12-15-2016

Decentralized Convex Optimization for Wireless Sensor Networks

Goutham Kamath

Recommended Citation
Kamath, Goutham, "Decentralized Convex Optimization for Wireless Sensor Networks." Dissertation, Georgia State University, 2016.
doi: https://doi.org/10.57709/9442990

DECENTRALIZED CONVEX OPTIMIZATION FOR WIRELESS SENSOR NETWORKS

by

GOUTHAM KAMATH

Under the Direction of WenZhan Song, PhD

ABSTRACT

Many real-world applications arising in domains such as large-scale machine learning, wired and wireless networks can be formulated as distributed linear least-squares over a large network. These problems often have their data naturally distributed. For instance applications such as seismic imaging, smart grid have the sensors geographically distributed and the current algorithms to analyze these data rely on centralized approach. The data is either gathered manually, or relayed by expensive broadband stations, and then processed at a base station. This approach is time-consuming (weeks to months) and hazardous as the task involves manual data gathering in

extreme conditions. To obtain the solution in real-time, we require decentralized algorithms that do not rely on a fusion center, cluster heads, or multi-hop communication. In this thesis, we propose several decentralized least squares optimization algorithm that are suitable for performing real-time seismic imaging in a sensor network. The algorithms are evaluated and tested using both synthetic and real-data traces. The results validate that our distributed algorithm is able to obtain a satisfactory image similar to centralized computation under constraints of network resources, while distributing the computational burden to sensor nodes.

INDEX WORDS: Decentralized Optimization, Machine Learning, Seismic Tomography

TITLE: DECENTRALIZED CONVEX OPTIMIZATION FOR WIRELESS SENSOR NETWORKS

by

GOUTHAM KAMATH

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2016

TITLE: DECENTRALIZED CONVEX OPTIMIZATION FOR WIRELESS SENSOR NETWORKS

by

GOUTHAM KAMATH

Committee Chair: WenZhan Song

Committee: Sushil Prasad
Xiaojun Cao
Edmond Chow
Xiaojing Ye

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
December 2016

DEDICATION

This dissertation is dedicated to my parents Ramachandra and Rajani Kamath for their endless support, sacrifice, hard work. I also want to thank my wife Rashmi to be part of this journey.

ACKNOWLEDGEMENTS

This dissertation work would not have been possible without the support of many people. I want to express my gratitude to my advisor WenZhan Song for his continuous guidance, patience and support. It is not only his invaluable academic knowledge and methodologies, but also his passionate attitude and discipline to succeed in my future career development. He gave me the invaluable passion and effort for quality research.

I'd also like to thank my fellow lab-mates and co-workers for helping me through valuable knowledge sharing and contributions. I made many great friends and co-workers at Georgia State University and University of Wyoming during my PhD and MS endeavors. I would like to thank Anup Rao for his guidance and support at every stage of my life since highschool. Special thanks goes to Guru and Sharath to support me right from my MS through PhD. I want to thank my lab mates Lei Shi and Song Tan for their wonderful and memorable companionship with research works, productive discussions and support.

Finally, I wish to thank all of my family members and all my friends for their unconditional support, love, patience and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

- GSU - Georgia State University
- CS - Computer Science
- WSN - Wireless Sensor Network
- CPS - Cyber Physical System
- SVD - Singular Value Decomposition
- GPU - Graphics Processing Units
- CAV - Component Averaging
- CARP - Component Average Row Projection

CHAPTER 1

INTRODUCTION

Advancement in wireless sensor network (WSN) technology has enabled us to deploy large number of small, low cost sensors that can sense, actuate and communicate information in areas such as environmental monitoring [1], structural health monitoring [2] and smart grids [3]. These real-world applications involve parameter estimation and can be formulated as a least-squares problem. In distributed Cyber-Physical System (CPS), each sensor node observes partial phenomena due to spatial and temporal restriction and is able to form only partial rows of least-squares. Traditionally, these partial measurements were gathered at a centralized location, however, with the increase in sensors and their measurements, aggregation is becoming challenging and in some cases infeasible. For instance in volcano monitoring, a dense network of stations are used to record seismic vibrations and obtain high-resolution images of volcano conduit [4]. The data recorded by these stations however, are being manually gathered due to its high fidelity and bandwidth limitations [5]. In other CPS such as smart grid, sharing of sensor measurements are restricted due to privacy and security concerns. These constraints demand distributed algorithms that can run on a loosely coupled system such as WSN.

1.1 Analytics on the Edge

Many real-world applications arising in domains such as distributed control [6], large-scale machine learning [7], wired and wireless networks [8], can be formulated as distributed linear least-squares over a large network. These problems have data naturally distributed among various nodes e.g peer-to-peer or sensor network. Today, due to the large volume of the data, we face a challenge to process it in real-time. Therefore, we are particularly interested in solving such problems in a distributed way, with each node handling its local component. This gives rise to several novel techniques commonly referred to as *parallel* and *distributed* algorithms. Both of these

paradigms assume that there exists a central coordinator either to perform intermediate operation or control/monitor the deployed devices. Such algorithms are suitable when the number of devices are small and their interconnection is reliable. In this thesis we will also propose algorithms that are flexible to operate on devices that have unreliable connectivity. Such algorithms are commonly referred to as decentralized algorithms. These method do not require a fusion center, cluster heads, or multi-hop communication as long as the network is connected. We show in the thesis that simple communication with the neighbors would be sufficient for few algorithms to attain optimal solution.

1.2 Design Challenges

Designing a decentralized algorithm presents three main difficulties. First, the underlying hardware in the distributed networks are heterogeneous and often unreliable. Individual sensors may fail at any time, and the communication network that connects them could be highly unstable. Second, the on-board processor often has limited memory, computational power and energy that restricts us from using basic linear algebra tools like SVD, matrix-matrix or matrix-vector multiplications for local computation. Third, due to the large number of nodes and the volatility of the network, any reliance on central coordinator will limit the systems scalability and the performance. The decentralized algorithm must be fault-tolerant as node and link failures are a common occurrence in such systems.

From the recent trends in big data optimization, we see a renewed interest in randomized (stochastic) algorithms both for computation [9] and communication [10]. Today, stochastic methods can solve data-intensive problems on a inexpensive hardware at a faster rate compared to the deterministic methods [11]. For instance, Randomized Kaczmarz has linear convergence rate and outperforms the traditional the conjugate gradient method in some cases [12]. Even in network communications, randomized methods such as gossip protocols are emerging as a new communication paradigm for decentralized systems [13]. These methods guarantee convergence in expectation (probabilistic) similar to stochastic methods; achieve high stability under disruptions, and scale gracefully to a large number of nodes. In comparison, traditional communication techniques

have certain guarantees, but are unstable or fail to make progress during periods of even modest disruption.

1.3 Seismic Application

In this thesis, we develop a suite of algorithms from parallel to decentralized. As a case study we choose to solve a distributed linear system arising from seismic imaging. Current geophysical techniques for visualizing seismic activity employ image reconstruction methods that rely on a centralized approach for processing the raw data captured by seismic sensors. The data is either gathered manually, or relayed by expensive broadband stations, and then processed at a base station. This approach is time-consuming (weeks to months) and hazardous as the task involves manual data gathering in extreme conditions. Also, raw seismic samples are typically in the range of $16-24$ bit, sampled at $50-200$Hz and transferring this high fidelity sample from large number of sensors to a centralized station results in a bottleneck due to bandwidth limitations [14]. To avoid these issues, a new distributed method is required which processes raw seismic samples inside each node and obtains a high-resolution seismic tomography in real time. In this thesis, we propose several decentralized algorithms that can be implemented on sensor network to perform real-time seismic monitoring. The solutions proposed here will help the scientists to analyze and predict the occurrences of volcanoes in real-time. The algorithms are first evaluated for the correctness using a synthetic model in a CORE emulator. Later, the proposed algorithms are run using the real data obtained from Mt. St. Helens, WA, USA. The results validate that the proposed algorithms are able to obtain a satisfactory image similar to the centralized computation under constraints of network resources, while distributing the computational burden to sensor nodes [15].

The rest of the thesis is structured as follows: In Chapter 2 we describe the previous works related to decentralized optimization and seismic tomography. We provide background information of seismic tomography in Chapter 3. In Chapter 4 we present Component Averaging based algorithm that uses fusion center in order to obtain high resolution tomography. Next, in Chapter 5 we improve the component averaging method using adaptive mesh refinement technique. A new distributed algorithm based on gossip method is proposed in Chapter 6. In Chapter 7 we propose

a decentralized asynchronous algorithm suitable for tomography inversion. Finally, we conclude this thesis in Chapter 8.

CHAPTER 2

LITERATURE REVIEW

2.1 Seismic Tomography

The tomography inversion process mainly involves solving a large sparse system of linear equations (Eq. (3.9)). To solve such large sparse system, iterative methods become almost mandatory as they are more efficient in-terms of memory and computational requirements compared to direct methods [16]. Several parallel and distributed iterative methods have been developed and are currently used to solve a large variety of problems [17], [18]. These methods are developed mainly for GPU computing and assume large bandwidth and reliable communication. Communication in sensor networks are unreliable and has a limited bandwidth. These constraints prevent us from using existing methods on sensor network for tomography inversion process.

In order to handle large data, many iterative methods suitable for parallel computing have been developed recently [19]. Among them we found Component Averaging (CAV) [20] and Component Averaged Row Projections (CARP) [21] methods to be suitable for distributed tomography computation over sensor networks [4]. CAV [20] is a cimmino-type method [22] that projects current iterates simultaneously onto all the systems' hyperplanes. n this method, each projection is scaled with a sparsity-related weight, rather than fixed weights used in cimmino, exhibiting faster numerical convergence. CAV retains the desired convergence properties of the Cimmino's method, in the sense that it converges in the inconsistent case.

