ABSTRACT

Title of thesis: DISTRIBUTED FLOW OPTIMIZATION
IN DENSE WIRELESS NETWORKS
Sina Zahedpour Anaraki, Master of Science, 2011

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

Due to large number of variables, optimizing information flow in a dense wireless network using discrete methods can be computationally prohibitive. Instead of treating the nodes as discrete entities, these networks can be modeled as continuum of nodes providing a medium for information transport. In this scenario multi-hop information routes transform into an *information flow* vector field that is defined over the geographical domain of the network. At each point of the network, the orientation of the vector field shows the direction of the flow of information, and its magnitude shows the density of information flow. Modeling the dense network in continuous domain enables us to study the large scale behavior of the network under existing routing policies; furthermore, it justifies incorporation of multivariate calculus techniques in order to find new routing policies that optimize a suitable cost function, which only depend on large scale properties of the network. Thus, finding an optimum routing method translates into finding an optimal information flow vector field that minimizes the cost function.

In order to transform the optimal information flow vector field into a routing

policy, connections between discrete space (small scale) and continuous space (large scale) variables should be made and the question that how the nodes should interact with each other in the microscopic scale in order that their large scale behavior become optimal should be answered. In the past works, a centralized method of calculating the optimal information flow over the entire geographical area that encompasses the network has been suggested; however, using a centralized method to optimize information flow in a dynamic network is undesirable. Furthermore, the value of information flow vector field is needed only at the locations of randomly scattered nodes in the network, thus calculation of the information flow vector field for the entire network region (as suggested in previous models) is an unnecessary overhead. This poses a gap between the continuous space and discrete space models of information flow in dense wireless networks. This gap is how to calculate and apply the optimum information flow derived in continuous domain to a network with finite number of nodes. As a first step to fill this gap, a specific quadratic cost function is considered. In previous works, it is proved that the the vector field that minimizes this cost function is irrotational, thus it is written as the gradient of a potential function. This potential function satisfies a Poisson Partial Differential Equation (PDE) which in conjunction with Neumann boundary condition has a unique solution up to a constant. In this thesis the PDE resulted by optimization in continuous domain is discretized at locations of the nodes. By assuming a random node distribution with uniform density, the symmetries present enable us to solve the PDE in a distributed fashion. This solution is based on Jacobi iterations and requires only neighboring nodes to share their local information with each other.

The gradient of the resulting potential defines the routes that the traffic should be forwarded.

Furthermore, based on a graph theory model, we generalize our distributed solution to a more general cost function, namely, the p-norm cost function. This model also enables us to enhance the convergence rate of the Jacobi iterations.

DISTRIBUTED FLOW OPTIMIZATION IN DENSE WIRELESS NETWORKS

by

Sina Zahedpour Anaraki

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
2011

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

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I would like to thank my advisor, Dr. Mehdi Kalantari for giving me an invaluable opportunity to work on challenging and extremely interesting projects. He has always made himself available for help and advice and there has never been an occasion when I've knocked on his door and he hasn't given me time. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank my co-advisor, Professor Richard La. Without his help and ideas, this thesis would have been a distant dream. Thanks are due to Professor Mark Shayman for both teaching the random processes course, agreeing to serve on my thesis committee and for sparing his invaluable time reviewing the manuscript.

I owe my deepest thanks to my family - my mother, father and sister who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them.

This work was partially supported by NSF under grant CCF-0729129 and grant 0931957.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

PDE	Partial Differential Equation
BC	Boundary Condition
MAC	Medium Access Control
FDM	Finite Difference Method
FEM	Finite Element Method
CG	Conjugate Gradient
GS	Gauss Seidel
SOR	Successive Over Relaxation
SQP	Sequential Quadratic Programming
rv	Random variable
i.i.d.	Independent and identically distributed

Chapter 1

Introduction

A dense wireless sensor network is a network of large number of homogenous devices distributed in a geographical area to collect data about environmental conditions or events. Each device is equipped with a microcontroller, a short range radio transceiver, and a battery. Common applications of such networks include distributed systems of sensors and actuators, and monitoring and surveillance of a geographic area. Recent advancements in low power electronics and wireless communications have made it possible to manufacture such devices (which hereafter we call them nodes) in very small size and with low cost. As a result, very large scale deployment of these devices is possible in applications of interest. The information generated by each node is transmitted through radio connection to a special node with enough energy and computational power to be further processed using network data fusion techniques[2]. We refer to this node as the data sink or *sink* in short; note that large networks may have several sinks. Most of the nodes that generate traffic are not in the communication range of the sinks, thus data packets should be relayed along multi-hop paths through other nodes to be delivered to data sinks.

Since the nodes are usually powered by battery, it is important that the information flow from the sources to sinks be such that the traffic load is not concentrated on a small number of bottleneck nodes, otherwise their batteries will be depleted

and the connectivity of the network can be lost. In the modern wireless devices, the main source of energy consumption is radio front-end [22], thus excessive radio transmissions accelerates the depletion of the battery. Hence, developing Medium Access Control (MAC) and routing protocols for these conditions is and active research topic [27, 24, 20]. In this thesis, we focus on studying information flow for dense wireless networks, based on the framework developed in previous works by [18, 17, 19].

Common methods for studying and analyzing wireless networks model the network as a discrete set of connected nodes [1]. As the density of nodes in a network grows, careful behavioral analysis (in small scale) and global optimization of routing protocols becomes very hard by using conventional methods that employ a discrete model in space, as the number of variable grow rapidly.

A proposed method to overcome complexity issues of a dense wireless network is to model the network as a continuum of nodes providing a medium for information flow. In this model, information is treated as a fluid-like entity being transported through a massively dense communication medium. The information generated by each node goes through a sequence of many small range multi-hop transmissions in the medium of nodes until it is received by a sink. This continuous space model for flow of information in a dense network was introduced in previous works [18, 17, 19] and independently in [26, 25].

In the proposed continuous space model, an *information flow vector field* models the transportation of nodes' traffic. This vector field has two components at each location of the network: a magnitude representing the spatial density of information

that flows at that location and an orientation that gives the direction to which the traffic is forwarded. Basic flow conservation laws result in a differential equation and a boundary condition that govern information flow in the entire netwrok. Based on these equations the following observations are made. Firstly, the flow conservation equations by themselves do not define a unique information flow vector field. This opens room to find a vector field that minimizes a suitable cost function. In the first part of this thesis we consider a cost function that is a *quadratic* form of the magnitude of information density (which in turn is proportional to the communication cost in terms of energy). This optimization results in another differential equation. Secondly, the information flow vector field that solves these equations is conservative (or irrotational). As a result, the information flow vector field is the gradient of a potential function defined over the domain of the network. As we see later, this potential function defines the paths that traffic flows in the network. Finally, in [26], it is shown that finding an information flow vector field that minimizes the quadratic cost function, also minimizes the number of nodes required to carry a specified information density in a network.

It should be noted that the form of flow conservation laws governing the information flow vector field parallel some of Maxwell's equations (more specifically Gauss' Law and Maxwell-Faraday equation) studied in electrostatics. As an example, Maxwell-Faraday equation states that the electric field is conservative in a static system, thus it can be represented by an electric potential. An analogous flow conservation applies to the information flow vector field; i.e., it can be represented by a potential function. The potential function satisfies a Poisson Partial Differential

Equation (PDE) with Neumann boundary condition. Solution of this PDE leads to a simple mechanism for routing the traffic: each node forwards the traffic to a neighbor with least potential (steepest potential decent).

An important advantage of continuous space models for information flow is the use of strong analytical tools and techniques in vector calculus and partial differential equations for flow optimization; however, a main shortcoming of the previous works in continuous information flow models is that the potential function is assumed to be calculated in a central node; All of the previous works that use continuous model for information flow assume that a central node calculates the potential as a function of space and then it sends the results to all nodes. This is a significant drawback and usefulness of continuous space models of information flow models will be questionable in absence of practical methods that calculate the potential values at the locations of nodes in a decentralized way.

The contribution for this thesis is twofold; firstly, in this work we bridge the gap between continuous spaces models and discrete space models of information flow in massively dense wireless networks. We study the network in both continuous and discrete space and make connections between them. Secondly, we provide a method to compute the potential at each node in a distributed fashion such that each node computes its potential by using simple iterations. Such iterations use the potential value at a node and the values broadcasted only by its neighboring nodes to find a new potential value for that node. Simulations show that the iterations at each node converge to the continuous space potential function at the location of that node. The discrete and decentralized scheme for calculation of potential function enables the

routing method based on the potential function to be employed in networks with large number of nodes in a geographically distributed region.

To achieve this goal, the Poisson PDE in continuous domain is approximated by equations in discrete domain that is valid at locations of wireless nodes. We use a method similar to Finite Difference Method (FDM) for a random setting suited for wireless networks in which the nodes are distributed independently in the network area with uniform distribution. The symmetries present in a uniform distribution enable us to solve these equations using Jacobi iterations, without any information about the distance of the neighboring nodes; i.e., each wireless node only needs to communicate with its neighboring nodes to compute and update its potential value, based on a simple formula. The broadcast nature of the wireless communication is directly used in the proposed method.

Related Works: The continuous domain flow presented in this work falls in the category of *physical inspired* models. Toumpis in [26] has independently proposed a similar continuous flow method inspired by electric fields in electrostatics problems. His other work in [25] provides a survey and reference list for other physical inspired methods, most importantly, the seminal work of Jacquet [13] where he finds analogies between flow paths in a network with varying density to paths of rays of light in an inhomogeneous medium. In [4], Altman considers a network with directional antennae and uses traffic engineering concepts in order to find optimal traffic paths. A comprehensive survey of routing methods in wireless sensor networks is provided in [3, 1, 2]. A recent work by Jung *et al.* [14, 15] proposes applying a Finite Element Method (FEM) to solve the Poisson equation, however it is unclear

how FEM can be applied in a random setting where the nodes are not aware of their positions and distances with respect to each other.

1.1 Notations and Definitions

Random variables (rv's) are denoted by boldface, and vectors are designated by an arrow. We assume that the network is distributed in a closed, bounded and connected region $\mathcal{A} \subset \mathbb{R}^2$ and denote the boundary of this region by $\partial\mathcal{A}$. A position vector is denoted by $\vec{r} = [x, y]$ in Cartesian coordination system, and by $\vec{r} = (r, \phi)$ in a polar system. The gradient of a scalar valued function $u(\vec{r})$ is $\nabla u(\vec{r}) = \frac{\partial}{\partial x}u(\vec{r})\hat{x} + \frac{\partial}{\partial y}u(\vec{r})\hat{y}$, where $\hat{x}$ and $\hat{y}$ are the unit vectors of the Cartesian coordinates along the x and y axes respectively. The divergence and curl of a vector field $\vec{D}(\vec{r}) = D_x(\vec{r})\hat{x} + D_y(\vec{r})\hat{y}$ are $\nabla \cdot \vec{D}(\vec{r}) = \frac{\partial}{\partial x}D_x(\vec{r}) + \frac{\partial}{\partial y}D_y(\vec{r})$ and $\nabla \times \vec{D}(\vec{r}) = \big(-\frac{\partial}{\partial y}D_x(\vec{r}) + \frac{\partial}{\partial x}D_y(\vec{r})\big)\hat{z}$, respectively, where $\hat{z} = \hat{x} \times \hat{y}$ represents the unit vector in direction of z axis.

1.2 Background on Information Flow Vector Field

In this section we review the previous works on flow optimization in dense wireless networks [18, 17, 19]. Typically in a wireless network, the source and destination of traffic are not in communication range of each other, thus the traffic should be relayed through intermediate nodes in order to reach the destination. Figure 1.1(a) depicts an example of a multi-hop path. As the number of nodes grows large and the communication range of the nodes get smaller, the path approaches

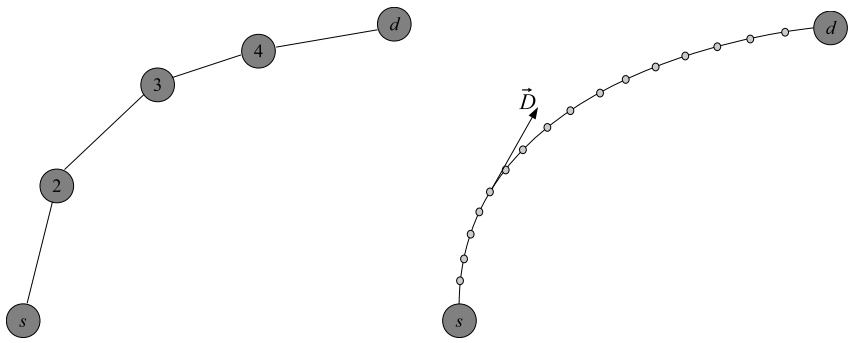

Figure 1.1: When the source and destination nodes are not in communication range, traffic is relayed through multi-hop path. As the number of nodes in the path gets large, this path approaches to a smooth curve.

to a smooth curve that connects the source to destination. This is depicted in figure 1.1(b). At each point, $\vec{r}$, on the curve a tangent vector $\vec{D}$ is defined, such that its magnitude is proportional to the amount of traffic and its direction specifies the direction in which the traffic is relayed by a node at point $\vec{r}$, with infinitesimally small communication range. As the density of nodes approaches infinity, $\vec{D}$ (which hereafter we call information flow vector field), becomes defined not only at the locations of the nodes, but over the entire domain of the network $\mathcal{A}$. The network in this situation becomes a continuum of nodes which provides a medium that passes the flow form sources to destinations through infinitesimal small hops. Let $\mathcal{C}$ be a closed curve in $\mathcal{A}$, $\hat{n}$ denote the unit normal vector at each point on the curve and $d\ell$ denote an element of the curve, then based on the definition of information flow vector field, $\oint_{\mathcal{C}} \vec{D} \cdot \hat{n} d\ell$ is the total flow (in bps) that passes through $\mathcal{C}$.

