. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 2063–2068. IEEE, 2008.

[22] Gaurav Gupta and Mohamed Younis. Load-balanced clustering of wireless sensor networks. In *Communications, 2003. ICC'03. IEEE International Conference on*, volume 3, pages 1848–1852. IEEE, 2003.

[23] Hui Dai and Richard Han. A node-centric load balancing algorithm for wireless sensor networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 1, pages 548–552. IEEE, 2003.

[24] Ritesh Madan and Sanjay Lall. Distributed algorithms for maximum lifetime routing in wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 5(8):2185–2193, 2006.

[25] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[26] Roland Glowinski and A Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

[27] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[28] Gong Chen and Marc Teboulle. A proximal-based decomposition method for convex minimization problems. *Mathematical Programming*, 64(1-3):81–101, 1994.

[29] Paul Tseng. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, 29(1):119–138, 1991.

[30] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[31] Junfeng Yang and Yin Zhang. Alternating direction algorithms for ℓ_1 problems in compressive sensing. *SIAM journal on scientific computing*, 33(1):250–278, 2011.

[32] Manya V Afonso, José M Bioucas-Dias, and Mário AT Figueiredo. Fast image recovery using variable splitting and constrained optimization. *Image Processing, IEEE Transactions on*, 19(9):2345–2356, 2010.

[33] Mário AT Figueiredo and José M Bioucas-Dias. Restoration of poissonian images using alternating direction optimization. *Image Processing, IEEE Transactions on*, 19(12):3133–3145, 2010.

[34] Ioannis D Schizas, Alejandro Ribeiro, and Georgios B Giannakis. Consensus in ad hoc wsns with noisy linkspart i: Distributed estimation of deterministic signals. *Signal Processing, IEEE Transactions on*, 56(1):350–364, 2008.

[35] Ioannis D Schizas, Georgios B Giannakis, Stergios I Roumeliotis, and Alejandro Ribeiro. Consensus in ad hoc wsns with noisy linkspart ii: Distributed estimation and smoothing of random signals. *Signal Processing, IEEE Transactions on*, 56(4):1650–1666, 2008.

[36] Min Tao and Xiaoming Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011.

[37] Michael K Ng, Pierre Weiss, and Xiaoming Yuan. Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods. *SIAM journal on Scientific Computing*, 32(5):2710–2736, 2010.

[38] Pedro A Forero, Alfonso Cano, and Georgios B Giannakis. Consensus-based distributed support vector machines. *The Journal of Machine Learning Research*, 11:1663–1707, 2010.

[39] Daniel Gabay. Chapter ix applications of the method of multipliers to variational inequalities. *Studies in mathematics and its applications*, 15:299–331, 1983.

[40] Jonathan Eckstein and Dimitri P Bertsekas. On the douglasrachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[41] Ernie Esser. Applications of lagrangian-based alternating direction methods and connections to split bregman. *CAM report*, 9:31, 2009.

[42] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[43] Alireza Sheikhattar and Mehdi Kalantari. Fast convergence scheme for potential-based routing in wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 1980–1985. IEEE, 2013.

[44] Mehdi Kalantari, Masoumeh Haghpanahi, and Mark Shayman. A p-norm flow optimization problem in dense wireless sensor networks. In *INFOCOM 2008.*

The 27th Conference on Computer Communications. IEEE, pages 341–345. IEEE, 2008.

[45] Sina Zahedpour Anaraki. *Distributed flow optimization in dense wireless networks*. PhD thesis, 2011.

[46] Anirban Banerjee and Jürgen Jost. On the spectrum of the normalized graph laplacian. *Linear algebra and its applications*, 428(11):3015–3022, 2008.

[47] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[48] Béla Bollobás. *Modern graph theory*, volume 184. Springer, 1998.

[49] Roland Glowinski and Patrick Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, volume 9. SIAM, 1989.

[50] Donald Goldfarb and Shiqian Ma. Fast multiple-splitting algorithms for convex optimization. *SIAM Journal on Optimization*, 22(2):533–556, 2012.

[51] Donald Goldfarb, Shiqian Ma, and Katya Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, 141(1-2):349–382, 2013.

[52] Tom Goldstein, Brendan ODonoghue, and Simon Setzer. Fast alternating direction optimization methods. *CAM report*, pages 12–35, 2012.

[53] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. Technical report, DTIC Document, 2012.

[54] Jonathan Eckstein. Parallel alternating direction multiplier decomposition of convex programs. *Journal of Optimization Theory and Applications*, 80(1):39–62, 1994.

[55] Daniel A Spielman. Algorithms, graph theory, and linear equations in laplacian matrices. In *Proceedings of the International Congress of Mathematicians*, volume 4, pages 2698–2722, 2010.

[56] Samuel Isaac Daitch. *Efficient graph-based algorithms for linear equations, network flows, and machine learning*. Yale University, 2009.

[57] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.

[58] Roland Bulirsch and Josef Stoer. *Introduction to numerical analysis*. Springer Heidelberg, 2002.