In CARP, the equations are divided into blocks, and each block is assigned to one processor. The processors operate in parallel and each processor performs one or more Kaczmarz projections on its assigned equations. The different solutions are then merged to become the next iterate by averaging the values of each component across all the blocks in which the component appears [21]. This process is repeated until convergence. CARP does not place any restrictions on the system matrix or on the selection of the blocks and is shown to be robust and memory efficient [21].

Authors in [19] have compared the performance of various block parallel methods on GPU's.

2.2 Decentralized Optimization

In recent years, several decentralized optimization methods have been considered for solving least squares problem. These include, fast gradient methods [23] and decentralized gradient descent (DGD) [24], [25]. For the general convex case, under assumptions of fixed step size and Lipschitz continuous, bounded gradient, [23] shows an outer-loop convergence rate of $O(1/k^2)$ in terms of an objective error and iteration k. It utilizes Nesterov's acceleration, which is the best theoretical rate known so far. However, the inner loop of the algorithm performs substantial consensus computation, without which diminishing step sizes $\alpha_k = k^{-1/3}$ lead to a reduced rate of $O(\log k/k)$. Without bounded gradient, [25] derives a correction method based on mixing matrix for a regular decentralized gradient decent method and obtains $O(1/k)$ convergence rate without diminishing step sizes. All these method assume some form of synchronization across the network between each iteration. For instance, in [24] after each local iteration, all the nodes have to synchronize and exchange information with their neighbors to perform weighted average. The process resumes once all the nodes finish updating the average. Due to the large number of nodes, synchronization or any reliance on central coordinator will limit the systems scalability and the performance.

Asynchronous decentralized methods based on augmented Lagrangian and dual methods [26], [27] shows a linear convergence under strong convexity. Some of the other recent asynchronous decentralized methods include the (sub)gradient method [6],(sub)gradient-push method [28] and the dual-averaging method [8]. Unlike DGD [24], these method exhibit asynchrony, however, these algorithms require an extensive local computation. The significant local computation costs restricts us from using it in sensor network applications.

Freris and Zouzias [29] proposed an asynchronous decentralized method for a sensor network. This algorithm is specially designed for network clock synchronization application, in which the linear system is Laplacian. This algorithm is a special case of the GRK, where the system is restricted to a Laplacian rather than a general linear system. Moreover, the algorithm assumes each sub-matrix $\{A_i, b_i\}$ to have a row dimension $m_i = 1$ and column $n = N$. Our proposed algorithm

have no such assumption and can be extended to a more general least squares problem $m_i \gg n$. Apart from our empirical results and application to a seismic problem, the convergence results and proofs are different.

Unrelated to the decentralized optimization, we note the recent work of HOGWILD! [11], [30]. HOGWILD! framework allows us to perform distributed stochastic gradient descent using several processors. The processors have access to a shared memory that has the iterate and updates them asynchronously without obtaining any locks. The lock-free technique is achieved using the sparsity property of the problem. This framework allows us to obtain a linear convergence for a general convex problem [11] and a sub-linear convergence for the least squares problem [30]. HOGWILD! requires all the processors to have access to a shared memory, which restricts us from using it to a decentralized system such as sensor network.

CHAPTER 3

BACKGROUND

3.1 Seismic Tomography

Tomography can be defined as the science of computing reconstructions in 2D and 3D from projections, i.e., solving the system of linear equations obtained by integrations along the rays that penetrate a domain Ω, typically a rectangle in 2D, and a box in 3D. In this paper we use first-arrival travel time of the p-wave (primary wave) to derive the internal velocity structure of a volcano. Fig. 3.1(a) shows the sample of p-wave obtained at four different stations and the blue line indicates its corresponding arrival time. Next, we explain four basic principles involved in travel-time seismic tomography.

3.2 Phases of Seismic Tomography

i) P-wave arrival-time picking: P-waves travel faster than any other waves though the earth and are the first to be recorded in the seismic sensors. Inside the earth, density varies due to the presence of different layers and materials. These cause the seismic waves to travel at different velocities in different directions as shown in Fig. 3.1(b). By picking the arrival time of the p-wave at different stations we can obtain the difference in the propagation delay. Picking arrival time is inherently distributed and authors in [31] have proposed methods to automatically pick the arrival time which can be implemented on each station.

ii) Event Location: Once the arrival times of p-waves has been detected by each station, their differences can be used to obtain the exact location and the origin time of the earthquake. Geiger's [32] method is used to calculate the event location along with origin time and it requires travel time differences from at least four different stations. This method is one of the classic and widely used event localization schemes to obtain the exact location and time.

iii) Ray Tracing: This is the technique used to find the ray paths between the seismic source lo-

cations (earthquake) and the receiver nodes with minimum travel time, following an event. Once an earthquake occurs, the seismic rays originating from these events perturb and register the anomalous residuals. Given the source location of the seismic events and current velocity model of the volcano, the ray tracing method finds the ray paths from the event source locations to the nodes, as shown in Fig. 3.1(c). The p-waves traveling in different mediums under the earth tend to bend due to reflection and refraction resulting in a curved path during the ray tracing [33]. However, for the sake of simplicity in this paper we assume that the rays travel in a straight path.

iv) Tomography Inversion: The traced ray paths are in turn used to estimate the velocity model of the volcano. Inheriting the concept from medical tomography, the volcano can be partitioned into smaller blocks called pixels (2D) and voxels (3D), as shown in Fig. 3.1(d), to form a large, sparse system of linear equations. In the next section, we show the formulation of the travel-time seismic tomography problem using the perturbation model.

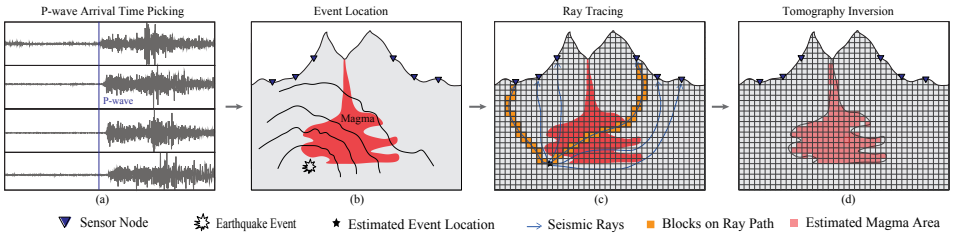


Figure (3.1) Different steps involved in seismic tomography. a) Identifying the first arrival time of P-wave at each sensor. b) Earthquake localization using P-wave arrival time. c) Calculating the path between earthquake location and the sensor traveled by the P-wave. d) Inversion process to generate the seismic tomography.

3.3 Tomography Formulation

Let us assume that the time taken by seismic signal to travel from the source (earthquake) to a receiver (station), in a given medium, is a function of the seismic velocity of the material and the ray path. Given the velocity model, source and receiver locations, calculating the travel time is

called a forward problem, given by

$$T(v, ray) = \int_{ray} \frac{1}{v(y)} d\tau, \tag{3.1}$$

where, T is the travel time, v(y) represents velocity of the material and spatial position y, and $d\tau$ is a differential line element along the path. Since, T does not depend linearly of the velocity, v(y), for convenience we introduce *slowness* where, $s(y) = 1/v(y)$. Therefore Eq. (3.1) becomes

$$T(v, ray) = \int_{ray} s(y) d\tau. \tag{3.2}$$

Since the ray path traveled depends of the slowness, it is non-linear and can be linearized using the perturbation approach [34]. Suppose, we have a reference model $s_0(y)$, the slowness model $s(y)$ we intend to obtain is given by,

$$s(y) = s_0(y) + \delta s(y). \tag{3.3}$$

Substituting Eq. (3.3) in Eq. (3.2), we get,

$$T = \int_{ray} s_0(y) d\tau + \int_{ray} \delta s(y) d\tau, \tag{3.4}$$

and by rearranging we get,

$$\delta T = T - T_0 \approx \int_{ray} \delta s(y) d\tau, \tag{3.5}$$

where, δT is called travel time residual. This gives the linear relationship between δT and slowness perturbation δs.

Next we will show how the continuous function $\delta s(y)$ can be discretized by dividing a square domain $\Omega = [0, 1] \times [0, 1]$ into a grid of pixels. For simplicity we assume 2D scenario with regular $N \times N$ grids. When Eq. (3.5) is discretized on $N \times N$ array of pixels, let us assume $s(y)$ takes a constant value $s_{k\ell}$ over the pixels (k, ℓ), where k and $\ell \in \{1, \cdots, N\}$.