Let $\rho : \mathcal{A} \to \mathbb{R}$ model the spatial density of information sources that generate the flow at each point of the network, i.e., at point $\vec{r}$, $\rho(\vec{r})$ (bps/m^2) of traffic is generated. If $\rho(\vec{r}) > 0$, flow is being injected in the network, whereas $\rho(\vec{r}) < 0$

means there is a sink at point $\vec{r}$. Let $\mathcal{S}$ be in $\mathcal{A}$ and ds be a surface element of $\mathcal{S}$, then $\int_{\mathcal{S}} \rho(\vec{r})ds$ is the total amount of flow generated in (or depending on the sign, taken out of) the network by the nodes in area $\mathcal{S}$. The definitions of information flow vector field $\vec{D}$ and the source term ρ is not complete unless we assume a relation between the two.

Assumption 1.1. *Information flow is conserved in the network.*

It follows from assumption 1.1 that $\oint_{\partial\mathcal{S}} \vec{D} \cdot \hat{n} d\ell = \int_{\mathcal{S}} \rho(\vec{r})ds$, for every $\mathcal{S} \subseteq \mathcal{A}$. However, since $\int_S \nabla \cdot \vec{D} ds = \int_{\partial S} \vec{D} \cdot \hat{n} dl$ (*divergence theorem*), it follows that

$$\int_S \nabla \cdot \vec{D} ds = \int_{\mathcal{S}} \rho(\vec{r})ds$$

is true for every $\mathcal{S} \subseteq \mathcal{A}$ which leads to

$$\nabla \cdot \vec{D}(\vec{r}) = \rho(\vec{r}) \quad \vec{r} \in \mathcal{A} \tag{1.1}$$

Assumption 1.2. *The information flow vector field has no normal component at the boundary of $\mathcal{A}$.*

A non-zero normal component on the boundary, means that a node on the boundary is trying to forward flow to outside the network, where there are no nodes available. By assumption 1.2 we limit $\vec{D}$ to be such that all the traffic is contained in the network region $\mathcal{A}$. Thus we have

$$\vec{D}(\vec{r}) \cdot \hat{n} = 0 \quad \vec{r} \in \partial\mathcal{A} \tag{1.2}$$

where $\hat{n}$ is the unit normal vector on the boundary, $\partial\mathcal{A}$. Equation (1.1) and (1.2) do not define $\vec{D}$ uniquely, which opens room for finding a flow that optimizes a suitable cost function. In previous works [18, 17, 19], vector fields that optimize a quadratic cost function of the form

$$J(\vec{D}) = \int_{\mathcal{A}} \|\vec{D}\|^2 ds \tag{1.3}$$

were studied.

Lemma 1.1. *A vector field $\vec{D}$ with $\nabla \cdot \vec{D} = \rho$ and $\vec{D}(\vec{r}) \cdot \hat{n} = 0$ optimizes (1.3) if and only if $\nabla \times \vec{D} = 0$.*

Proof. Let $\vec{D}$ be a general vector field with $\nabla \cdot \vec{D} = \rho$ and $\vec{D} \cdot \hat{n} = 0$. By Helmholtz decomposition, for every vector field $\vec{D}$ with smooth enough derivatives (which we assume $\vec{D}$ satisfies), we can write $\vec{D} = \vec{G} + \vec{F}$, with $\vec{G} = -\nabla u$, $\nabla \cdot \vec{F} = 0$ and $\vec{F} \cdot \hat{n} = 0$ on the boundary [1]. Using integration by parts we derive that $\vec{G}$ and $\vec{F}$ are orthogonal,

$$\begin{aligned}
\int_{\mathcal{A}} \vec{G} \cdot \vec{F} ds &= -\int_{\mathcal{A}} \nabla u \cdot \vec{F} ds \\
&= -\int_{\partial\mathcal{A}} u\vec{F} \cdot \hat{n} dl + \int_{\mathcal{A}} u \nabla \cdot \vec{F} ds \\
&= 0
\end{aligned}$$

[1] $\vec{G}$ captures the irrotational component of $\vec{D}$, whereas $\vec{F}$ captures the rotational part. $\vec{G}$ is simply found by solving $\nabla^2 u(\vec{r}) = -\rho(\vec{r})$, $\vec{r} \in \mathcal{A}$, $\nabla u(\vec{r}) \cdot \hat{n} = 0$, $\vec{r} \in \partial\mathcal{A}$. Furthermore, $\vec{F} = \vec{D} + \nabla u$.

now we have

$$\begin{aligned} J(\vec{D}) &= \int_{\mathcal{A}} \|\vec{G}\|^2 ds + \int_{\mathcal{A}} \|\vec{F}\|^2 ds + 2\int_{\mathcal{A}} \vec{G}\cdot\vec{F} ds \\ &= \int_{\mathcal{A}} \|\vec{G}\|^2 ds + \int_{\mathcal{A}} \|\vec{F}\|^2 ds \\ &\geq J(\vec{G}) \end{aligned} \tag{1.4}$$

with equality if and only if $\vec{F} \equiv 0$. Thus a vector is minimizes J if and only if $\vec{D} = -\nabla u$ or $\nabla \times \vec{D} = 0$. □

Lemma 1.1 enables us to write (1.1) and in the form

$$\nabla \cdot \nabla u(\vec{r}) = -\rho(\vec{r}) \quad \vec{r} \in \mathcal{A}$$

$$\nabla u(\vec{r}) \cdot \hat{n} = 0 \quad \vec{r} \in \partial\mathcal{A}$$

or

$$\nabla^2 u(\vec{r}) = -\rho(\vec{r}) \quad \vec{r} \in \mathcal{A} \tag{1.5}$$

$$\frac{\partial u(\vec{r})}{\partial n} = 0 \quad \vec{r} \in \partial\mathcal{A} \tag{1.6}$$

which is the Poisson PDE with Neumann boundary condition. Due to its similarity with the electric potential function in electrostatics, we refer to u as the potential function.

Remark. From (1.1) and (1.2) we have

$$\begin{aligned} \int_{\mathcal{A}} \rho ds &= \int_{\mathcal{A}} \nabla \cdot \vec{D} ds \\ &= \int_{\partial\mathcal{A}} \vec{D} \cdot \hat{n} d\ell = 0 \end{aligned} \tag{1.7}$$

i.e., there rate at which the sources generated traffic should be equal to the rate at which the sinks gather it.

Remark. Equation (1.5) in conjunction with (1.6) expresses a problem with a unique solution up to a constant; i.e., if u is a solution then $u + c$, c being a constant, is also a solution. However this issue does not pose a problem for finding optimal information flow vector field, $\vec{D} = -\nabla u$, since regardless of c, the flow paths will remain the same, as $\nabla(u + c) = \nabla u$.

Chapter 2

Distributed Solution for Optimizing the Quadratic Cost Function in Continuous Space

In the previous chapter we derived the PDE and boundary condition governing the optimal information flow for quadratic cost function $J(\vec{D})$, and showed that the flow is written in terms of the gradient of the potential function. This implies that having the potentials calculated, the nodes simply forward the flow along the direction of steepest decent of potential (neighbor node with smallest potential). This packet routing scheme results in flow paths in the network that implement the continuous domain information flow vector field in the discrete setting of the real network.

In this chapter, we will provide a distributed method to calculate the potential function u numerically at the locations of the nodes, where the nodes are distributed randomly and independently, with uniform distribution over the network area $\mathcal{A}$. Our proposed distributed method has three properties: first, it is iterative, and in each iteration a node updates its own potential (u) only based on the potentials of its neighboring nodes. Second, since the computational power of the nodes is assumed to be small, the iterations should be simple, which in case of our proposed method they are. Third, since the nodes are deployed massively, it is desirable that the nodes be homogeneous, i.e., they all perform the same iterations regardless of

their location in the network (e.g., nodes close to the boundary and nodes in the interior).

2.1 Distributed Solution to Poisson PDE on a Uniform Grid

In this section, we focus on solving the Poisson PDE with Neumann boundary conditions, when the nodes are positioned on a *uniform grid*. We will extend this special case to a more general random setting in the next section.

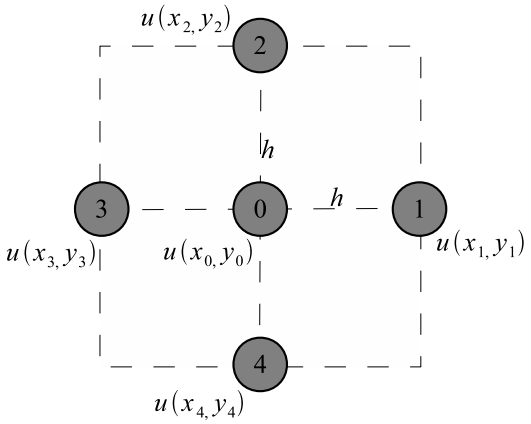

Figure 2.1: Nodes $0 \ldots 4$ are placed on a grid. The potential of the nodes is calculated by (2.5).

A simple method to solve an elliptic PDEs numerically is finite difference method, where in its simplest form, the region $\mathcal{A}$ is divided by a well defined uniform grid with spacing h. As depicted in Fig. 2.1, let us assume that a node is placed at each grid point. The second partial derivatives of u with respect to x and y are approximated by the following finite differences:

$$\frac{\partial^2}{\partial x^2} u(x_0, y_0) \simeq \frac{u(x_0 - h, y_0) - 2u(x_0, y_0) + u(x_0 + h, y_0)}{h^2} \tag{2.1}$$

$$\frac{\partial^2}{\partial y^2} u(x_0, y_0) \simeq \frac{u(x_0, y_0 - h) - 2u(x_0, y_0) + u(x_0, y_0 + h)}{h^2} \tag{2.2}$$

Note that $x_0 + h = x_1$ and $x_0 - h = x_3$. Substituting these approximates in the Poisson equation $\frac{\partial^2}{\partial x^2}u + \frac{\partial^2}{\partial y^2}u = -\rho$ we get

$$4u(x_0, y_0) - \big(u(x_1, y_1) + u(x_2, y_2) + u(x_3, y_3) + u(x_4, y_4)\big) = h^2\rho(x_0, y_0) \tag{2.3}$$

Applying the Neumann boundary conditions in this simple scenario is also very straightforward [12]. Assuming node 0 is on a boundary parallel to the y-axis, node 1 becomes an imaginary node and the boundary condition $\frac{\partial u}{\partial n} = 0$ adds the constraint $\frac{\partial u}{\partial x} = \frac{u(x_1,y_1)-u(x_3,y_3)}{h} = 0$, or $u(x_1, y_1) = u(x_3, y_3)$.

This procedure can be repeated for all of the nodes in the interior of the network and on the boundary to get an equation for each node potential. By giving an index to each of the nodes, these equations can be written in matrix form:

$$AU = h^2P \tag{2.4}$$

Here, U and P are vectors containing the potential and traffic activity of all of the nodes in the network arranged by their index. A is a matrix with block structure and it contains the 4 and -1 coefficients corresponding to each equation.

This matrix equation can be solved with well known fast and efficient methods based on Conjugate Gradient (CG), however we use the Jacobi method, since it directly lends itself to a distributed solution. As a first step, we rearrange (2.3) to get

$$\begin{aligned} u(x_0, y_0) \simeq \frac{1}{4}\Big(u(x_1, y_1) + u(x_2, y_2) + u(x_3, y_3) + u(x_4, y_4)\Big) \\ + \frac{h^2}{4}\rho(x_0, y_0) \end{aligned} \tag{2.5}$$

Assuming (k) denotes the k^{th} iteration, the Jacobi method for solving (2.5) involves iterations of the form

$$u(x_0, y_0)^{(k+1)} \simeq \frac{1}{4}\Big(u(x_1, y_1)^{(k)} + u(x_2, y_2)^{(k)} + u(x_3, y_3)^{(k)} + u(x_4, y_4)^{(k)}\Big) + \frac{h^2}{4}\rho(x_0, y_0) \tag{2.6}$$

i.e., each node should iteratively approximate its potential by averaging the potentials of its neighbors and adding a bias proportional to ρ. The approximation improves by increasing the number of iterations. The convergence of the Jacobi method for an FDM mesh is proved in [23].

2.2 Solution for a Randomly Distributed Network with Uniform Distribution

While the FDM method discussed in the previous section is distributed, it can not be used in our problem, since the nodes are distributed randomly, not on a grid. We extend this method to deal with a random node deployment, where nodes are distributed uniformly and independently through the region $\mathcal{A}$. By the assumption of uniform distribution, the probability that a node position z falls in region $\mathcal{B}$ is $\mathbb{P}(z \in \mathcal{B}) = \frac{|\mathcal{B}|}{|\mathcal{A}|}$. The independence assumption yields to $\mathbb{P}(\cap_{i=1}^{N} z_i \in \mathcal{B}_i) = \Pi_{i=1}^{N}\mathbb{P}(z_i \in \mathcal{B}_i)$.

In the first step we generalize (2.5) for the case that the nodes are uniformly (with equal space) placed on a circle with radius r.

Lemma 2.1. *Let u be the solution of* (1.5)*, then the value of u at each point is*

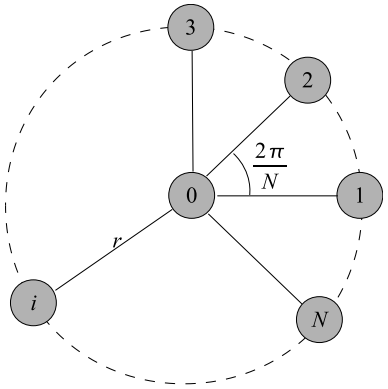

Figure 2.2: Nodes $1 \cdots N$ are arranged on a circular pattern. This configuration will be used as a starting point for the situation where the nodes are randomly distributed in the region $\mathcal{A}$ with uniform distribution.

the average of u around a circle plus a bias proportional to ρ at that point, i.e., the value of u at $\vec{r}_0$ is

$$u(\vec{r}_0) \simeq \int_0^{2\pi} u(r,\phi)\frac{1}{2\pi}d\phi + \frac{r^2}{4}\rho(\vec{r}_0) \tag{2.7}$$

Proof. Assume node 0 is at position $\vec{r}_0$, and nodes $1 \ldots N$ are positioned with uniform spacing around a circle with center at $\vec{r}_0$ and radius r. Thus, the nodes divide the circle to arcs of length $\frac{2\pi}{N}r$. Assume the center of a polar coordinate system is placed at $\vec{r}_0$ and the polar axis is parallel to the x axis. The positions of nodes $1, \ldots, N$ in Cartesian and polar coordinates are $\vec{r}_i = \vec{r}_0 + r(\cos\phi_i, \sin\phi_i)$ and $\vec{r}_i = (r, \phi_i)$, respectively, where $\phi_i = \frac{2\pi}{N}(i-1)$. This configuration is depicted in Fig. 2.2.