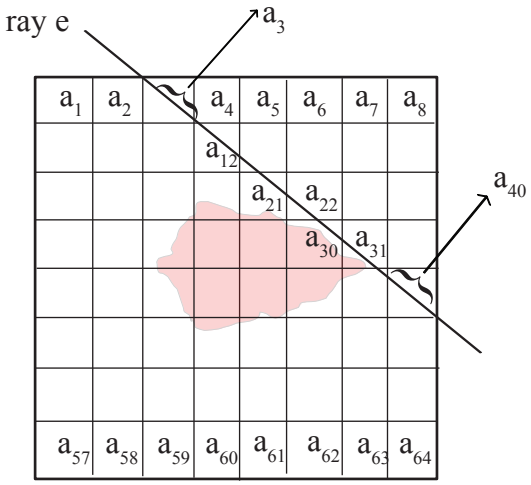


Figure (3.2) Example of discretized tomography with $N = 8$ in sensor i. The e-th ray intersect a total of 9 pixels, and thus the e-th row of matrix A_i has 9 non-zero elements (in columns 3, 4, 12, 13, 21, 22, 30, 31, and 40)

Let us consider a single ray $e \in \{1, \cdots, m_i\}$, where m_i denotes total number of rays in sensor node i. Also let the travel time residual for ray e is given by $b_e = \delta T_e$. Now, by substituting the discretized $s_{k\ell}$ into (3.5) we get,

$$b_e = \sum_{(k,\ell)\in ray_e} s_{k\ell} \Delta L^e_{k\ell}, \tag{3.6}$$

where, $\Delta L^e_{k\ell}$ = length of ray_e in pixel (k, ℓ). We can re-write the above equation in a more simplified form by numbering pixel (k, ℓ) with j i.e. $x_j = s_{k\ell}$ and $a_{ej} = \Delta L^e_{k\ell}$, where $j = (\ell - 1)N + k$, we get

$$b_e = \sum_{j=1}^{n} a_{ej} x_j, \quad e \in \{1, \cdots, m_i\}; \quad n = N^2 \tag{3.7}$$

In the matrix form the system of equations formed at the i^{th} sensor node is given by,

$$A_i x = B_i, \qquad A_i \in \mathbb{R}^{m_i \times n}, x \in \mathbb{R}^n \tag{3.8}$$

where,

$$B_i = [b_1, \cdots, b_{m_i}]^T, a_{ej} = \begin{cases} \Delta L^e_{k\ell} & (k,\ell) \in ray_e \\ 0 & \text{otherwise} \end{cases}$$

Now for P number of sensors i.e. $i \in \{1, \cdots, P\}$ we have

$$Ax = B \tag{3.9}$$

where,

$$\mathbb{A} = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{pmatrix}; b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}; A_i \in \mathbb{R}^{m_i \times n}; b_i \in \mathbb{R}^{m_i}, \tag{3.10}$$

In this thesis, we consider solving the Eq. (3.9) in a distributed way over a loosely connected decentralized network. We assume that each node i has $\{A_i, B_i\}$ obtained from steps such as arrival time picking, event location and ray tracing.

3.4 Convex Optimization

In this thesis, we consider N edge nodes connected to form an arbitrary topology given by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V} = \{1, \cdots, N\}$ and edge set $\mathcal{E}$ that contains set of links in the network. We have $\{i, j\} \in \mathcal{E}$ if node i and node j can communicate with each other. $\mathcal{N}_i$ denotes the neighbor set of node i. Each edge node i is connected to one or more seismic sensors and have m_i training samples given by, $\{(x_1, y_1), (x_2, y_2), \cdots, (x_{m_i}, y_{m_i})\}$, where $\sum_{i=1}^{N} m_i = m$. Now we can form a local loss function at node i given by $f_i(x) = \frac{1}{2m_i} \sum_{i=1}^{m_i} \|y_i - x_i^\intercal \theta\|_2^2 + \lambda \|\theta\|_2^2$.

Now combining all to a single equation we get the following optimization problem,

$$\min_{\theta \in \mathbb{R}^n} \quad \frac{1}{2N} \sum_{i=1}^{N} \|Y_i - X_i^\intercal \theta\|_2^2 + \lambda \|\theta\|_2^2, \tag{3.11}$$

where, $X_i \in \mathbb{R}^{m_i \times n}$; $Y_i \in \mathbb{R}^{m_i}$ denote the m_i rows training set at node i. Here, the *regularization pa-*

rameter λ is a positive number that controls the weight between $\|Y_i - X_i^\intercal \theta\|_2^2$ (goodness fit measure) and $\|\|\|_2^2$ (regularity measure). We represent θ^k feature estimated at the iteration k. The goal now is to develop an algorithm to solve regularized regression problem Eq. (7.1) in a decentralized and asynchronous way on the edge network.

CHAPTER 4

PARALLEL ALGORITHMS FOR WSN

4.1 Introduction

Image reconstruction problems have large sparse linear systems and iterative methods are routinely used to solve them [35]. Algebraic Reconstruction Technique (ART) [36] also know as Kaczmarz (KACZ) [37] was the first iterative method used to solve such problems. This algorithm is inherently sequential, where initial vector x is projected onto rows of A in a cyclic manner. At each iteration, the previous iterate is projected orthogonally onto the hyperplane defined by the equation $\langle a_i, x\rangle = b_i$. Here, we denote a_i and b_i as the i^{th} row vector of the matrix A and i^{th} element of b respectively. The Kaczmarz algorithm is given by

Algorithm 1 Kaczmarz

1: Initialize: $x^0 \leftarrow 0$
2: **for** $k \leftarrow 0$ until convergence or max iteration **do**
3: $\quad i \leftarrow$ k mod m + 1
4: $\quad x^{(k+1)} = x^{(k)} + \rho_i \frac{b_i - \langle a_i, x^{(k)}\rangle}{\|a_i\|^2} a_i$
5: **end**

where, ρ_i is a cyclic relaxation parameter that extends the projections either in front of the hyperplane ($\rho_i < 1$), on the hyperplane ($\rho_i = 1$), or beyond the hyperplane ($\rho_i > 1$). The convergence of KACZ with relaxation parameter ($0 < \rho < 2$) for a consistent system has been shown in [38, 39].

In order to perform distributed tomography in sensor network, reconstruction algorithms such as CAV and CARP were found to be suitable [4]. These algorithms belong to parallel reconstruction techniques, where during each iteration, all the blocks (single equation in case of CAV) are processed in parallel (by some algorithm operating on each block independently of the others), and then the next iterate is formed by applying some operation to the partial solutions from the different blocks. These algorithms are also known as *string averaging* methods [20]. CARP in particular is designed as a general method and places no restriction on the system matrix or the selection of

the blocks. CARP uses ART to perform local iteration. The partial solutions from all the blocks are then combined to form the next iterate using component wise averaging. CARP is shown to be equivalent to KACZ in some superspace and this makes it attractive for distributed tomography. The row projection property of KACZ, i.e processing each row at a time, makes it promising to be implemented on tiny devices like Beaglebone black due to its small memory footprint (each row being sparse).

4.2 Parallel Kaczmarz using Component Averaging

Until now, Kaczmarz was used only in a centralized setup to generate seismic tomography. All the raw data sampled from each station had to be manually collected, pre-processed, analyzed, and then finally interpreted. Data gathering is the most challenging step, and this motivated us to develop in-network distributed algorithms. In this section, we provide details regarding Distributed Bayesian ART (D-BART) that can be implemented on sensor nodes to perform distributed tomographic inversion.

Suppose there are P sensor nodes in the network. Let the i^{th} sensor perform event detection and ray tracing as mentioned in Section 3.3 to form $A_i \in \mathbb{R}^{m_i \times n}$ and $B_i \in \mathbb{R}^{m_i}$. Here n denotes the resolution of the tomography image and m_i is the number of rows of A_i, which is the number of p-waves detected and traced. Now we let A_i^j represent the j^{th} column of the matrix A_i, and we identify the non-zero columns of A_i, i.e for i^{th} station and $1 \leq j \leq n$ we let

$$s_i^j = \begin{cases} 1 & \text{if } A_i^j \neq 0 \\ 0 & \text{if } A_i^j = 0 \end{cases} \tag{4.1}$$

This s_i^j is sent to a *SINK* node where $S^j = \sum_{i=1}^{P} s_i^j \;\; 1 \leq j \leq n$ is calculated. S^j denotes total number of blocks that have at least one non-zero element in the j^{th} column. This S^j will be used to perform component averaging, where partial slowness obtained from each station will be combined with others. Next, we show how component averaging is done to obtain the next iterate.

Let $A = \{A_1, \cdots, A_P\}$ and $\bar{x}_i^j$ denote the j^{th} component of partial slowness obtained from i^{th}

station after solving the linear equation $A_i x = B_i$. The component averaging operator relative to A is the transfer operator $CA_A : (\mathbb{R}^n)^P \to (\mathbb{R}^n)$ and is defined as follows: Let $\{\bar{x}_1, \cdots, \bar{x}_P\} \in \mathbb{R}^n$ be partial solution from all P sensor nodes. Then $CA_A(\bar{x}_1, \cdots, \bar{x}_P)$ is the point in $\mathbb{R}^n$ whose j^{th} component is given by

$$CA_A(\bar{x}_1, \cdots, \bar{x}_P)^j = \frac{1}{S^j} \sum_{t=1}^{P} \bar{x}_t^j.$$

Algorithm 2 Parallel Kaczmarz

1: Initialize: $x^0 \leftarrow 0$; λ; κ_1; κ_2
2: **for** $k \leftarrow 0$ until convergence or maximum number of iteration **do**
3: **for** $i \leftarrow 1, \cdots, P$ execute in Parallel
4: $x_i^k = \text{KACZ}^t(A_i, b_i, x^{k-1}, \lambda, \rho_k)$
5: **end**
6: $(x^j)^k = \frac{1}{S^j} \sum_{i=1}^{P} (x_i^j)^k \quad \forall\, \text{j} = 1, \cdots, n$
7: **end**

Algorithm (2) presents Parallel Kaczmarz where KACZ^t denotes t internal iterations of Kaczmarz algorithm (1).

4.3 Convergence and Analysis

To prove the convergence of Parallel Kaczmarz we first transform the system given in Eq. (3.9) into system of equations in some superspace $\mathbb{R}^s$ of $\mathbb{R}^n$. Let $B_t = \{A_t, b_t\}$, be a tuple containing known terms of subsystem and $x_t \in \mathbb{R}^n$ be the partial solution of subsystem t for $1 \leq t \leq P$. We know that, x_t can share some common variable with $x_{t'}$ if A_t and $A_{t'}$ has common non-zero column. Now without loss of generality, for $1 \leq r \leq n$ the components $\{x_1, \cdots, x_r\}$ be exactly share with two or more nodes i.e $s_1, \cdots, s_r \geq 2$, while $s_{r+1}, \cdots, s_n = 1$. From this we have $n - r$ components of x not shared by any blocks and can be computed without needing any data exchange.

Now,the expansion mapping can be given by $E : \mathbb{R}^n \to \mathbb{R}^s$:

$$E(x_1, \cdots, x_n) =$$

$$(y_{1,1}, \cdots, y_{1,s_1}, \cdots, y_{r,1}, \cdots, y_{r,s_r}, y_{r+1}, \cdots, y_n),$$

where $y_{j,1} = \cdots = y_{j,s_j} = x_j$ for $1 \leq j \leq r$ and $y_j = x_j$ for $r < j \leq n$.

Similarly, we can transform the equation of the subsystem B_t from $\mathbb{R}^n$ to $\mathbb{R}^s$ which we will denote it as $B'_t = \{A'_t, b'_t\}$. The new transformed equation in B'_t do not share any common variable with any other blocks. Let, $B' = \cup_{t=1}^{P} B'_t$ represent all the subsystem stacked together. Now, parallel execution RK on each node B_t for $1 \leq t \leq P$ is equivalent to performing RK on B'. Next, we will show that averaging the shared variable of the system B, is equivalent to certain row projections. From this it follows that Par-RK is just RK in $\mathbb{R}^s$.

Lemma 1 *Let $1 \leq m \leq P$, $y^0 = (y_1^0, \cdots, y_P^0) \in (\mathbb{R})^P$ and let $y^1 = (y_1^1, \cdots, y_P^1) \in (\mathbb{R})^P$ be defined as follows: $y_i^1 = (y_1^0 + \cdots + y_m^0)/m$ for $1 \leq i \leq m$, and $y_i^1 = y_i^0$ for $m < i \leq P$. Then y^1 can be obtained from y^0 by performing a sequence of $(m-1)$ orthogonal projections on hyperplanes of $\mathbb{R}^n$ as in KACZ.*

Proof 1 *The proof is by induction on m. For m = 1, there is nothing to prove. For m = 2, project y^0 onto the plane defined by the equation $-y_1 + y_2 = 0$. The vector of coefficient of a are $\{-1, 1, 0, \cdots, 0\}$ and $\|a\|^2 = 2$. The projection $\tilde{y} = \{\tilde{y}_1, \tilde{y}_2, \cdots\}$ is*

$$\begin{aligned}
\tilde{y} &= y^0 - \frac{1}{2}\langle y^0, a\rangle a \\
&= (y_1^0, y_2^0, \cdots) - \frac{1}{2}(-y_1^0 + y_2^0)(-1, 1, 0, \cdots, 0) \\
&= \left(\frac{1}{2}(y_1^0 + y_2^0), \frac{1}{2}(y_1^0 + y_2^0), y_3^0, y_4^0, \cdots\right)
\end{aligned}$$

In other words, for m = 2 nodes we obtain $\tilde{y}_1 = \tilde{y}_2 = \frac{(y_1^0 + y_2^0)}{2}$ by performing one orthogonal projection on a suitable hyperplane. We assume that the statement is true for m, and we will prove it for m+1. Let, $y^0 = \{y_1^0, \cdots, y_n^0\}$ and we project y^0 onto the hyperlane defined by the equation $-y_1 - y_2 \cdots - y_m + m y_{m+1} = 0$. Now, $a = (-1, \cdots, -1, m, 0, \cdots, 0)$ and $\|a\|^2 = m + m^2 = m(m+1)$. The projection is the point $y' = \{y'_1, \cdots, y'_n\|$ defined by $y' = y^0 - (\langle y^0, a\rangle a)/(m(m+1))$. Substituting y^0 and a, we have

$$y' = (y_1^0, \cdots, y_n^0) - \frac{-y_1 - y_2 \cdots - y_m + m y_{m+1}}{m(m+1)}$$
$$(-1, \cdots, -1, m, 0, \cdots, 0)$$

For each $1 \leq i \leq m$*, we have*

$$y_i' = y_i^0 - \frac{1}{m(m+1)} \left(\sum_{j=1}^{m} y_j^0 - m y_{m+1}^0 \right),$$

and the (m+1)st coefficient is

$$\begin{aligned} y_{m+1}' &= y_{m+1}^0 + \frac{(y_1^0 + \cdots + y_m^0 - m y_{m+1}^0}{m(m+1)} m \\ &= \frac{1}{m+1}(y_1^0 + \cdots + y_m^0) + \left(1 - \frac{m}{m+1}\right) y_{m+1}^0 \\ &= \frac{1}{m+1}(y_1^0 + \cdots + y_m^0) + \frac{m+1-m}{m+1} y_{m+1}^0 \\ &= \frac{1}{m+1}(y_1^0 + \cdots + y_m^0 + y_{m+1}^0) \end{aligned}$$

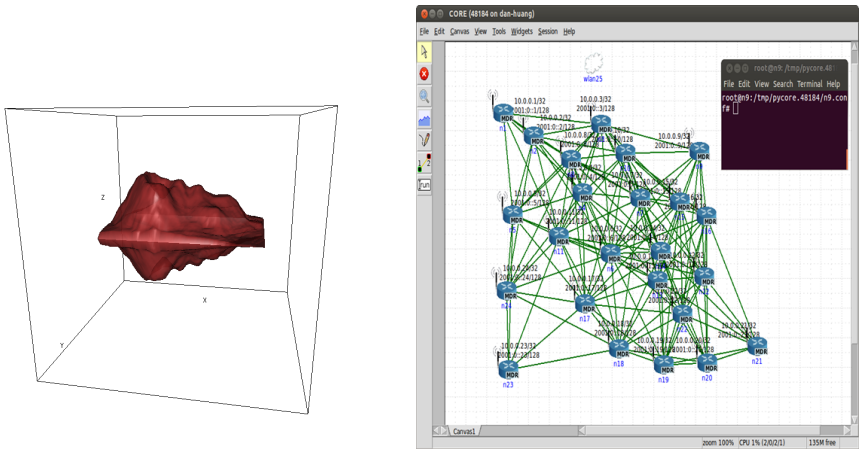
This proves the induction hypothesis and the lemma

We have shown that averaging is equivalent to row projection in some superspace $\mathbb{R}^s$. Let, $\tilde{B}'$ be the auxiliary averaging equation and we form B'' by adding B' with $\tilde{B}'$, with increased row proportional to number of shared variables. Now, if the Eq. (3.9) is consistent, then set of transformed equations B' and average equations $\tilde{B}'$ are also consistent. Strohmer and Vershynin [12] showed that RK converges on a consistent system even if the projections are not performed cyclically; all that is required is that each equation should be used infinitely often. This allows us to perform Kaczmarz in Algorithm 2 for any positive number of iteration in each block. We refer to work of [40] in case of the inconsistent system. This proves the convergence of Par-RK.

4.4 Evaluation

To evaluate the algorithm, the data generator is implemented to generate a magma area and earthquake events assuming the tomography model is a cube of dimension $10 \times 10 \times 10$ km. Then we set a predefined magma area as the ground truth as shown in Fig. 4.1(a). The velocities of seismic waves inside and outside the magma area are V and $0.9V$ where V is 4.5km/s which is a typical P-wave velocity.

A network of 100 nodes (Fig. 4.1(b)) are setup in CORE emulator to monitor the magma area. We set the final tomography resolution to be $32 \times 32 \times 32$ where each block is of the size 0.315 km^3. The data generator then generates earthquake events with random location and time, and calculates ray travel time from event location to all sensor nodes. To simulate the event location estimation and ray tracing errors, a white Gaussian noise is added to the travel time. Each node can calculate the predicted travel time based on the initial model in different resolution.



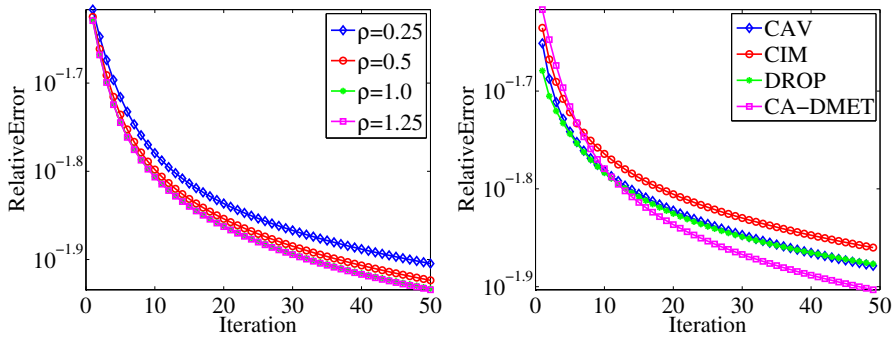
(a) 3D Synthetic Phantom (b) Station setup in CORE

Figure (4.1) (a) 3D synthetic magma model (b) Snapshot of the CORE GUI which displays the nodes along with communication link.

In the implementation, the Bayesian ART method is performed for 10 iterations locally to solve the equation system on each node. We use the relative slowness perturbation updates of the estimation between the two sweeps (one sweep means that all partial slowness perturbation is averaged to calculate next iterate) as the stopping criteria. If the relative update (ϕ) is less than

a tolerance, the CA-DMET stops. To compare the performance of CA-DMET, we also used the centralized Bayesian ART to solve the system at target resolution with all 900 events (case (4)).

Performance of CA-DMET is compared using their relative update ($\phi = |x^{(k+1)} - x^{(k)}|/|x^{(k)}|$), relative residual ($\chi = ||Ax^k - B||/||B||$) and relative error ($\delta = |x^{(k)} - x^{truth}|/||x^{truth}||$).