In Cartesian coordinates, using Taylor's expansion for u and keeping up to

second order terms, we have

$$\begin{aligned} u(\vec{r}_i) &\simeq u(\vec{r}_0) + (\vec{r}_i - \vec{r}_0)^T \nabla u(\vec{r}_0) \\ &\quad + \frac{1}{2}(\vec{r}_i - \vec{r}_0)^T H(\vec{r}_0)(\vec{r}_i - \vec{r}_0) \end{aligned} \tag{2.8}$$

where $H(\vec{r}_0)$ is the Hessian matrix of u at position $\vec{r}_0$. Substituting $\vec{r}_i - \vec{r}_0$ by $r(\cos\phi_i, \sin\phi_i)$ in (2.8) yields

$$\begin{aligned} u(\vec{r}_i) \simeq & u(\vec{r}_0) + r \begin{bmatrix} \cos\phi_i \sin_i \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial u}{\partial y} \end{bmatrix} \\ & + \frac{1}{2} r^2 \begin{bmatrix} \cos\phi_i \sin_i \end{bmatrix} \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} & \frac{\partial^2 u}{\partial x \partial y} \\ \\ \frac{\partial^2 u}{\partial x \partial y} & \frac{\partial^2 u}{\partial y^2} \end{bmatrix} \begin{bmatrix} \cos\phi_i \\ \\ \sin_i \end{bmatrix} \end{aligned}$$

or

$$\begin{aligned} u(\vec{r}_i) \simeq & \, u(\vec{r}_0) + r\left(\frac{\partial}{\partial x} u(\vec{r}_0) \cos\phi_i + \frac{\partial}{\partial y} u(\vec{r}_0) \sin\phi_i \right) \\ & + \frac{1}{2} r^2 \left(\frac{\partial^2}{\partial x^2} u(\vec{r}_0) \cos^2\phi_i + 2 \frac{\partial^2}{\partial x \partial y} u(\vec{r}_0) \sin\phi_i \cos\phi_i \right. \\ & \left. + \frac{\partial^2}{\partial y^2} u(\vec{r}_0) \sin^2\phi_i \right) \end{aligned} \tag{2.9}$$

We sum $u(\vec{r}_i)$ over $i = 1, \ldots, N$ and use identities

$$\sum_{i=1}^{N} \cos\phi_i = \sum_{i=1}^{N} \sin\phi_i = 0 \tag{2.10}$$

$$\sum_{i=1}^{N} \cos^2\phi_i = \sum_{i=1}^{N} \sin^2\phi_i = \frac{N}{2} \tag{2.11}$$

$$\sum_{i=1}^{N} \cos\phi_i \sin\phi_i = 0 \tag{2.12}$$

which results in:

$$\sum_{i=1}^{N} u(\vec{r}_i) \simeq N u(\vec{r}_0) + N \frac{r^2}{4} \left(\frac{\partial^2}{\partial x^2} u(\vec{r}_0) + \frac{\partial^2}{\partial y^2} u(\vec{r}_0) \right) \tag{2.13}$$

But $\frac{\partial^2}{\partial x^2}u(\vec{r}_0) + \frac{\partial^2}{\partial y^2}u(\vec{r}_0) = -\rho(\vec{r}_0)$, thus we have

$$u(\vec{r}_0) \simeq \frac{1}{N}\sum_{i=1}^{N} u(\vec{r}_i) + \frac{r^2}{4}\rho(\vec{r}_0) \tag{2.14}$$

Let $N \uparrow \infty$, the Riemann sum converges to an integral, thus equation (2.14) would approach

$$u(\vec{r}_0) \simeq \int_0^{2\pi} u(r,\phi)\frac{1}{2\pi}d\phi + \frac{r^2}{4}\rho(\vec{r}_0) \tag{2.15}$$

In this equation $u(r,\phi)$ is the value of u at (r,ϕ) in polar coordinates. □

Up to here we have assumed that the nodes $1,\ldots,N$ were distributed deterministically around node 0 on a circle. Now assume that all of the nodes in the network are distributed uniformly and independently in the area $\mathcal{A}$ and that the communication range of each node is R. Furthermore, assume that node 0 is located at $\vec{r}_0$ inside the network far from the boundary (with distance more than R) and it has $M+1$ neighbors in its communication range. These neighboring nodes are located at $(\mathbf{r},\mathbf{\Phi})$ and $(\mathbf{r}_i,\mathbf{\Phi}_i)$ $i = 1,\ldots,M$. It is easy to show that random variables $\mathbf{r}$, $\mathbf{r}_i$, $\mathbf{\Phi}$ and $\mathbf{\Phi}_i$ are independent for $i = 1,\ldots,M$. Furthermore, $\mathbf{\Phi}$ and $\mathbf{\Phi}_i$ $i = 1,\ldots,M$ are uniformly distributed in the interval $[0,2\pi)$, therefore, they are i.i.d. The random variables $\mathbf{r}$, $\mathbf{r}_i$ $i = 1,\ldots,M$ are also i.i.d. Let $g_R(r)$ denote the distribution of $\mathbf{r}$. Since all of the neighboring nodes are in the communication range, $g_R(r)$ is zero for $r > R$. The nodes are distributed uniformly, thus

$$\mathbb{P}_R(\mathbf{r} \le r | \mathbf{r} \le R) = \frac{\pi r^2}{\pi R^2} \quad r \le R$$

thus

$$g_R(r) = 2\frac{r}{R^2} \quad r \le R \tag{2.16}$$

and

$$g_R(r) = 0 \quad r > R \tag{2.17}$$

Lemma 2.2. *Let the nodes be distributed uniformly and independently in region $\mathcal{A}$. Furthermore, let u be the solution of (1.5), then the value of u at each point, with distance R or more from the boundary, is the average of the value of u for the nodes in the communication range plus a constant bias proportional to the value of ρ at that point.*

Proof. Using (2.15), we will evaluate $\mathbb{E}[u(\mathbf{r}, \mathbf{\Phi})]$ ($\mathbb{E}[\mathbf{x}]$ denotes mathematical expectation of random variable $\mathbf{x}$):

$$\begin{aligned}
\mathbb{E}[u(\mathbf{r}, \mathbf{\Phi})] &= \int_0^R \int_0^{2\pi} u(r, \phi) \frac{g_R(r)}{2\pi} dr d\phi \\
&= \int_0^R g_R(r) dr \int_0^{2\pi} u(r, \phi) \frac{1}{2\pi} d\phi \\
&= \int_0^R \big(u(\vec{r}_0) - \frac{r^2}{4} \rho(\vec{r}_0)\big) g_R(r) dr \\
&= u(\vec{r}_0) - \frac{\mathbb{E}[\mathbf{r}^2]}{4} \rho(\vec{r}_0)
\end{aligned} \tag{2.18}$$

where we have used the fact that $\int_0^R g_R(r)dr = 1$ and $\int_0^R r^2 g_R(r)dr = \mathbb{E}[\mathbf{r^2}]$. On the other hand, note that since $(\mathbf{r}_i, \mathbf{\Phi}_i)$ forms an i.i.d sequence, by strong law of large numbers we have

$$\frac{1}{M} \sum_{i=1}^{M} u(\mathbf{r}_i, \mathbf{\Phi}_i) \xrightarrow{a.s.} \mathbb{E}[u(\mathbf{r}, \mathbf{\Phi})] \quad M \uparrow \infty \tag{2.19}$$

For large M we can write:

$$u(\vec{r}_0) \simeq \frac{1}{M}\sum_{i=1}^{M} u(\mathbf{r}_i, \mathbf{\Phi}_i) + \frac{\mathbb{E}[\mathbf{r}^2]}{4}\rho(\vec{r}_0) \tag{2.20}$$

where from (2.16), $\mathbb{E}[\mathbf{r}^2] = \frac{R^2}{4}$. Equation (2.20) states that a node located at $\vec{r}_0$ can estimate its potential simply by averaging the potentials of its neighboring nodes and adding a bias proportional to $\rho(\vec{r}_0)$. Like (2.5), after several iterations, this estimation become more accurate. □

Equation (2.20) is well suited for distributed calculation of potential function u, since each node estimates its potential only based on the potential of its neighboring nodes. Each node in the network broadcasts its potential to its neighbors, and collects the potentials received from them.

2.3 Boundary Conditions

In the derivation of (2.20), we had two important assumptions: First, $\mathbf{\Phi}$ is uniformly distributed over $[0, 2\pi)$, and second $\mathbf{r}$ and $\mathbf{\Phi}$ are independent. Due to symmetry of the uniform node distribution, these conditions are satisfied when node 0 is far from the boundary or if the node is exactly on the boundary. However, when the distance of node 0 form the boundary is less than R (the communication range) these assumptions do not hold. We call these nodes boundary nodes.

For the nodes exactly on the boundary, imposing Neumann boundary condition on these nodes should proceed along lines similar to those of the uniform grid case. Let us assume the boundary is smooth enough so its curvature at every point is much larger than R, thus every part of the boundary with length $2R$ can be approximated

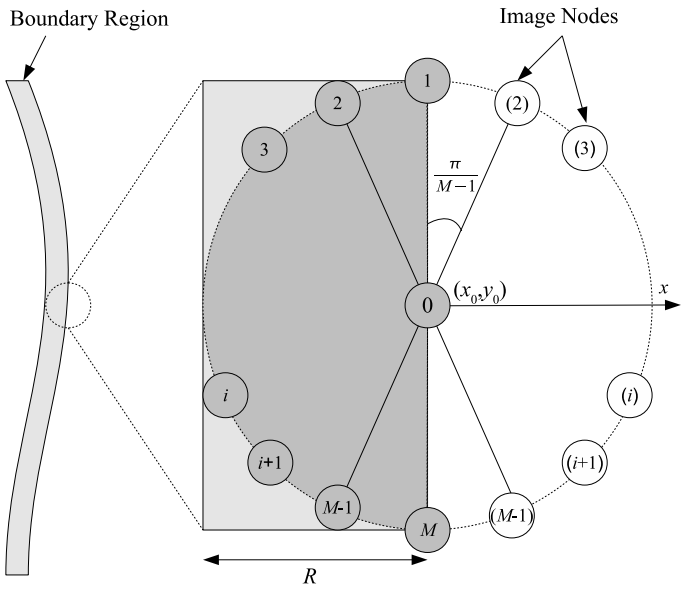

Figure 2.3: Image nodes on the boundary of region $\mathcal{A}$.

by a straight line, as depicted in figure 2.3. We rewrite Lemma 2.1 to deal with the situation that a node is on the boundary. We have arbitrarily assigned a Cartesian axes, where x is perpendicular to the boundary. In this figure, node 0 is located at $\vec{r}_0$, and nodes $1, \ldots, M$ are inside the network, near the boundary. Nodes $(2), \ldots, (M-1)$ are image nodes with respect to the boundary, with their potentials denoted by $u_{(i)}$.

Note that

$$\frac{u_{(M-i+1)} - u_i}{2R} \simeq \nabla u \cdot (\cos\phi_{(M-i+1)}, \sin\phi_{(M-i+1)}) \tag{2.21}$$

Due to symmetry with respect to $\vec{r}_0$ we have

$$\begin{aligned} \sum_{i=2}^{M-1} \frac{u_{(M-i+1)} - u_i}{2R} &\simeq \nabla u \cdot (\sum_{i=2}^{M-1} \cos\phi_{(i)}, \sum_{i=2}^{M-1} \sin\phi_{(i)}) \\ &\simeq c\frac{\partial u}{\partial x} \end{aligned} \tag{2.22}$$

with $c > 0$. The Neumann boundary condition for point $\vec{r_0}$ imposes that $\frac{\partial u}{\partial x} = 0$ which yields to

$$\sum_{i=2}^{M-1} u_{(i)} = \sum_{i=2}^{M-1} u_i \tag{2.23}$$

A sufficient condition for this to hold is $u_{(i)} = u_i$ for $i = 2, \ldots, M-1$, i.e., the potential of image node (i) should be equal to that of node i. Following the steps we took to prove Lemma 2.1, we have

$$u(\vec{r_0}) \simeq \frac{1}{M}\left(u_1 + u_M + \sum_{i=2}^{M-1}\left(u_i + u_{(i)}\right)\right) + \frac{r^2}{4}\rho(\vec{r_0}) \tag{2.24}$$

$$= \frac{1}{M}\left(u_1 + u_M + \sum_{i=2}^{M-1} 2u_i\right) + \frac{r^2}{4}\rho(\vec{r_0}) \tag{2.25}$$

Similar to the previous case, as M grows large $\frac{u_1+u_M}{M} \to 0$ and Riemann sum converges to an integral

$$u(\vec{r_0}) \;\simeq\; \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} u(r,\phi)\frac{1}{\pi}d\phi + \frac{r^2}{4}\rho(\vec{r_0}) \tag{2.26}$$

Assume the nodes distributed uniformly, and node 0 has $M+1$ neighbors in its communication range. These neighboring nodes are located at $(\mathbf{r}, \mathbf{\Phi})$ and $(\mathbf{r}_i, \mathbf{\Phi}_i)$, $\quad i = 1, \ldots, M$. Due to symmetry, random variables $\mathbf{r}$, $\mathbf{r}_i$, $\mathbf{\Phi}$ and $\mathbf{\Phi}_i$ are independent for $i = 1, \ldots, M$, as in the previous case. However, $\mathbf{\Phi}$ and $\mathbf{\Phi}_i$ $\;i = 1, \ldots, M$ are uniformly distributed in the interval $[-\frac{\pi}{2}, \frac{\pi}{2})$. The random variables $\mathbf{r}$, $\mathbf{r}_i$ $\;i = 1, \ldots, M$ are also i.i.d. Following the steps taken to prove Lemma 2.2 and by using (2.26), we can show that for a node exactly on the boundary the value of

u, is the average of the value of u for nodes in communication range plus a constant bias.

So far we have shown that the averaging technique works for estimating the potential value both for nodes that are in the interior of the network and for the nodes on the boundary. In order to use the scheme for all nodes, the remaining work is to extend the method for the nodes that are not on exactly on the boundary nor in the interior of the network. We refer to these nodes to semi-boundary nodes. The distance of semi-boundary nodes from the boundary is nonzero but it is smaller than R. Unfortunately, using the image technique (as presented for the boundary nodes) will not help to generalize the averaging technique to semi-boundary nodes due to lack of symmetry in the geometrical properties of the image nodes. One useful fact is that the potential function is generally continuous. Note that the potential function varies continuously (and often smoothly) while transitioning from boundary nodes to the interior of the network. This fact helps use the conjecture that since the averaging technique works for boundary nodes and interior nodes, therefore, it must work for the nodes in between; hence averaging technique can be used for semi-boundary nodes. We have verified validity of this conjecture through some numerical examples discussed in the next chapter.

In the random node distribution the density of the nodes should be high enough for two reasons: First increasing the node density increases the number of neighbors in the communication range of each node, which would increase the accuracy of the ML estimator, as suggested by (2.19), and second, the density of nodes should be high enough to guaranty connectivity of the network and implementability of the

information flow lines. In [11] it is shown that if the total traffic in the network is θ bps, density of the nodes should be $O\big(\theta^2 log\theta\big)$ in order to guarantee these conditions.

Chapter 3

Simulation Results

In this chapter we present two examples to test the proposed algorithm for approximating potential function.

Example 3.1. In the first example, we scattered 1000 nodes in a unit square. The communication range of each node is 0.1. A single source is located at $(0.1, 0.5)$ and a sink is placed at $(0.9, 0.5)$. Other nodes act as relays to pass the traffic from source to sink. In order to compute the potential function u, each node first finds the neighboring nodes in its communication range and then uses (2.20) to calculate its potential. This process is carried on for 10000 iterations to increase the accuracy of the potential. Note that after much smaller number of iterations the information flow paths converge and remain almost unchanged.

Figure 3.1 depicts the placement of the nodes. The nodes are distributed uniformly and independently in the area $\mathcal{A} = (0\ 1) \times (0\ 1)$. Figure 3.2 depicts the calculated potential, where the potential is interpolated on a grid and then plotted. Figure 3.3 shows the potential function when calculated by MATLAB ® function *assempde*, where the potentials are calculated by finite element method on a mesh. It is observable that the two graph follow the same pattern: a peak at the source and a smooth slope to the sink. Note that the source causes the potential of neighboring nodes to increase while the sink causes this potential to decrease. After

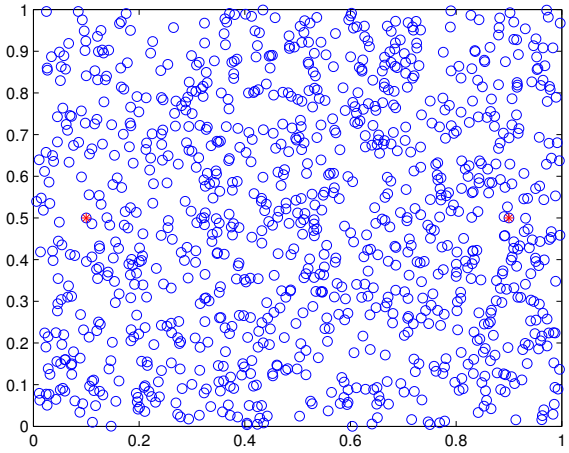

Figure 3.1: The position of nodes in the simulated network. The nodes are distributed uniformly and independently in area $\mathcal{A} = (0\ 1) \times (0\ 1)$.

calculating the potentials, when a node needs to forward a packet, it searches among its neighbors and sends its packet to the nearest neighbor with least potential. The slope from source to sink in the potential function guarantees that the packets will be delivered to the sink.

Example 3.2. In this example we derive information flow lines from the potential function and use it for routing packets from a source to a sink. We simulated a network placed on a uniform grid using the method presented earlier in equation (2.5). The nodes are placed on a uniform grid so that the similarity of the flow lines with the electric fields in an electrostatic setting is presented clearly. In the source, a Poisson process generates approximately 12000 packets within 20000 of simulation cycles. The calculated potential (by Jacobi method) is used to route the packets from the source to the sink in the direction of the decent of the potential. In order to utilize all of the nodes in the network and avoid energy depletion in the nodes residing on a flow path, each node passes traffic with the highest probability

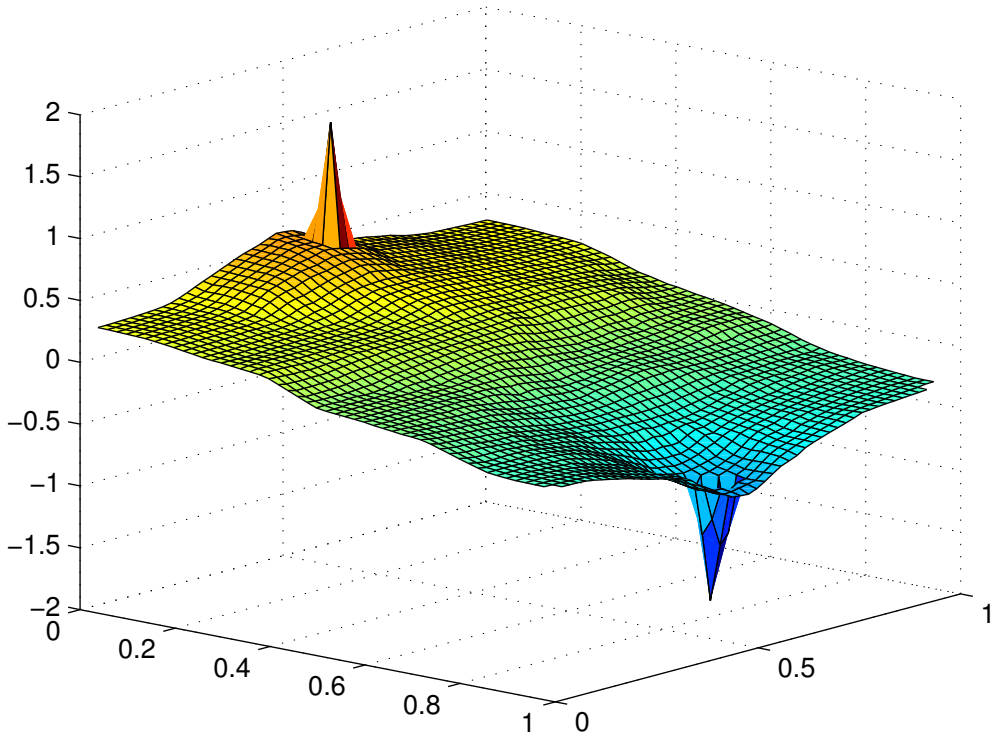

Figure 3.2: The computed potentials after 10000 Jacobi iterations (2.20). Note that the Neumann boundary condition is implicitly applied by using (2.20) for all nodes regardless of their location.

to a neighbor with minimum potential, but also passes traffic to other neighbors at lower potential with less probability (the probability is proportional to potential difference between the sender and the receiver). In order to compute the traffic flow, we assumed that transmission of a packet from a node to another neighboring node accounts for 1 unit of traffic moving in direction of a vector starting from the sending node and ending to the receiving node. The resulted vector is accumulated over the simulation period and then normalized. Figure 3.4 depicts the traffic flow in the vicinity of the source and sink.

Figure 3.5 shows the traffic near the boundary. We can observe that the flow

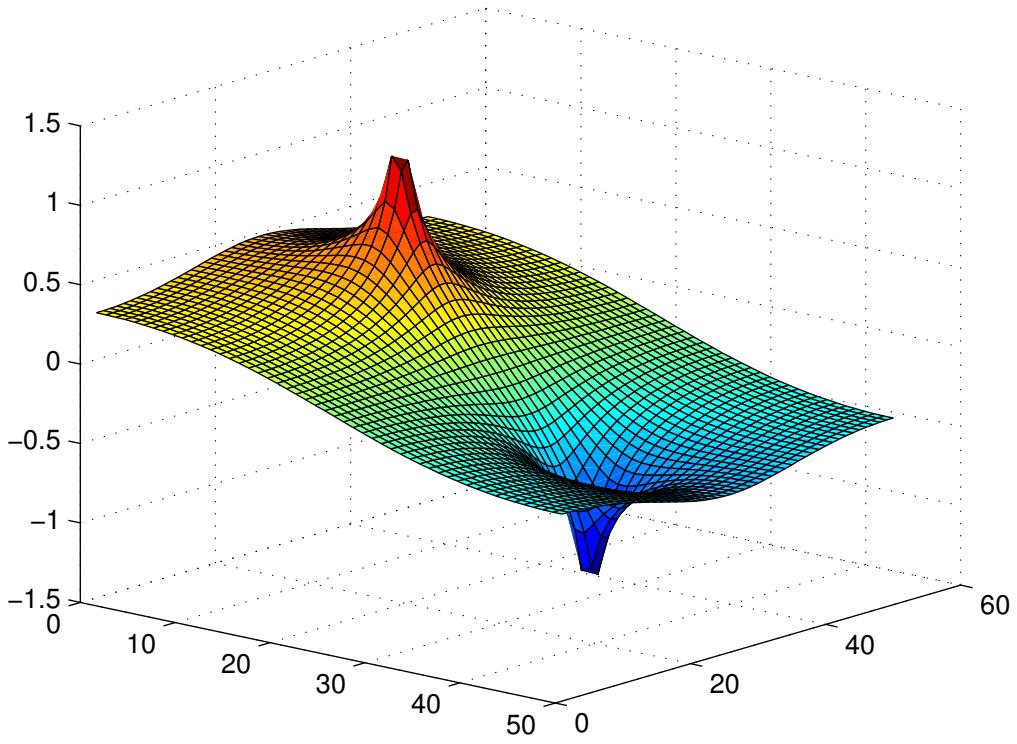

Figure 3.3: The potential computed using MATLAB assempde function.

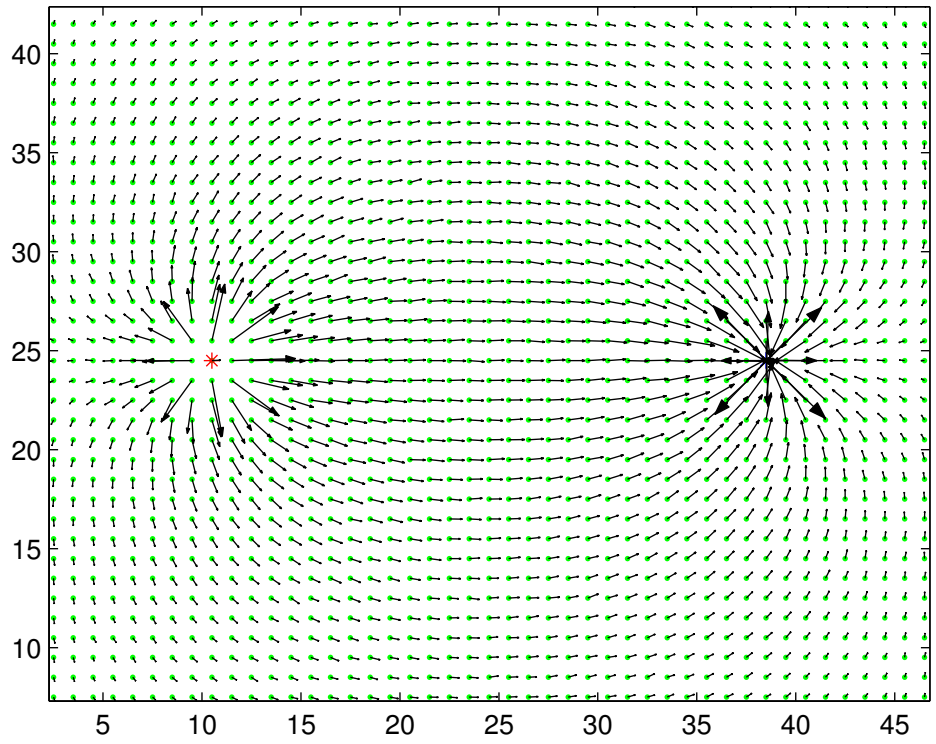

Figure 3.4: The flow of traffic from source to sink. The source generates 12000 packets in 20000 simulation cycles. Similarity of the flow lines with the electric fields in an electrostatic setting is due to the fact that in both settings a Poisson PDE is solved in the domain.

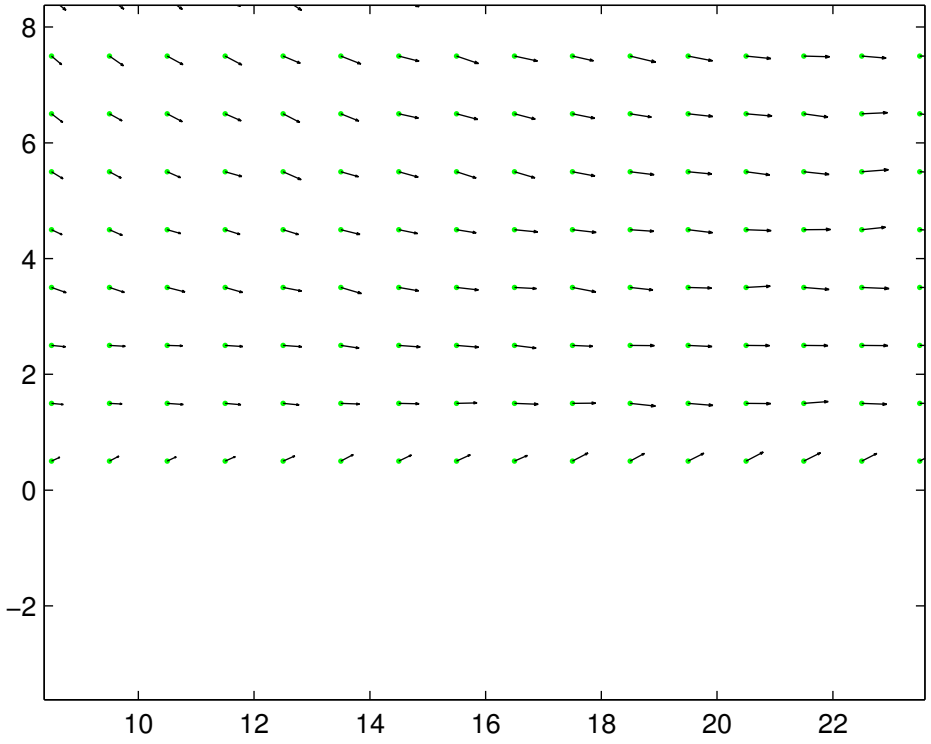

Figure 3.5: The traffic near the boundary of the network. The flow lines are approximately parallel to the boundary.

is approximately parallel to the boundary which is required by Neumann boundary condition.