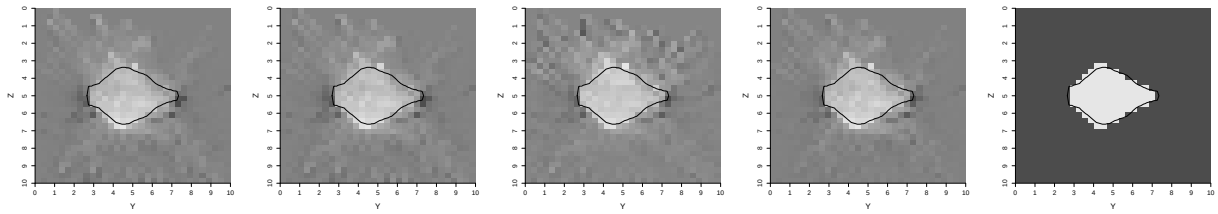
(a) (b) Relative Residual

Figure (4.2) a) Relative Error measurement with various relaxation parameters - Case 1 b) Comparison of different algorithms

We first run experiments to show the behavior of different relaxation parameters on CA-DMET and also choose an optimal ρ for a given set of synthetic data. CA-DMET is performed on case 3 by varying ρ from 0.25 to 1.25 and the plots are shown in Fig. 4.2. Starting from a small relaxation parameter, each successive value of ρ increases the convergence rate until an optimal value is reached. We found that further increase in the relaxation parameter worsened the result and an optimal relaxation parameter was found to be 1.25 for the given synthetic magma model. In all our experiments with synthetic data, the relaxation parameter remained constant throughout the iterations, i.e., $\rho^k = \rho$ for all $k \geq 0$.

In the next set of experiments, we compare the relative performance of CA-DMET with three different algorithms: CAV, Cimmino and un-weighted DROP [41]. We use relative error as the parameter for comparison and results shown in Fig. 4.2(b) demonstrate that there is a difference in the initial convergence between CAV, Cimmino, DROP and CA-DMET. A visual verification of all the algorithms are shown in Fig. 4.3. All the algorithms are run for same number of iterations. The reconstructed image from different algorithms reveal that CA-DMET is able to obtain a better

magma image with less perturbation noise outside the region compared to other algorithms.



(a) Cimmino Res: 32^3 (b) CAV Res: 32^3 (c) DROP Res: 32^3 (d) CA-DMET Res: 32^3 (e) Phantom Res: 32^3

Figure (4.3) 2D Tomography Rendering from Different Algorithms

CHAPTER 5

PRE-CONDITIONING USING ADAPTIVE MESH

5.1 Introduction

Adaptive mesh refinement (AMR) has been studied widely and has been used as discretization tool for partial differential equation as early as 1980 [42]. However, only until early 90's it was used by seismology community to solve inverse problem on small set of data [43]. [44] used SVD to interactively change the boundaries while, [45] used genetic algorithm to optimize the parametrization. These algorithms were suitable for small size data sets and required high computational power to run efficiently. [46] came up with a less computation intensive solution to parametrize the coefficient and this algorithm could run efficiently even for large matrices. However, this algorithm is only suitable for centralized architecture and is not feasible to be implemented in a distributed scenario. To perform adaptive mesh in a distributed case, we had to come up with some novel method which was computationally light and also satisfy the requirements such as faster convergence. To the best of our knowledge, our work is the first attempt to compute seismic tomography using adaptive mesh over a distributed sensor networks.

Many geophysical inverse problems are ill-conditioned i.e. model space contains more details than it can be resolved using available data space [47]. Model space matrix A typically has large null space and because of this, few portions of the solution cannot be resolved leading to its non-uniqueness. Geophysicists commonly use two approaches to overcome this problem: firstly, by making the problem well conditioned using some a-priori information such as smoothness constraints, regularization etc [36]. However, obtaining reliable prior information is hard and also sometimes the smoothness constraints can be unrealistic. The second approach is by identifying the eigenvalue and eigenvector corresponding to the null space of model and later removing them explicitly. Although this method reduces the effective information content of the data set in the model space in a nontrivial manner, it can be used to obtain maximum amount of information that

can be resolved from the current data set which will then reduce the amount of additional a-priori information to be included in the solution.

Removal of null space from the model data is equivalent to parameter reduction and here we will explain this with a simple example. Consider the path geometry of the ray produced by source (circle) and receiver (square) as shown in Figure 5.1. Let us suppose that we have an error-free measurement of average velocity and each of the four cells in Figure 5.1(a) have exactly equal ray path coverage. From measurements along the two left-hand paths, the quantity $v1 + v2$ can be determined exactly. The quantity $v1 - v2$, however, remains entirely unresolved by the data, and hence velocities v1 and v2 cannot be determined. Similarly, the combination $v3 + v4$ may be determined exactly, but not $v3 - v4$ and hence v3 and v4 cannot be determined.

Let $v = [v_1, \cdots, v_2]^T$, then well-determined combinations are $e_1.v$ and $e_2.v$. Whereas, the undetermined combinations are $e_3.v$ and $e_4.v$, where

$$e_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}; e_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}; e_3 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}; e_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

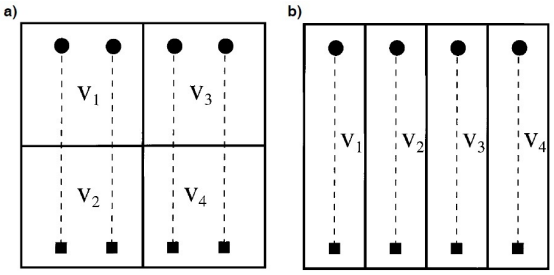


Figure (5.1) Cell geometries for the event (circle) station (square) paths shown. In (a) the cells bisect each path whereas in (b) each cell contains exactly one path length. Both cell geometries have exactly the same homogeneous path coverage within each cell

Vectors e_1 to e_4 are the eigenvectors of the inverse problem and v_1 to v_4 the velocities we are interested. Parameter combinations parallel to eigenvectors e_1 and e_2 are completely determined if and only if (iff) the corresponding eigenvalues are large, however combinations parallel

to eigenvectors e_3 and e_4 are undetermined iff they correspond to small or zero eigenvalues.

Remark 1 *Removal of parameters that creates null space involve modification of the grid structure based on ray coverage which also changes the resolution and information content.*

5.2 Adaptive mesh using quadtree/octree

To perform quadtree based AMR on seismic tomography first we need to generate a density matrix I which is checked for homogeneity criteria and later split. Matrix I should be in $\mathbb{R}^{n\times n}$ where $n \in 2^k, k > 0$ and this restricts us to have the finest resolution to be power of 2. We use hit count to generate I i.e. I_{ij} = number of rays passing through ij-th grid of finest resolution. There are other ways to choose I such as region of interest and other hybrid methods [46] and this requires domain knowledge and will not be discussed in this paper. Now with only hit count, I can be generated by performing columnSum(A) and then reshaping it to $n \times n$. To generate quadtree S we use QTDecomp function which first initializes S to size of I which becomes the root. Later this S is divided into four equal sized squares if the corresponding blocks in I satisfy the criterion of homogeneity. This process is further continued recursively and stops if either the block does not meet the criteria or depth of the tree is k i.e. until finest resolution. After obtaining the quadtree S we can easily generate Δ by using algorithm [3]. In this paper, we do not discuss the efficient way to implement quadtree data structures and further details on this can be found in [48].

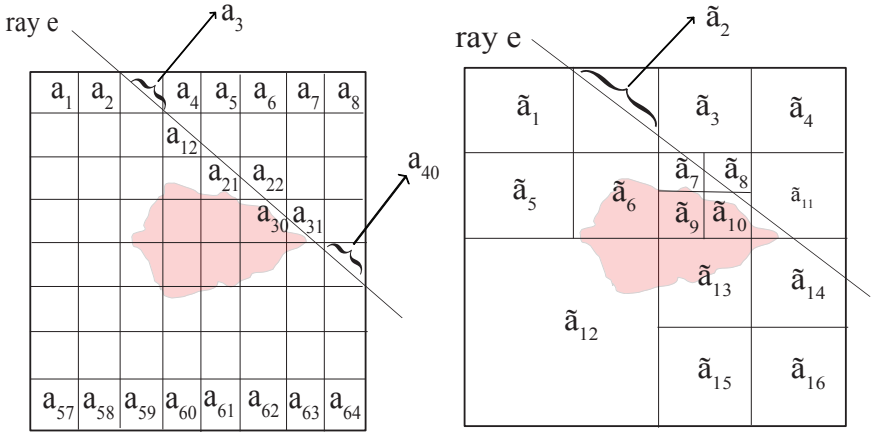
Algorithm 3 $\Delta \leftarrow$ TransMat(S)

```
for i ← root until all nodes do
   if(i ==leafNode(S))
      idx = getIndex(i)
      Δ(i,idx) = 1
   endif
endfor
```

Next we will show how we can apply AMR to our problem and derive equations that satisfy the travel time setup. Let $\tilde{A}$ denote the data kernel formed by irregular mesh and the coefficients in $\tilde{A}_{ek}$ denotes the length of e-th ray in k cell. Computing $\tilde{A}_{ek}$ solely by using ray tracing can be

computationally expensive especially because of the irregular geometry of the mesh. Therefore, to avoid that we obtain the relationship between ray lengths on irregular and regular grids, and it is given by,

$$\tilde{A}_{ek} = \sum_{j=1}^{N} \Delta_{kj} A_{ej} \tag{5.1}$$



(a) Regular Grid (b) Adaptive Grid

Figure (5.2) Relation between ray tracing in different grids

Figure 5.2 shows the relation between rays on two grids that is given by (5.1). In this, $\tilde{a}_2$ is obtained by summing $\{a_3, a_4, a_{11}, a_{12}\}$ whereas, $\tilde{a}_7$ to $\tilde{a}_{10}$ maintains the finest resolution as regular grid. This criteria of splitting is decided by Δ as discussed earlier and now the equation can be re-written in matrix form as,

$$\tilde{A} = A\Delta^T \tag{5.2}$$

Substituting (5.1) and (5.2) in (3.8) equation at each node becomes,

$$\begin{aligned} B_i &= A_i \Delta_i^T y \\ B_i &= \tilde{A}_i y \end{aligned} \tag{5.3}$$

where, B_i and Δ_i is the travel time vector and transformation matrix formed at node i respec-

tively. y is the new model vector on irregular grid and can be transformed to original grid by $x = \Delta_i y$

We summarize the entire process of adaptive mesh refinement using quadtree in Figure 5.3.

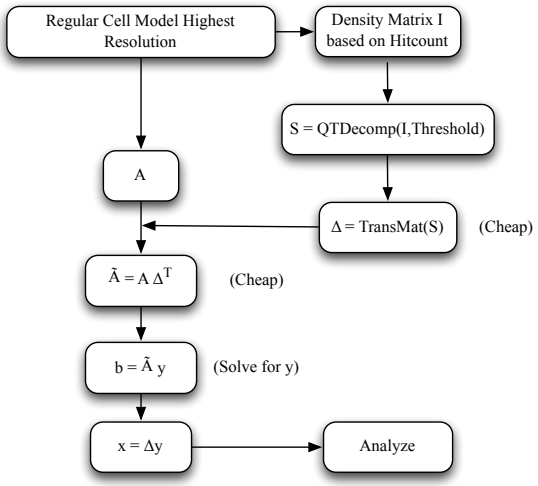
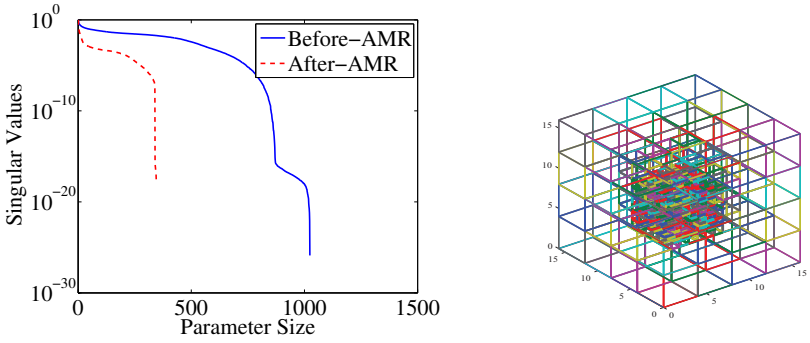


Figure (5.3) Flowchart of the mesh refinement process



(a) Singular values of A and $\tilde{A}$ (b) Octree based AMR

Figure (5.4) Effect of AMR on singular values and Octree based AMR

We applied adaptive mesh refinement on seismic tomography problem and we observed that it can improve the condition number of matrix A by removing the small singular values as shown in Figure 5.4(a). Figure 5.4(b) shows the adaptive mesh refinement on 3D problem using octree decomposition and have finer grids at the center of magma and becomes courser towards outside.

Remark 2 *Adaptive mesh refinement can be viewed as a non-trivial way of adaptive pre-conditioning as it decreases the condition number of the data kernel.*

The technique of selecting pre-conditioner using geometry of the problem and simple parametrization technique (quadtree/octree) is computationally less intensive. However, it should be noted that AMR relies on the threshold we choose and this requires domain knowledge. Also, if thresholds are not selected carefully it may result in removal of good singular values thereby leading us to different or bad solution. We will show the analysis of threshold sensitivity in the evaluation section. Until now we have seen the working of AMR for seismic tomography and we have shown how quadtree/octree based approach is suitable to run on sensor nodes. In the next section we will discuss how these transformed system of linear equation can be solved distributedly over a sensor network.

5.3 Distributed Tomography Inversion using Adaptive Mesh

In the previous section we have seen how to transform the system of linear equation formed over regular grid to a more well-condition system using irregular partitioning. Now applying (5.2) and (5.3) to (3.9) we get,

$$\mathbb{A}\Delta^T y = \mathbb{B}$$

$$\tilde{\mathbb{A}} y = \mathbb{B}$$

where,

$$\tilde{\mathbb{A}} = \begin{pmatrix} A_1\Delta_1^T \\ A_2\Delta_2^T \\ \vdots \\ A_p\Delta_p^T \end{pmatrix} = \begin{pmatrix} \tilde{A}_1 \\ \tilde{A}_2 \\ \vdots \\ \tilde{A}_p \end{pmatrix}; \mathbb{B} = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_p \end{pmatrix}; \tilde{A}_i \in \mathbb{R}^{m_i \times n_i}$$

From the above equation we see that at each sensor $i \in \{1, \cdots, P\}$ $A_i \in \mathrm{R}^{m_i \times n}$ gets transformed to $\tilde{A}_i \in \mathrm{R}^{m_i \times n_i}$. Note that during this transformation the number of rays i.e. m_i and right hand side B_i are unchanged while the number of grids are changed from n to n_i where $n_i \leq n$. At each sensor station the new linear subsystem formed after AMR is given by $\tilde{A}_i y_i = B_i$. The rows of this local subsystem contains rays and can be solved locally using row projection method like kaczmarz algorithm (1). The complete DT-AMR is given in algorithm (4). In this algorithm the initialization, ray tracing and adaptive mesh is performed only once and done simultaneously on each node. The steps that involves communication is highlighted in **bold**. Sending Δ_i to SINK in step 4 is done only once and can be done cheaply as Δ_i can be encoded efficiently. The actual communication in the network occurs in the line 5 and 7 of distributed tomography which involves aggregation of partial solution of size n_i from all the nodes and then broadcast averaged result back. The worst case communication cost for this given by $P \sum_i dim(x_i)$. Since this needs to be broadcasted back and algorithm converges after k iterations, the worst case communication cost is $2kP \sum_i dim(x_i)$. Although this cost may seem to be high, we will see in the next section that it is infact lesser than centralized methods where each row of $\{A_i, B_i\}$ needs to transferred. Moreover, the averaging over the networks can be done by communicating only with the neighbors and we leave this to the future work.

5.4 Evaluation

We evaluated the communication cost of DT-AMR algorithm using CORE network emulator [49]. We select CORE as the development and evaluation platform because the sensors that will be deployed on the real volcano will be some-low powered linux based devices such as android, beagle-bone or raspberry pi. Code developed in CORE emulator can be transferred to these devices without any modification. A network of 32 nodes are deployed which detects the event and traces the ray as shown in the Figure 4.1(b). We add Gaussian noise to the obtained travel time to model the receiver error. The finest resolution of dimension 32×32 is used as a regular grid to construct adaptive mesh using quadtree. The threshold for the hitcount is chosen to be 20. For the iterative methods, the selection of relaxation parameter ρ are critical and in all of our experiments

Algorithm 4 Distributed Tomography using Adaptive Mesh Refinement (DT-AMR)

Initialize

1: Number of seismic sensors P and AMR threshold T
2: Finest resolution dimension $Q = d \times d$ or $Q = d \times d \times d$
3: Initialize a $SINK$ node for aggregation.

Ray Tracing at each node i

1: Upon the detection of events
2: Perform ray tracing on each node simultaneously to obtain A_i and B_i

Adaptive Mesh at each node i

1: Obtain Density matrix I from A_i
2: S = QTDecomp(I,T) ; Δ_i = TransMat(S)
3: $\tilde{A}_i = A_i \Delta_i^T$
4: **Send(** Δ_i**)** $\rightarrow$ **SINK**

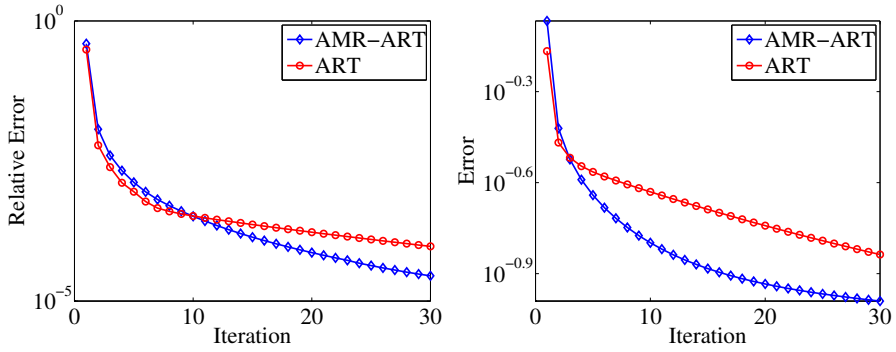
Distributed Tomography

1: $k \leftarrow 0$, $x^k \leftarrow 0$
2: **while** not converged **do**
3: In Every node i for $1 \leq i \leq P$ do in parallel
4: $\bar{x}_i \leftarrow$ ART($\lambda, \tilde{A}_i, B_i, x^k$,Iteration)
5: **Send(** $\bar{x}_i$**)** $\rightarrow$ **SINK to average**
6: $x^{(k+1)} = \left\{ \frac{1}{P} \sum_{i=1}^{P} \Delta_i \bar{x}_i \right.$
7: **Broadcast** $x^{(k+1)}$ **to all the node** P
8: $k \leftarrow k + 1$
9: **end while**
10: Update slowness model: $x = x^{k-1}$
11: TERMINATE

this remains constant throughout the iterations, i.e., $\rho^k = \rho = 0.25$ for all $k \geq 0$. Rate of convergence of different algorithms are compared using relative updates, $\phi = |x^{(k+1)} - x^{(k)}|/|x^{(k)}|$, residuals $\chi = \|Ax^k - b\|$ and absolute error $\epsilon = \|x^* - x^k\|$ where x^* is the ground truth.