Let us assume that each transmission causes 1 unit of energy to be consumed in the transmitting node. Figure 3.6 depicts the energy consumption of the network over the period of simulation. It can be seen that the energy consumption is higher in the nodes near the source and the sink. Thus these nodes require higher initial energy, or the density of the nodes near the source and the sink should be higher.

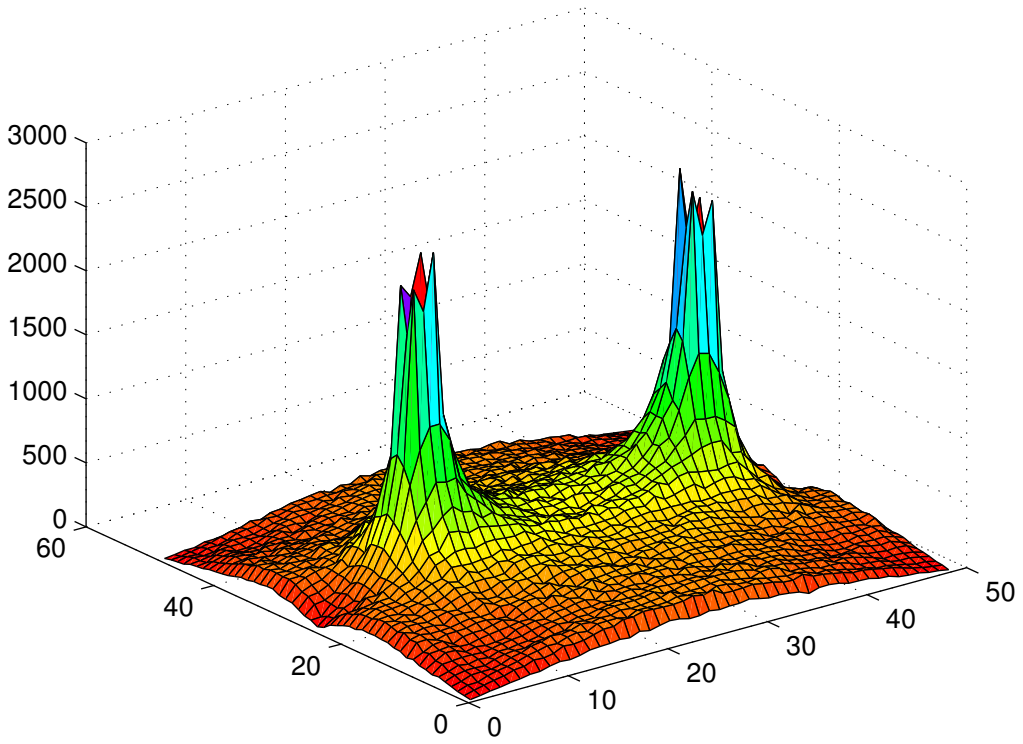

Figure 3.6: Energy consumption in the nodes during the simulation.

Chapter 4

Network Model in Discrete Space

Balancing traffic load within a network has been studied extensively. It is well known that assigning flow along the shortest path causes congestion in bottleneck links, while other links are not utilized properly[6]. In order to avoid congestion, some methods assign multiple paths from sources to destinations[9, 21]. The effect of multipath routing on load balancing have been studied previously, where it is shown that while in a wired network using multipath routing is beneficial, in a wireless setting it has negligible effect on load balancing and congestion, unless the number of paths is very large[10]. This has an important consequence since load balancing and congestion control are key factors that determine quality of service and delay.

In this chapter we study the problem of flow assignment in a general framework. We apply convex optimization techniques and present a *distributed* algorithm to assign flows to the links such that a convex cost function, namely the p-norm of network flow, is optimized. This optimization results in the most balanced network flow allocation. Furthermore, in a network with a given configuration and channel capacities, we show that if it is feasible to route the traffic of a set of sources to the sinks without congestion, then the proposed flow assignment policy achieves it. The distributed nature of the proposed method is an important factor that makes it suitable for practical networks. We study the convergence rate of the proposed

method, introduce a method based on least squared error estimation to increases the convergence rate and compare it with other well-known techniques for accelerating distributed iterations. This is particularly important in battery powered wireless networks where the energy cost of communication is high.

A Wireless Sensor Network (WSN) [28] is an example of a network that both load balancing and reducing the energy cost overhead of routing have direct effect on the lifespan of the network. In these networks, the nodes gather environmental information and relay the information back to a single or multiple data collection units for further processing. Since usually the sinks are outside of the communication range of the sources, the traffic should be relayed through other nodes that are closer to sinks. The wireless devices are usually battery powered, thus large traffic load on a specific path depletes the batteries of the nodes on that path, which reduces the reliability and lifespan of the network.

In our network model, we assume that multiple nodes are connected by communication links with different capacities, and a set of sources generate a *single commodity* traffic that should be routed to a set of sinks (i.e., there is no priority or distinction between the sinks). Furthermore, we assume flow conservation law holds, i.e., the outgoing traffic form a node is equal to the sum of incoming traffic and traffic generated at the node. The network is represented by a graph, where a vertex models a node and a weighted edge models the links. The weight of an edge designates the link capacity. We employ the Sequential Quadratic Programming (SQP) method and propose a distributed method to allocate the flows on the links, such that a specific function is minimized. SQP reduces the minimization problem

to iterations of constrained quadratic programming and is known for stability and fast convergence[5]. The quadratic programming problem results in a set of linear equations, that are solved through distributed methods. After several iterations in which only neighboring nodes are involved, an approximation of the global optimal flow is achieved. This scheme can be used to find distributed methods to optimize any general separable convex cost function; however, in this work we apply it to the p-norm cost function. In the case of $p = 2$, the optimum flow resembles the current flow that passes through an electric network. As p grows large, the flow generated by the sources is redistributed more evenly in the network, and as $p \uparrow \infty$ the optimal flow approaches to the solution of the minimax problem.

A drawback of this method is that the linear equations resulted from SQP are solved using Jacobi method[23], which is known to converge slowly to the final optimal solution. Large number of iterations result in unacceptable routing overhead an energy consumption. However, we show that the Jacobi method converges to the final solution with a time constant that is proportional to the eigenvalues of a specific matrix. This matrix has a dominant eigenvalue that can be estimated from a small number of observations based on non-linear Least Squared Error estimation method.

4.1 Network Model

A single commodity network is modeled by a directed graph $G(V, E, c)$, where the set of vertices, V, models the nodes, the set of edges, $E \subseteq V \times V$, models the communication links and a weight function, $c : E \to \mathbb{R}^+$, represents the channel

capacities of the links. We are assuming that the graph G has no self-loops[1] and it has at most one edge between every pair of nodes. Let $|V| = n$ and $|E| = m$, i.e., the network is composed of n nodes and m links. Since G is assumed to be connected, we have $m \geq n - 1$. We denote a link that starts from node v_i and ends to v_j by (v_i, v_j). We define a function $I : E \to \{1, 2, \ldots, m\}$ that assigns a unique index to each link. Furthermore, for each link we define a real valued flow that models the network traffic on the link, and denote the flow on a link with index k by f_k. Note that the links are assumed to be able to pass the flow in both directions; the direction of an edge is merely used as a reference for the direction of the flow. For link k, if $f_k > 0$ then the traffic flow is along the direction of the edge, whereas $f_k < 0$ means that the traffic flows in the opposite direction of the edge.

The traffic generated in the network is collected by a set of special nodes, namely, sinks. Let us denote the set of all sinks by $\mathcal{D}$. The set of other nodes that are not collecting traffic is denoted by $\mathcal{S} = V \backslash \mathcal{D}$, with $|\mathcal{S}| = s$. Furthermore, without loss of generality, let us assume $\mathcal{S} = \{v_1, \ldots, v_s\}$. For each node $v_i \in \mathcal{S}$ we define a non-negative value b_i that represents the amount of traffic that is injected into the network by that node. Notice that $b_i = 0$ indicates that v_i is not injecting any traffic into the network, thus v_i only contributes in relaying the traffic of other sources to the sinks.

In our network model, we assume that flow is conserved, i.e., the outgoing traffic flow from a node is equal to the sum of incoming traffic from other nodes and

[1] A self-loop is an edge between a node and itself.

the traffic generated by the node itself. Thus for every node $v_i \in \mathcal{S}$ we have

$$\sum_{(v_i,v_j)\in E} f_{I(v_i,v_j)} - \sum_{(v_j,v_i)\in E} f_{I(v_j,v_i)} = b_i \quad \forall\ v_i \in \mathcal{S} \tag{4.1}$$

Using the definition of the incidence matrix of graph G, we rewrite this system of s linear equations in matrix form. Recall that the incidence matrix, $\hat{\mathbf{K}}$ (an $n \times m$ matrix), of a directed graph is defined as

$$\hat{\mathbf{K}}_{i,k} = \begin{cases} 1 & \exists v_j \in V \ s.t.\ I(v_i, v_j) = k \\ -1 & \exists v_j \in V \ s.t.\ I(v_j, v_i) = k \\ 0 & o.w. \end{cases} \tag{4.2}$$

In a general graph, $\hat{\mathbf{K}}\mathbf{f} = \hat{\mathbf{b}}$ expresses the flow conservation in the network, where $\hat{\mathbf{b}} = [b_1, \ldots, b_n]$ is the source vector for all n nodes. However, in the network, there are sink (destination) nodes, such that they can gather all of the flow they receive from their incoming links. In our problem there are $|\mathcal{D}|$ sinks that gather the flow, thus $|\mathcal{D}|$ of flow equations expressed by $\hat{\mathbf{K}}\mathbf{f} = \hat{\mathbf{b}}$ corresponding to the sink nodes are redundant.

Let us eliminate the rows of $\hat{\mathbf{K}}$ that correspond to sink nodes (rows $s+1, \ldots, n$) and denote the resulting $s \times m$ matrix by $\mathbf{K}$, which is a matrix with rank s. Then the flow conservation equations are written as

$$\mathbf{K}\mathbf{f} = \mathbf{b} \tag{4.3}$$

where $\mathbf{f} = [f_1, \ldots, f_m]^T$ is the flow vector and $\mathbf{b} = [b_1, \ldots, b_s]^T$ is the source vector for nodes in $\mathcal{S}$.

4.2 The Minimax Network Flow Optimiaztion

Except for trivial networks[2], equation (4.3) does not define the flow $\mathbf{f}$ uniquely. This opens a room to find an optimum flow that minimizes a suitable cost function. Let us denote the set of all $\mathbf{f}$'s that satisfy (4.3) by $\mathcal{F}(\mathbf{b})$. Furthermore, let us denote the set of all *feasible* source vectors, $\mathbf{b}$, by $\mathcal{B}$. A feasible source vector, $\mathbf{b}$, is one which has the property that at least there exists one flow vector $\mathbf{f} \in \mathcal{F}(\mathbf{b})$ such that it satisfies the channel capacity constraints $|f_k| \leq c_k,\ k = 1, \ldots, m$, where c_k is the channel capacity of k^{th} link. In our previous works[16, 29], we defined the p-norm cost function, J_p, and proposed a method to find its unique solution to (4.3) by solving the following convex optimization problem

$$\begin{aligned} \underset{\mathbf{f}\in\mathbb{R}^m}{\text{minimize}} \quad & J_p(\mathbf{f}) = \sum_{k=1}^{m} \left(\frac{|f_k|}{c_k} \right)^p \\ \text{subject to} \quad & \mathbf{Kf} = \mathbf{b}. \end{aligned} \tag{4.4}$$

As $p \uparrow \infty$, the unique solution of optimization problem (4.4) approaches to one of the solutions of the *minimax* flow optimization

$$\begin{aligned} \underset{\mathbf{f}\in\mathbb{R}^m}{\text{minimize}} \quad & \max\{\frac{|f_k|}{c_k}, k = 1 \cdots m\} \\ \text{subject to} \quad & \mathbf{Kf} = \mathbf{b}. \end{aligned} \tag{4.5}$$

Let us denote the solution of (4.4) by $\mathbf{f}^{(p)*}$ and a solution of (4.5) by $\mathbf{f}^{(\infty)*}$. We proved that the solution of problem (4.5) has the following properties.

Lemma 4.1. *If* $\exists k \in \{1 \cdots m\}$ *s.t.* $|f_k^{(\infty)*}| > c_k$, *then* $\forall \mathbf{f} \in \mathcal{F}(\mathbf{b}),\ \exists \ell \in \{1 \cdots m\}$ *s.t.* $|f_\ell| > c_\ell$.

[2] An example is a network with one sink, with $m = n - 1$ which makes its graph form a tree. In this case $\mathbf{K}$ becomes an $(n-1) \times (n-1)$ square matrix.

Lemma 4.1 states that if a source vector is infeasible for the minimax problem (i.e., there exists a link such that the optimal minimax flow exceeds the capacity of the link), then it is also infeasible for every other flow allocation methods. Thus the minimax flow assignment policy results in the largest set of feasible source vectors, with its feasible set equal to $\mathcal{B}$.