Adaptive mesh has been applied on seismic tomography in a centralized setup earlier and has proven to perform better [44] [46]. However, quadtree based adaptive mesh for seismic tomography has been applied for the first time and we validate our approach using similar steps. Quadtree based AMR has been developed specifically to work in distributed environment and we do not expect it to perform better than centralized algorithms mentioned in [44] [46]. We perform quadtree based AMR on synthetic model and run ART which is a common centralized algorithm used for computing tomography. We see the advantage of AMR in Figure 5.5(a) and (b), where the relative and absolute error decreases significantly when AMR is used. AMR makes the system well

conditioned i.e reduce the zero eigenvalue and when the system is well-conditioned the solution obtained will be closer to the ground truth. From these tests we can conclude that ART with AMR is better and we can obtain better solution.

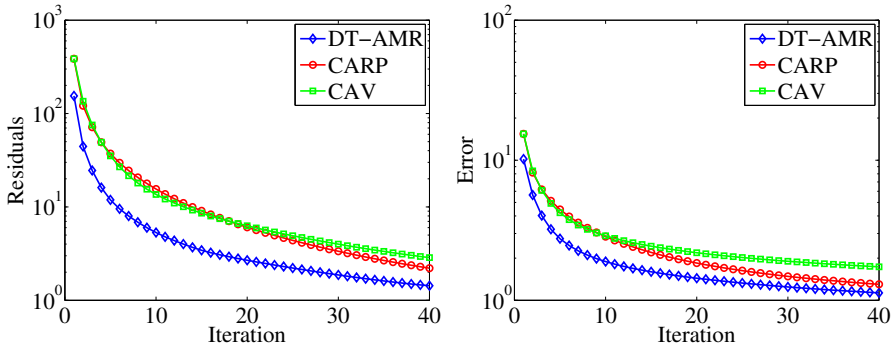


(a) Relative Error in ART (b) Absolute Error in ART

Figure (5.5) Comparing effect of AMR in Centralized ART

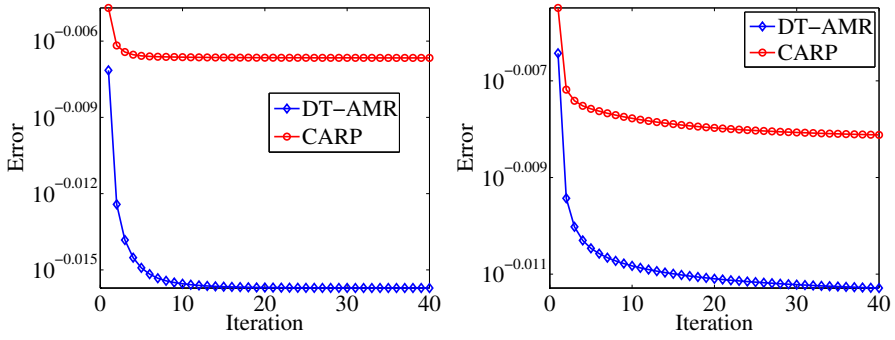
After validating the performance of our quadtree based AMR in centralized setup, we now perform series of experiment on distributed network. We compare our DT-AMR with standard distributed algorithms such as CARP and CAV. We use residuals and absolute error as the parameter for comparison and results are shown in Figure 5.6. These plots demonstrate that there is a difference in the initial convergence behavior in these algorithms both in-terms of residuals and absolute error. The final solution obtained from DT-AMR is also more closer to the ground truth and can be obtained with fewer iteration. The iteration on the x-axis represents the total number exchange of partial solution required during the intermediate step.

DT-AMR takes the advantage of partitioning the system of equation based on its resolving power at each node making it more well conditioned. The local computation on the well conditioned system on single node will accelerate the convergence and this is shown in Figure 5.7, where we analyze the performance on node 1 and 4. We observe that even at each node the partial solution obtained from DT-AMR is significantly better than CARP (Theorem 1) and this is the main reason for the improved performance of DT-AMR in distributed environment.



(a) Residuals (b) Absolute Error

Figure (5.6) Comparing effect of DT-AMR with CARP and CAV



(a) Absolute Error in Node 1 (b) Absolute Error in Node 4

Figure (5.7) Comparing effect of DT-AMR in single Node

CHAPTER 6

DISTRIBUTED ALGORITHMS VIA GOSSIPING

6.1 Introduction

Recently, methods like stochastic gradient descent (SGD), randomized coordinate descent (RCD) have received renewed attention for confronting very large scale problems, especially in the context of machine learning (ML) [7]. These methods are based on the computation of partial gradients involving random sampling of a subset of the entire system and have been proven to be better for noisy data due to random sampling. Randomized Kaczmarz (RK) is a type of SGD which has been recently popular for its exponential convergence rate with appropriate choice of rows for projection [12]. RK processes single row at a time and requires only $O(n)$ storage. For a extremely large sparse system of linear equations, RK is even more efficient than the conjugate gradient method [12].

In this section, we use $\mathcal{G}(\mathcal{V}, \mathcal{E})$ to denote undirected connected graph with node (sensor) set $\mathcal{V} = \{1, \cdots, P\}$ and edge set $\mathcal{E}$, where each edge $\{i, j\} \in \mathcal{E}$ is unordered pair of distinct node. Node i carries out communication only with its neighbors $\mathcal{N}_i = (j|\{i, j\}) \in \mathcal{E}$. Let $x \in \mathbb{R}^n$ be column vector and $x^k_{(i)}$ be the partial solution obtained at node i after k^{th} iteration for every $i \in \mathcal{V}$. Also, let x_j denote the j^{th} component of x. From Eq. (3.9) let $F(x) = \frac{1}{2}||Ax - b||_2^2$ and according to the gradient descent method the optimal solution x_* is obtained by traversing towards $-\nabla F(x^k)$ at every iteration $k = \{1, 2, \cdots\}$ by certain step size ρ_k starting from initial value x_0. In other words, $(k+1)^{th}$ iteration is given by, $x^{k+1} = x^k - \rho_k \nabla F(x_k)$.

Decentralized version of this problem recently developed by [23, 25], aims to minimize each nodes objective objective function $F_i(x)$ independent of other nodes. At every k^{th} iteration $\nabla F_i(x^k)$ requires the computation of $A_i^T(A_i x^k - b_i)$. Matrix multiplication becomes an issue for a large data set especially on a sensor with a very limited memory footprint. To avoid this, we use only partial gradients that involve sparse vector multiplication such as stochastic gradient descent (SGD).

Recently, methods like SGD are becoming popular in ML communities, and it involves random sampling to compute the gradient of a subsystem instead of an entire system. This method has been proven to be better for noisy data due to random sampling.

Randomized Kacmarz (RK) is a type of SGD commonly used to compute Eq. (3.9). This method starts with an arbitrary initial vector x^0 and at every iteration k, it randomly selects a row $i(k) \in \{1, \cdots .m\}$ of the linear system (with probability of choosing row i is $\frac{\|a_i\|_2^2}{\|A\|_F^2}$, where $||||_F$ denotes the frobenius norm). Next, it performs an orthogonal projection of the current estimate vector onto the hyperplane $a_{i_{(j)}}^T x_i = b_{i_{(j)}}$ as shown in Algorithm 5.

Algorithm 5 Randomized Kaczmarz Algorithm

1: **for** $k \leftarrow 0$ until convergence or max iteration **do**
2: Pick $i_{(k)} \in \{1, \cdots .m\}$ with probability $p_i = \frac{\|a_i\|_2^2}{\|A\|_F^2}$
3: $x^{(k+1)} = x^{(k)} + \rho_{i_{(k)}} \frac{b_{i_{(k)}} - \langle a_{i_{(k)}}, x^{(k)} \rangle}{\|a_{i_{(k)}}\|^2} a_{i_{(k)}}$
4: **end**

Recently, Strohmer and Vershynin [12] proved the following exponential bound on the expected rate of convergence of RK given by, $\mathbb{E}\|x_k - x\|_2^2 \leq (1 - \frac{1}{R})^k \|x_0 - x\|_2^2$ where $R = \|A^{-1}\|^2 \|A\|_F^2$, x_0 is an arbitrary initial value, while $\mathbb{E}$ denotes the expectation (over the choice of rows). Needell [40] studied the convergence of RK for the inconsistent system. RK and it's variants are inherently sequential and assume the availability of entire matrix A and vector b at a central location. This method cannot be directly applied to a loosely coupled system such as WSN and for this reason we first develop a parallel version of RK using fusion center.

6.2 Distributed Randomized Kaczmarz

Gossip methods are emerging as a new communication paradigm for large-scale distributed systems [13]. Some of the features that makes gossip methods attractive are: i) absence of central entity or coordinator node ii) high fault tolerance and robustness iii) *self healing* or error recovery mechanism [50] iv) efficient message exchange due to only neighbor communication v) provision for asynchronous communication. These interesting characteristics make them suitable for WSN

to carry out decentralized computation [51].

There are several variants of gossip algorithms designed specifically for tasks such as i) disseminate information [52], ii) compute sum/average [13] and iii) reach consensus [53]. The design of these algorithms vary slightly based on the communication pattern and also with the type of information exchanged at each iteration. For e.g., in push-sum [13], only one node wakes up at a time and exchanges information with another neighboring node whereas, in broadcast gossip [53] information is sent to all its neighboring node.