Proof. If $\exists k \in \{1 \cdots m\}$ s.t. $|f_k^{(\infty)*}| > c_k$, then $\max_{l\in\{1\cdots m\}} \frac{|f_l^{(\infty)*}|}{c_l} > 1$. Since $\mathbf{f}^{(\infty)*}$ solves the minimax problem, for all $\mathbf{f} \in \mathcal{F}(\mathbf{b})$ we have

$$1 < \max_{l\in\{1\cdots m\}} \frac{|f_l^{(\infty)*}|}{c_l} \leq \max_{l\in\{1\cdots m\}} \frac{|f_l|}{c_l} \tag{4.6}$$

□

Lemma 4.2. *If $\mathbf{f}^{(\infty)*}$ solves the minimax problem (4.5), then it also solves the following problem:*

$$\begin{aligned} &\underset{\mathbf{f}\in\mathbb{R}^m}{\textit{maximize}} \quad \min\{1 - \frac{|f_k|}{c_k}, k = 1 \cdots m\} \\ &\textit{subject to} \quad \mathbf{Kf} = \mathbf{b} \end{aligned} \tag{4.7}$$

Lemma 4.2 states the load balancing property of minimax flow allocation policy. In $\mathbf{f}^{(\infty)*}$ the traffic flow is distributed such that congestion (which results in packet loss) is avoided in every link as much as possible. Its proof is simply derived from lemma 4.1.

Geometrical interpretation of the p-norm optimization problem (4.4) and minimax optimization problem (4.5) is interesting. Assuming f_k's as independent variables, $\mathbf{Kf} = \mathbf{b}$ defines a hyperplane in an m-dimensional space. Let $y_k = \frac{f_k}{c_k}$, then for $p = 2$ the contours of $\|\mathbf{y}\|_p$ define m-dimensional ellipsoids. As $p \uparrow \infty$ the contours gradually deform into hyperrectangles. The optimization problem is finding

the smallest contour that touches the $\mathbf{Kf} = \mathbf{b}$ hyperplane. The rectangular contours of $\|\mathbf{y}\|_\infty$ do not define a smooth surface, whereas those of $\|\mathbf{y}\|_p$ are smooth, thus approaching the minimax optimization problem as the limiting case of p-norm optimization enables us to employ multivariate calculus and optimization methods for differentiable functions, such as SQP.

Our proposed method to find a distributed solution to (4.5) involves iterations of solving (4.4), where in each iteration p is increased until a stopping criterion is met. In the SQP method, a quadratic approximation of the cost function is calculated around an operating point. Based on the quadratic approximation, a steepest decent direction is calculated and the operating point is updated. Let $\mathbf{f} \in \mathcal{F}(\mathbf{b})$ be an initial operating point and $\mathbf{e}$ be a perturbation around $\mathbf{f}$. For p even we have

$$J_p(\mathbf{f} + \mathbf{e}) = J_p(\mathbf{f}) + \mathbf{e}^T\mathbf{h} + \frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} \tag{4.8}$$

where $\mathbf{h}$ and $\mathbf{Q}$ are the gradient vector and Hessian matrix of J_p at operating point $\mathbf{f}$, respectively. Notice that both $\mathbf{f}$ and $\mathbf{e}$ should satisfy the flow conservation law, thus $\mathbf{Ke} = \mathbf{0}$. The Lagrange function is defined by

$$\mathcal{L}(\mathbf{e}) = J_p(\mathbf{f}) + \mathbf{e}^T\mathbf{h} + \frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} - \mathbf{u}^T\mathbf{K}\mathbf{e} \tag{4.9}$$

where $\mathbf{u}$ is a vector composed of Lagrange multipliers. Using the method of Lagrangian multipliers, the optimum direction of decent is calculated by setting $\nabla_{\mathbf{e}}\mathcal{L} = \mathbf{0}$. We have

$$\mathbf{h} + \mathbf{Q}\mathbf{e} - \mathbf{K}^T\mathbf{u} = \mathbf{0} \tag{4.10}$$

which in conjunction with $\mathbf{Ke} = \mathbf{0}$ yields to

$$\mathbf{KQ}^{-1}\mathbf{K}^T\mathbf{u} = \mathbf{KQ}^{-1}\mathbf{h} \tag{4.11}$$

$$\mathbf{e} = \mathbf{Q}^{-1}(\mathbf{K}^T\mathbf{u} - \mathbf{h}) \tag{4.12}$$

Assuming $\mathbf{Q}$ is an $m \times m$ matrix with rank m, the $s \times s$ matrix $\mathbf{KQ}^{-1}\mathbf{K}^T$ is full rank and invertible since we have rank $(\mathbf{KQ}^{-1}\mathbf{K}^T) = \text{rank}(\mathbf{K})$, and we know that $\mathbf{K}$ is of rank s.

In the case of p-norm cost function $\mathbf{Q}$, the Hessian matrix at operating point $\mathbf{f}$, is

$$\mathbf{Q} = p(p-1)\begin{bmatrix} f_1^{p-2} & & & \\ & f_2^{p-2} & \mathbf{0} & \\ & \mathbf{0} & \ddots & \\ & & & f_m^{p-2} \end{bmatrix}_{m\times m} \tag{4.13}$$

For $\mathbf{h}$, the gradient vector, we have

$$\mathbf{h} = p\begin{bmatrix} f_1^{p-1} \\ f_2^{p-1} \\ \vdots \\ f_m^{p-1} \end{bmatrix}_{m\times m} \tag{4.14}$$

thus, $\mathbf{KQ}^{-1}\mathbf{h} = \frac{1}{p-1}\mathbf{b}$.

Remark. Since G is a connected graph, the rank of $\hat{\mathbf{K}}$ is $n-1$[8]. This means that eliminating one equation from n equation expressed by $\hat{\mathbf{K}}\mathbf{f} = \mathbf{b}$ results in $n-1$ linearly independent equations. Using rank-nullity theorem, the nullity of $\hat{\mathbf{K}}$ is $m - n + 1$, which is equal to the number of loops in the graph.

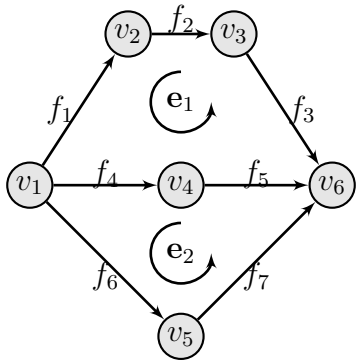

Figure 4.1: The circular flows $\mathbf{e}_1$ and $\mathbf{e}_2$ form a basis for the nullspace of matrix $\mathbf{K}$. Every circular flow $\mathbf{e}$ in the nullspace of $\mathbf{K}$ is a linear combination these basis.

Remark. In the special case of a network with one sink, the nullspaces of $\mathbf{K}$ and $\hat{\mathbf{K}}$ are equal. To show this, notice that nullspace($\mathbf{K}$) = (rowspace($\mathbf{K}$))$^\perp$ and rowspace($\mathbf{K}$) = rowspace($\hat{\mathbf{K}}$), where $\perp$ denotes the orthogonal complement of a subspace. A basis for the nullspace of $\mathbf{K}$ is the set of all circular flows that pass through the loops of the graph. As an example, figure 4.1 depicts a network with two loops. The circular flows $\mathbf{e}_1 = [0, 0, 0, -1, -1, 1, 1]^T$ and $\mathbf{e}_2 = [-1, -1, -1, 1, 1, 0, 0]^T$ are basis for the nullspace of $\mathbf{K}$. Every circular flow $\mathbf{e}$ in the nullspace of $\mathbf{K}$ is a linear combination of these basis. In this setting, starting from an initial flow (the operating point), the SQP iterations find an optimal circular flow, given by (4.12), such that when the circular flow is added to the initial operating point, the p-norm cost function gets closer to its minimum.

Observe that the SQP steps are reduced to solving a system of linear equations expressed by (4.11) and substituting its solution in (4.12). Thus we need to devise a distributed yet computationally efficient method to compute $\mathbf{u}$. The Conjugate Gradient (CG) method is computationally efficient for solving (4.11), but it is not suitable for distributed settings, since in each step the value of u_i and b_i for all of

the nodes is required, whereas in a distributed setting a node only has access to those values for its neighbors. An alternative approach is the *Jacobi* method. Let[3] $\mathbf{L} = \mathbf{K}\mathbf{Q}^{-1}\mathbf{K}^T$, $\mathbf{z} = \frac{1}{p-1}\mathbf{b}$ and $\mathbf{B}$ be a diagonal matrix with its diagonal elements equal to that of $\mathbf{L}$. The Jacobi method for solving $\mathbf{L}\mathbf{u} = \mathbf{z}$ is

$$\mathbf{u}^{(\ell+1)} = (\mathbf{I} - \mathbf{B}^{-1}\mathbf{L})\mathbf{u}^{(\ell)} + \mathbf{B}^{-1}\mathbf{z} \tag{4.15}$$

where $\mathbf{u}^{(\ell)}$ is the approximate solution of $\mathbf{L}\mathbf{u} = \mathbf{z}$ in the ℓ^{th} iteration and $\mathbf{I}$ is an $s \times s$ identity matrix. It is easy to check that the matrix $\mathbf{L}$ is *irreducibly diagonally dominant*. It is proved in [23] that under this condition, the Jacobi method for solving $\mathbf{L}\mathbf{u} = \mathbf{z}$ converges to the exact solution, regardless of the initial value for $\mathbf{u}^{(1)}$. Furthermore, It is straightforward to show that elements of the i^{th} row of $\mathbf{L}$ are non zero only for the neighbors of node u_i. Since $\mathbf{B}$ is defined to be a diagonal matrix, $\mathbf{I} - \mathbf{B}^{-1}\mathbf{L}$ has the same property. This implies that a node v_i updates the value of u_i simply by calculating a weighted average of u_j's of its neighboring nodes and adds a bias proportional to b_i. In each iteration only neighboring nodes need to share their value of u_j, hence the Jacobi method is directly applicable to a distributed setting. However, the convergence rate of Jacobi method, especially compared to that of CG method, is generally very slow.

[3] In graph theory literature $\mathbf{L}$ is referred to as the Laplacian matrix of a weighted graph, because it approximates the Laplacian operator in discrete space.

4.3 Acceleration of Jacobi Iterations

In a network setting where the energy cost of *node-to-node* communication is considerably larger than the energy cost of *in-node* computation (e.g. battery powered wireless sensor networks), it is desirable to devise a distributed algorithm that is more based on in-node calculations rather than node-to-node communications. There are standard approaches to increase the convergence rate of Jacobi method, such as Gauss-Seidel (GS) method and Successive Over Relaxation (SOR)[23]. The drawback of GS approach is that it requires a complex synchronization mechanism in the network. The drawback of SOR is that it is excessively sensitive to network configuration. We propose another approach that is based on Least Squared Error (LSE) estimation.

Let us rewrite (4.15) in the following closed form

$$\begin{aligned} \begin{bmatrix} 1 \\ \mathbf{u}^{(\ell+1)} \end{bmatrix} &= \mathbf{W} \begin{bmatrix} 1 \\ \mathbf{u}^{(\ell)} \end{bmatrix} \\ &= \mathbf{W}^{\ell} \begin{bmatrix} 1 \\ \mathbf{u}^{(1)} \end{bmatrix} \end{aligned} \tag{4.16}$$

where

$$\mathbf{W} = \left[\begin{array}{c|c} 1 & \mathbf{0}_{1\times s} \\ \hline \mathbf{B}^{-1}\mathbf{z} & \mathbf{I}-\mathbf{B}^{-1}\mathbf{L} \end{array}\right] \tag{4.17}$$

The eigenvalues of $\mathbf{W}$ are $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_{s+1}$, where $\lambda_1 = 1$ and $\lambda_2, \ldots, \lambda_{s+1}$ are the eigenvalues of $\mathbf{I} - \mathbf{B}^{-1}\mathbf{L}$. Since the Jacobi method is known to converge in the case of an irreducibly diagonally dominant matrix, we also have

$\lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_{s+1}|$.

Let $\mathbf{W} = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}$ be the eigenvalue decomposition of $\mathbf{W}$. Furthermore, let $\mathbf{t}_1, \ldots, \mathbf{t}_{s+1}$ be the columns of $\mathbf{T}$, $\hat{\mathbf{t}}_1, \ldots, \hat{\mathbf{t}}_{s+1}$ be the rows of $\mathbf{T}^{-1}$, $\mathbf{u}^{(1)} = \mathbf{0}_{s\times 1}$ and $\mathbf{x} = [1, 0, 0, \ldots, 0]^T_{1\times(s+1)}$.

$$\begin{aligned}
\mathbf{W}^\ell \mathbf{x} &= \sum_{i=1}^{s+1} \lambda_i^\ell \langle \hat{\mathbf{t}}_i, \mathbf{x} \rangle \mathbf{t}_i \\
&= \langle \hat{\mathbf{t}}_1, \mathbf{x} \rangle \mathbf{t}_1 + \sum_{i=2}^{s+1} \lambda_i^\ell \langle \hat{\mathbf{t}}_i, \mathbf{x} \rangle \mathbf{t}_i \qquad (4.18) \\
&\approx \langle \hat{\mathbf{t}}_1, \mathbf{x} \rangle \mathbf{t}_1 + \lambda_2^\ell \langle \hat{\mathbf{t}}_2, \mathbf{x} \rangle \mathbf{t}_2 \qquad (4.19)
\end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. As ℓ grows large, the dominant eigenvalue, λ_2, determines the convergence rate of (4.18) and consequently the convergence rate of (4.16) . Thus for a given node v_i, as ℓ grows, $u_i^{(\ell)}$ converges to its final value u_i exponentially, with a time constant proportional to λ_2.

$$u_i^{(\ell)} \approx u_i + \beta \lambda_2^\ell \qquad (4.20)$$

In our proposed method for accelerating the Jacobi iterations, a node such as v_i performs q Jacobi iterations, then based on the last r observed values $u_i^{(q)}, u_i^{(q-1)}, \ldots, u_i^{(q-r+1)}$, it finds the least squared error estimates for u_i, λ_2 and β, it sets the value of $u_i^{(q+1)}$ equal to the estimated value for u_i and then it resumes to normal Jacobi iterations. This process is repeated until a stopping criterion is met. As suggested by equations (4.18) and (4.19), the value of q should be large enough so that in last r Jacobi iterations the effect of eigenvalues $\lambda_3, \ldots, \lambda s + 1$ become negligible.

4.4 Numerical Examples

Example 4.1. In this example we will use the accelerated Jacobi method to find optimal flow for J_2. Figure 4.2 demonstrates a simple network consisting of two sources, v_1 and v_2, where v_1 injects 1 and v_2 injects 2 units of traffic. Node v_{12} is the sink. The channel capacity of all links are assumed to be equal. Figure 4.3 shows the convergence rate of $u_1^{(\ell)}$ for different acceleration methods for optimization of J_2. In order to provide a benchmark we also included the CG method in the figure; however, as mentioned earlier, CG is not a distributed method. The Jacobi algorithm requires 136 iterations to converge to $u_1 = 4.38$ with error tolerance equal to 1%. For GS and SOR acceleration methods, the number of iterations are 69 and 64 respectively. For our proposed LSE acceleration method, only 22 iterations are sufficient for convergence, which is considerably less than other distributed methods. We set the parameters q and r equal to 20 and 4 respectively. Node v_1 initially performs 20 Jacobi iterations in order to compute $u_1^{(1)}, \ldots, u_1^{(20)}$. Then, based on observations $u_1^{(17)}, \ldots, u_1^{(20)}$, it computes the least squared error estimate of u_1, β, and λ_2, sets $u_1^{(21)}$ equal to the estimated value for u_1, then it resumes to Jacobi iterations. As depicted in figure 4.4, the convergence rates of $u_i^{(\ell)}$ show similar improvement.

p	2	4	6	8
$f_1^{(p)*}$	0.124	0.076	0.037	0.023
$f_2^{(p)*}$	0.384	0.454	0.480	0.488
$f_3^{(p)*}$	0.441	0.470	0.483	0.489
$f_4^{(p)*}$	0.260	0.454	0.480	0.481
$f_5^{(p)*}$	0.107	0.870	0.818	0.797
$f_6^{(p)*}$	0.800	0.752	0.740	0.738
$f_7^{(p)*}$	0.057	0.216	0.238	0.243
$f_8^{(p)*}$	0.538	0.693	0.722	0.732
$f_9^{(p)*}$	0.480	0.686	0.721	0.732
$f_{10}^{(p)*}$	-0.275	-0.615	-0.679	-0.703
$f_{11}^{(p)*}$	**1.335**	**1.485**	**1.497**	**1.500**
$f_{12}^{(p)*}$	**1.522**	**1.515**	**1.503**	**1.500**
$f_{13}^{(p)*}$	0.793	0.747	0.749	0.750
$f_{14}^{(p)*}$	0.633	0.738	0.748	0.750
$f_{15}^{(p)*}$	0.721	0.754	0.751	0.750
$f_{16}^{(p)*}$	0.851	0.762	0.752	0.750
$f_{17}^{(p)*}$	-0.1604	-0.251	-0.251	-0.250
$f_{18}^{(p)*}$	**1.015**	**0.998**	**1.000**	**1.000**
$f_{19}^{(p)*}$	0.130	0.237	0.248	0.250
$f_{20}^{(p)*}$	**1.175**	**1.003**	**1.000**	**1.000**
$f_{21}^{(p)*}$	**1.045**	**0.999**	**1.000**	**1.000**

Table 4.1: Optimal p-norm flow for the network shown in Fig. 4.2.

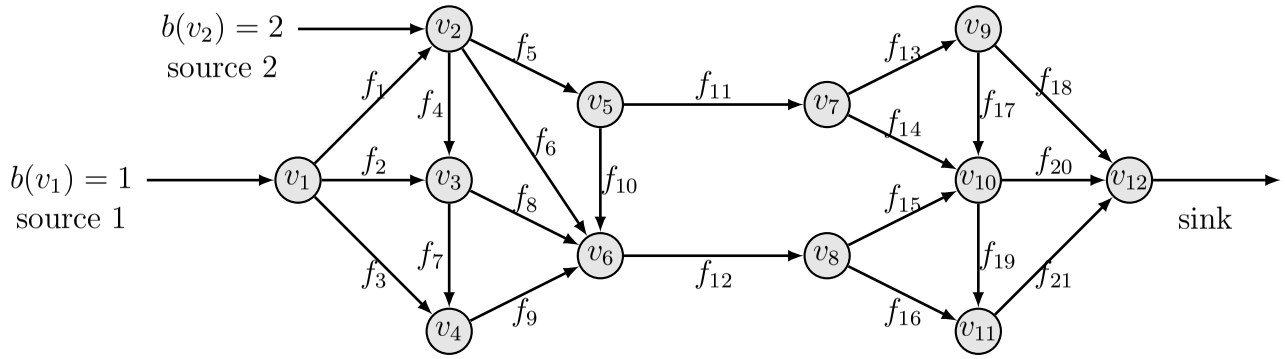

Figure 4.2: A simple network with 12 nodes, where v_1 and v_2 are generating 1 and 2 units of flow respectively, and node v_{12} is gathering the flow. We are assuming that the channel capacity of all of the links are equal.

Table 4.1 shows the optimum value of flow for each link in order to optimize $J_p,\ p = 2, 4, 6, 8$. The flows for $p = 2$ are derived from the calculated potentials using accelerated Jacobi ($\mathbf{f} = \mathbf{K}^T\mathbf{u}$). The flows for $p = 4, 6, 8$ are derived from direct SQP iterations. Observe that as p is increased to relatively small number 8, the flows on the bottleneck links, f_{11} and f_{12} become completely balanced. Increasing p furthermore has negligible effect on the flow allow allocation, thus the SQP iterations are terminated. Furthermore, note that the difference between the accelerated optimization of bottleneck flows for J_2 and the optimum flows for J_8 are negligible, thus in this case, optimizing the 2-norm flow results in a quasi-balanced network with small number of iterations.

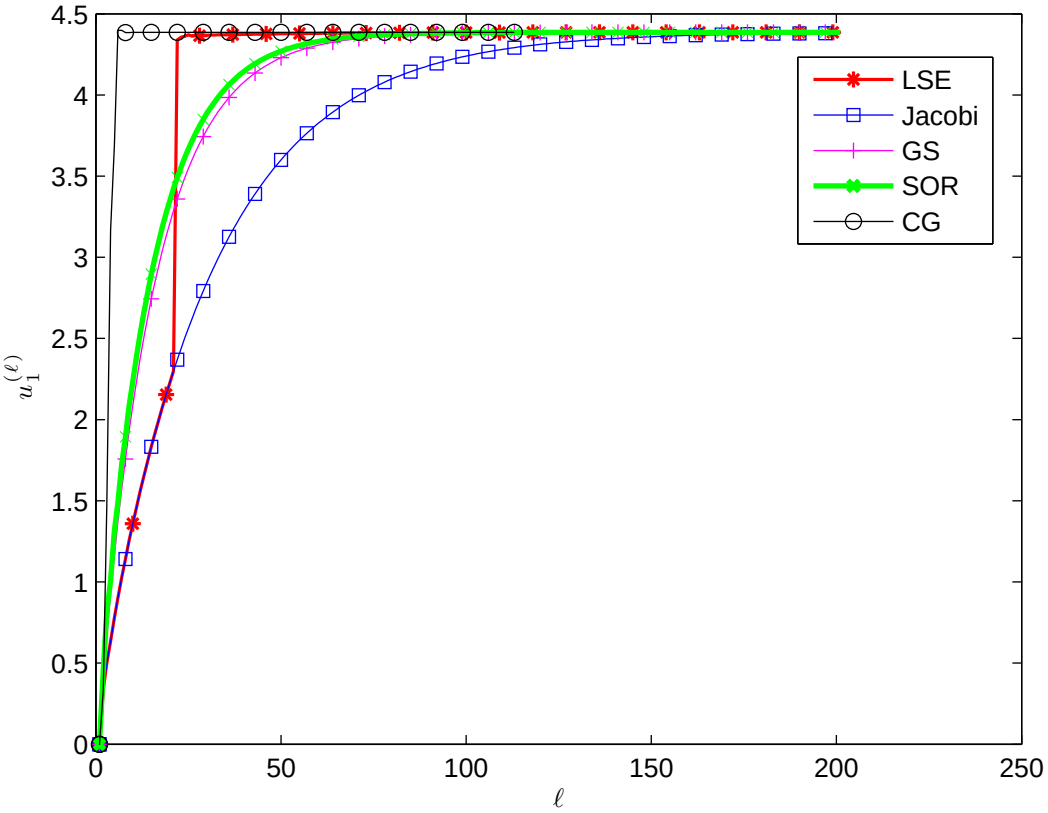

Figure 4.3: Comparison of the convergence rate of Jacobi method, LSE-accelerated Jacobi, Gauss-Seidel (GS), Successive Over Relaxation (SOR) with parameter 1.37 and Conjugate Gradient (CG) methods for optimizing J_2. The parameters of LSE are $q = 20$ and $r = 4$. In this figure, the horizontal axis is the number of iterations and the vertical axis is the value of $u_1^{(\ell)}$ (the Lagrangian multiplier for node 1) in each iteration. We observe the Jacobi, GS and SOR methods require 136, 69 and 64 iterations to converge with error tolerance 1%. Although CG is not a distributed method, we provided its convergence rate in this figure as a benchmark, where it converges in 8 iterations. Our proposed method, LSE accelerated Jacobi, is distributed and converges to the final solution in 22 iterations, which is considerably faster than other distributed methods.

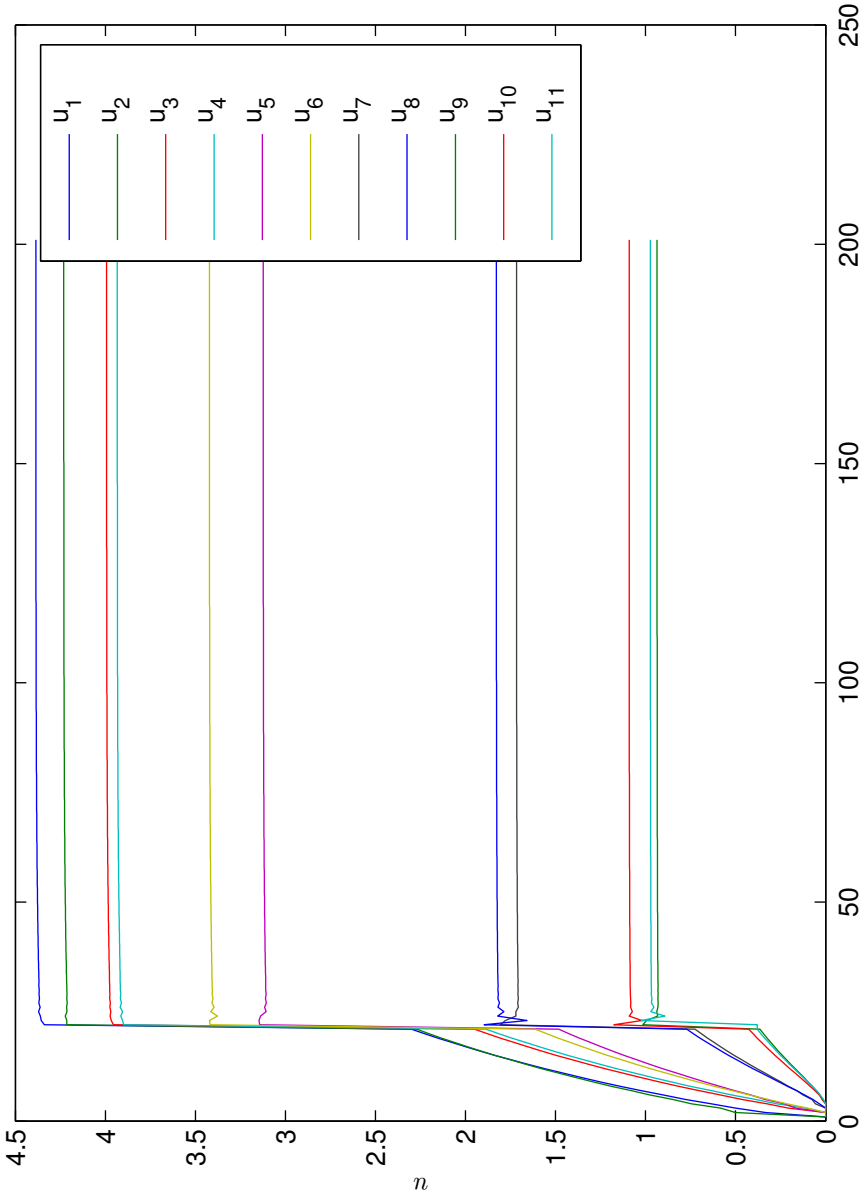

Figure 4.4: The convergence rate of Jacobi iterations for optimizing J_2.

Example 4.2. In this example we examine the eigenvalue distribution of the Laplacian and reduced Laplacian matrices of a network and it relation with convergence rate of the Jacobi method. In the first case we consider the graph studied in the previous example. The largest eigenvalue of matrix W, as depicted in Fig. 4.5, is 0.9669. Furthermore, the algebraic connecting of the graph is 0.4027. In the second case, consider a complete graph, with 12 nodes, such that all of the nodes are neighbors. In this case the largest eigenvalue of W, is 0.9167, and the algebraic connectivity is 12. In the final case, we consider a tandem (or string) graph where the nodes are placed on a line, and except the first and last nodes, all of the nodes only have two neighbors. In this case the largest eigenvalue of W is 0.9898 and the algebraic connectivity is 0.0681.

Figure 4.8 depicts the convergence rate of the Jacobi iterations for these graphs. It is observed that as the graph becomes more connected (the algebraic connectivity increases), the convergence of Jacobi iterations improve. Intuitively, this is an expected result, since as the graph become more connected, the noes can gather more information about the structure of the network, and estimate their potential more accurately. However, a proof for this result (for the connection between algebraic connectivity and convergence rate of Jacobi iterations) is not established in this thesis, and is postponed to future works.

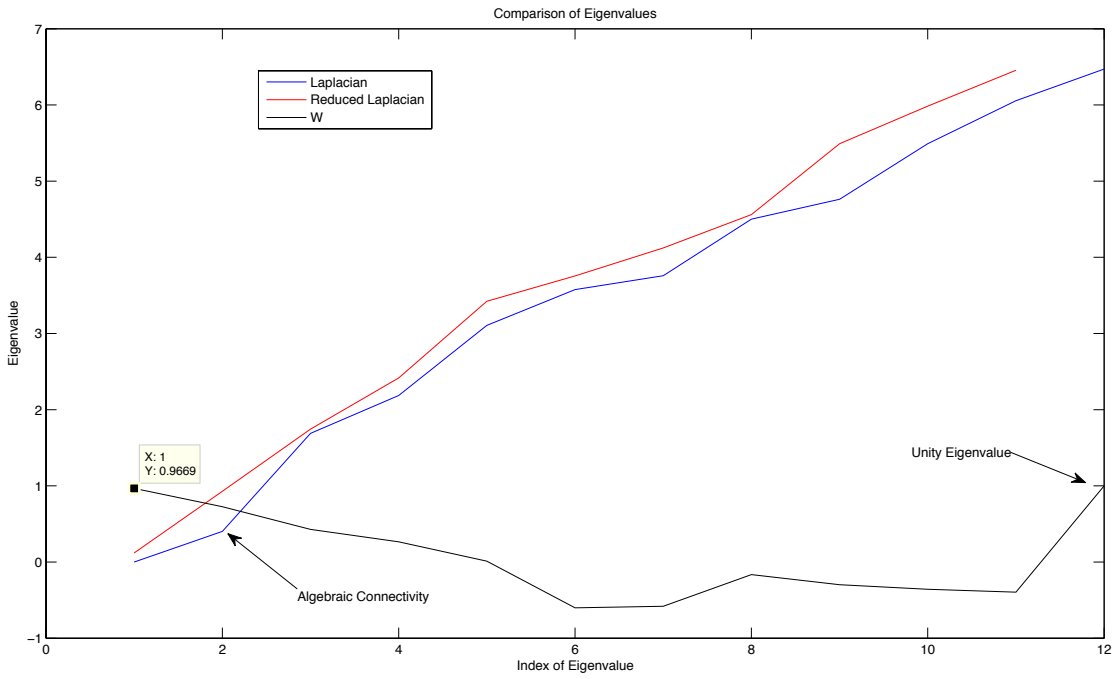

Figure 4.5: Eigenvalues of the Laplacian, reduced laplacian and W for graph depicted in fig. 4.2.

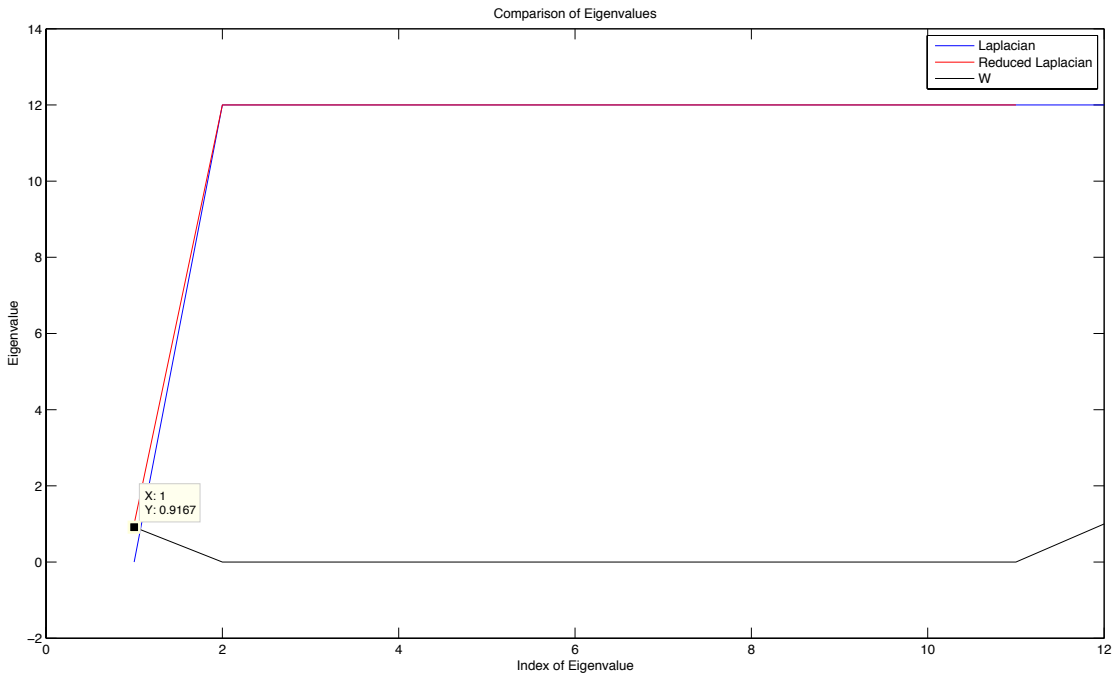

Figure 4.6: Eigenvalues of the Laplacian, reduced laplacian and W for a complete graph K_{12}.

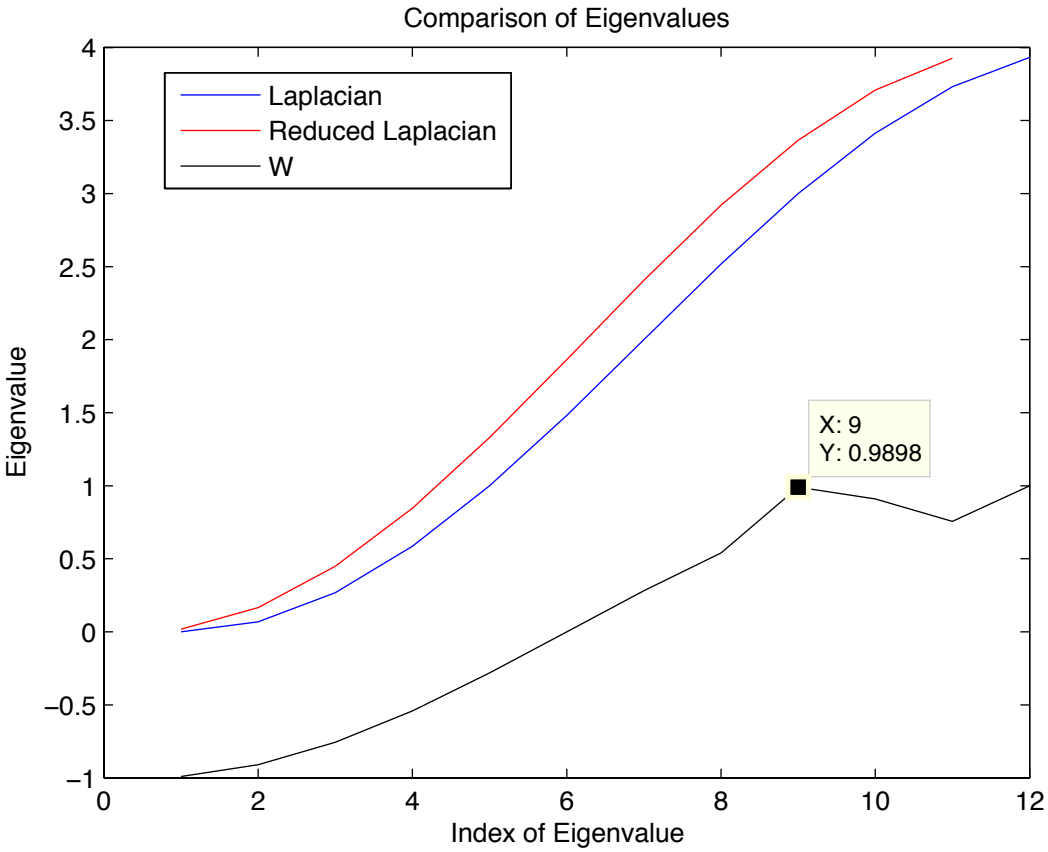

Figure 4.7: Eigenvalues of the Laplacian, reduced laplacian and W for a tandem (string) graph with 12 nodes.

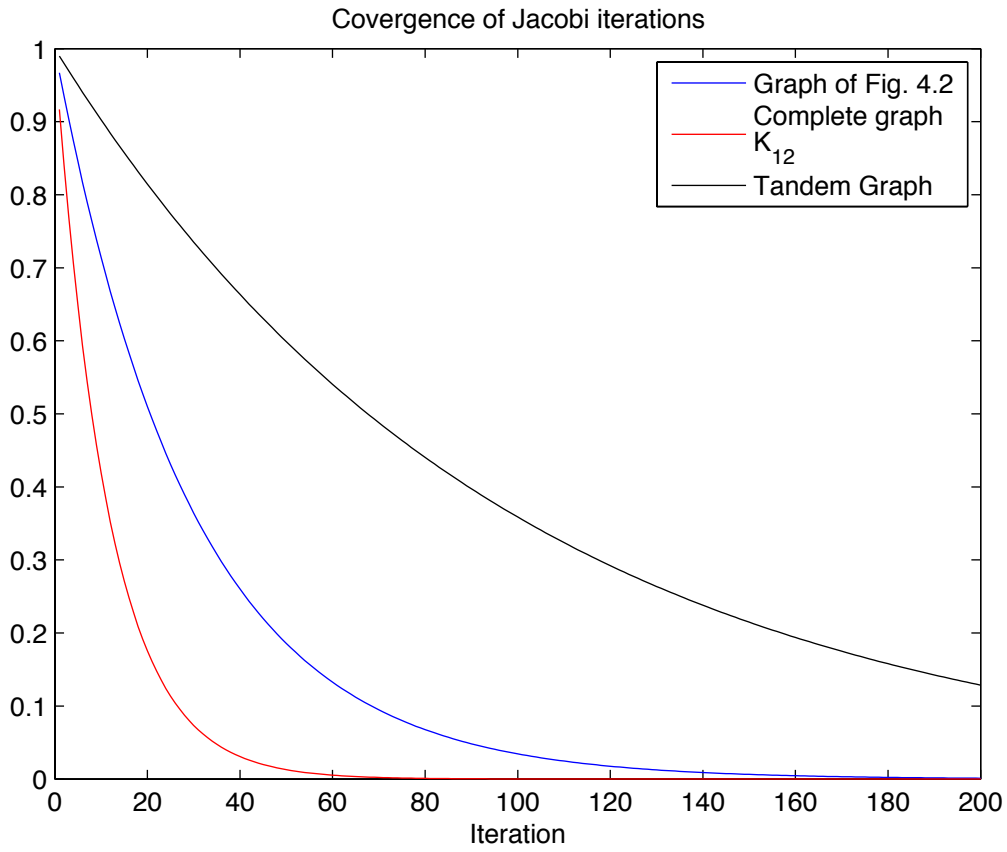

Figure 4.8: Convergence rate of Jacobi iterations for graphs discussed in example 2.

Bibliography

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

[2] IF Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[3] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE wireless communications*, 11(6):6–28, 2004.

[4] E. Altman, P. Bernhard, M. Debbah, and A. Silva. Continuum equilibria for routing in dense ad-hoc networks. In *45th Allerton Conference on Communication, Control and Computing, Illinois, USA, Sept*. Citeseer.

[5] P.T. Boggs and J.W. Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[6] J.A. Boyan and M.L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. *Advances in Neural Information Processing Systems*, pages 671–671, 1994.

[7] J. Bull and TL Freeman. Numerical performance of an asynchronous Jacobi iteration. *Parallel Processing: CONPAR 92VAPP V*, pages 361–366, 1992.

[8] W.K. Chen. *Graph theory and its engineering applications*. World Scientific Pub Co Inc, 1997.

[9] I. Cidon, R. Rom, and Y. Shavitt. Analysis of multi-path routing. *Networking, IEEE/ACM Transactions on*, 7(6):885–896, 1999.

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

[11] M. Haghpanahi, M. Kalantari, and M. Shayman. Implementing Information Paths in a Dense Wireless Sensor Network. *IEEE Global Communications Conference,GLOBECOM*, 2009.

[12] J.D. Hoffman. *Numerical methods for engineers and scientists*. CRC, 2001.

[13] P. Jacquet. Geometry of information propagation in massively dense ad hoc networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, page 162. ACM, 2004.

[14] S. Jung, M. Kserawi, D. Lee, and J.K.K. Rhee. Distributed potential field based routing and autonomous load balancing for wireless mesh networks. *Communications Letters, IEEE*, 13(6):429–431, 2009.

[15] S. Jung, J. Sung, Y. Bang, M. Kserawi, H. Kim, and J.K.K. Rhee. Greedy local routing strategy for autonomous global load balancing based on three-dimensional potential field. *Communications Letters, IEEE*, 14(9):839–841, 2010.

[16] M. Kalantari, M. Haghpanahi, and M. 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.

[17] M. Kalantari and M. Shayman. Energy efficient routing in wireless sensor networks. In *Proc. Conference on Information Sciences and Systems*. Citeseer, 2004.

[18] M. Kalantari and M. Shayman. Routing in wireless ad hoc networks by analogy to electrostatic theory. In *2004 IEEE International Conference on Communications*, volume 7, 2004.

[19] M. Kalantari and M. Shayman. Routing in multi-commodity sensor networks based on partial differential equations. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 402–406, 2006.

[20] M. Li and Y. Liu. Iso-map: Energy-efficient contour mapping in wireless sensor networks. *IEEE transactions on knowledge and data engineering*, pages 699–710, 2010.

[21] P.P. Pham and S. Perreau. Performance analysis of reactive shortest path and multipath routing mechanism with load balance. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 251–259. IEEE, 2003.

[22] M. Stemm and R. H. Katz. Measuring and Reducing Energy Consumption of Network Interfaces in Hand-held Devices. *IEICE Transactions on Communications, E80-B(8):11251131*, 1997.

[23] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, chapter 8. Springer Verlag, 2002.

[24] L. Tang, Y. Sun, O. Gurewitz, and D.B. Johnson. Pw-mac: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1305–1313. IEEE, 2011.

[25] S. Toumpis. Mother nature knows best: A survey of recent results on wireless networks based on analogies with physics. *Computer Networks*, 52(2):360–383, 2008.

[26] S. Toumpis and L. Tassiulas. Optimal deployment of large wireless sensor networks. *IEEE Transactions on Information Theory*, 52(7):2935–2953, 2006.

[27] T. Van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180. ACM, 2003.

[28] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[29] Sina Zahedpour and Mehdi Kalantari. p-norm flow optimization in a network. *CoRR*, abs/1011.2246, 2010.