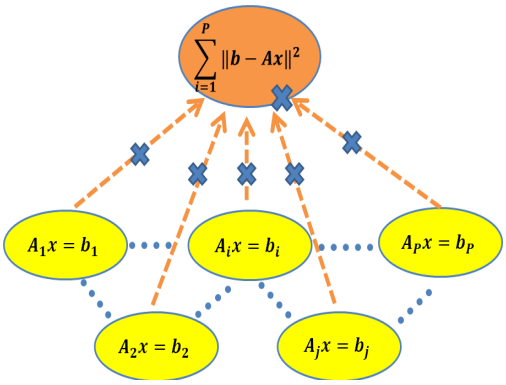


Figure (6.1) Gossip based Push Sum method where information is exchanged between neighbor without fusion center.

To design truly distributed method, we avoid the fusion center to compute average and replace it with decentralized methods such as push-sum [13]. In push-sum, when node i activates at t^{th} time slot, the following set of events occur: i) Node i sends its current state value x_i^t to neighboring node j. ii) Node j receives x_i^t and updates it in following way: $x_j^{t+1} = \frac{x_j^t}{2} + \frac{x_i^t}{2}$. iii) Node j sends x_j^{t+1} to i, where it updates $x_i^{t+1} = x_j^{t+1}$. iv) Remaining nodes update their value as: $x_\ell^{t+1} = x_\ell^t, \forall \ell \in \{\mathcal{V} - \{i, j\}\}$.

Now, if we denote $x^t \in \mathbb{R}^P$ a vector whose each component represents state of each node in network, then for every clock tick t we have, $x^{t+1} = W^t x^t$, where, W^t is a random matrix given by,

$$W_{ij}^t = \begin{cases} \frac{1}{P} & \{i, j\} \in \mathcal{E} \\ 1 - \frac{|\mathcal{N}_i|}{P} & i = j \\ 0 & \text{otherwise} \end{cases}$$

From the above weights, W^t exhibits following property: $W^t 1 = 1$ and $1^T W^t = 1^T$. Therefore, for every t, the iteration preserves the sums while vector of averages must be fixed point of iteration [10]. Next, we use the above gossip model and extend it to component-wise vector sum which will be used for computation of decentralized average.

Let $x^k_{(i)}$ denote intermediate solution of A_i, b_i at i^{th} node after k^{th} iteration. Also, let $x^k_{(i)j}$ denote j^{th} component of $x^k_{(i)}$. We define $X^t_j = (x^k_{(1)j}, \cdots, x^k_{(P)j})^T$, containing the j^{th} component of all the nodes. From the above gossip model we update the j^{th} component by $X^{t+1}_j = W^t X^t_j$. Similarly, we can extend this to all the component $j \in \{1, \cdots, n\}$. We denote this gossip scheme as push-vector.

Algorithm 6 D-RK Algorithm

1: **set** $x^0_\ell \in \mathbb{R}^n$ to an arbitrary value $\forall \ell \in \mathcal{V}$.
2: **for** $k \leftarrow 0$ until convergence or max iteration **do**
3: **for** each $1 \le l \le P$ in parallel **do**
4: $y_\ell = \mathrm{RK}(A^\ell, b^\ell, x^k_\ell, \rho_\ell)$
5: $\tilde{y}^0_{(\ell)} \leftarrow \quad y_\ell$
6: **end**
7: **for** $t \leftarrow 0$ until convergence or max iteration **do**
8: Node $i \in \mathcal{V}$ contacts $j \in \mathcal{N}_i$ and updates
9: $\tilde{y}^{t+1}_{(j)} = \tilde{y}^{t+1}_{(i)} = \frac{\tilde{y}^t_{(j)}}{2} + \frac{\tilde{y}^t_{(i)}}{2}$
10: **end**
11: $x^{(k+1)}_{(\ell)} = \tilde{y}^t_{(\ell)} \quad 1 \le l \le P$
12: **end**

Using the above definition of push-vector we propose the distributed randomized kaczmarz (D-RK) to solve linear equation over a decentralized system such as WSN. Algorithm 6 describes D-RK that combines component-wise gossip average with RK to solve a linear system of equations. In step 1 - 6 algorithm performs a certain iteration of RK simultaneously on all the nodes using its initial vector x^k. Step 7 - 10 of the algorithm describes push-vector. This algorithm is truly distributed and does not involve any fusion center. Push-vector continues until the relative update of the average is below a certain threshold.

6.3 Convergence Analysis

Lemma 2 *If $Q_i = \{x \in \mathbb{R}^n | A_i x = b_i\}$, $Q = \cap\{Q_i | i \in V\}$ and Eq. 3.9 has a solution $x^* \in Q$, then any sequence generated by Algorithm 6 converges to a fixed point $x^* \in Q$ for $\rho = 1$.*

Proof 2 *Since D-RK has similar structure as Par-RK except for averaging scheme, to prove the convergence, it is enough to show that push-vector is equivalent to row projection of RK. From above definition the update of z^{th} component of $x^t_{(i)}$ and $x^t_{(j)}$ is given by*

$$x^{t+1}_{(i)z} = x^{t+1}_{(j)z} = \frac{x^t_{(i)z} + x^t_{(j)z}}{2} \tag{6.1}$$

Let us assume $y^t = \{0, \cdots, x^t_{(i)z}, x^t_{(j)z}, \cdots, 0\}$ be a vector consisting of $x^t_{(i)z}$ and $x^t_{(j)z}$ at i^{th} and j^{th} position respectively. Also, assume a plane whose i^{th} and j^{th} components are related by $-m_i + m_j = 0$. Therefore, the coefficient $a = \{0, \cdots, -1, 1, \cdots, 0\}$, $\|a\|^2 = 2$ and $b = 0$. Now, from RK Algorithm 5 we have

$$\begin{aligned}
y^{t+1} &= y^t + \rho \frac{(b - \langle a, y^t \rangle)\, a}{\|a\|^2} \\
&= y^t - \rho \frac{\langle a, y^t \rangle a}{2} \\
&= (0, \cdots, x^t_{(i)z}, x^t_{(j)z}, \cdots, 0) - \frac{\rho}{2}(-x^t_{(i)z} + x^t_{(j)z}) \\
&(0, \cdots, -1, 1, \cdots, 0) \\
&\textit{for } \rho = 1 \textit{ we have,} \\
&= (0, \cdots, \frac{x^t_{(i)} + x^t_{(j)z}}{2}, \frac{x^t_{(i)z} + x^t_{(j)z}}{2}, \cdots, 0)
\end{aligned}$$

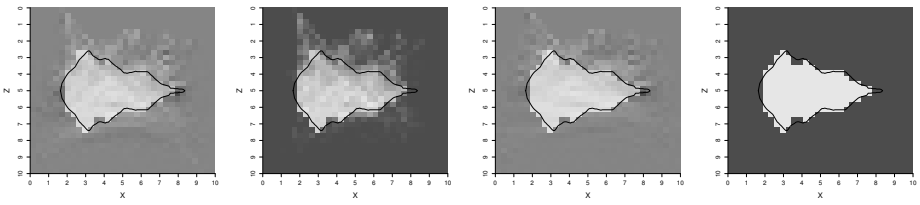
This is equivalent to expression in Eq. (6.1). Similarly, we can extend this argument for every component $z \in \{1, \cdots, n\}$ of the vector. Hence, push-vector updates are equivalent to row projection of random kaczmarz for $\rho = 1$.

Push-vector in D-RK performs gossip only with one neighboring node at any time slot t. In a

wireless sensor network, each node has an advantage of inherently broadcasting the messages to its neighbor within a certain radius. Now at the cost of one transmission, a node can gossip with all of its neighbors, and this has been studied under broadcast gossip [53].The broadcast gossip, however, converges to a consensus rather than an average as the weights used there does not preserve the sum (i.e., $1^T W \neq 1^T$). This will affect the convergence analysis of D-RK and will be studied in future.

6.4 Evaluation

In this set of experiments, we intend to demonstrate the correctness of our algorithm through visualization. We stopped the algorithm when its relative update $\phi \leq 0.001$. Fig. 6.2 shows the result slice by slice along the X and Y axes and Fig. 6.2(d) we have the ground truth. Each row of the figure shows the same tomography slice on some layer along with X or Y axes (the total layers of each figure is equal to the resolution dimension of the result). The black polygon gives the cross section outline of the surface of magma area represented in Fig. 4.1(a). From this experiment, we can see that both Par-RK and D-RK were able to generate tomography image almost similar to centralized RK. From the detailed examination of the Fig. 6.2 we can say that, Par-RK produces fewer artifacts near the boundary whereas, D-RK has sharper differences. We believe this is due to the fact that Par-RK calculates true average unlike D-RK. This evaluation suggests that both Par-RK and D-RK can be a good candidate for distributed tomographic inversion.



(a) Par-RK (b) D-RK (c) RK (d) Ground Truth

Figure (6.2) 2D vertical slice of 3D Tomography using Synthetic Data. The row represents slice 18 of 32 of different algorithms.

Next, we compare the performance of the proposed algorithm in terms of the relative update.

We compare Par-RK with algorithms like Cimmino, CAV, and DROP, which are all algorithms used to solve the system of linear equations using fusion center. On the other hand, we compare D-RK with decentralized algorithms such as EXTRA [25] and DGD [24] and results are shown in Fig. 6.3. From Fig. 6.3(a) we can infer that Par-RK performs better than other parallel methods in terms of convergence. This is due to the faster convergence rate of RK method compared to other methods. In case of D-RK (Fig. 6.3(b)) we see that it is faster than DGD, however, slower than EXTRA. In EXTRA, an optimal step size is calculated to accelerate the convergence. It should be noted that, Par-RK has faster convergence compared to D-RK and is because of the faster mixing of the partial solutions at the expense of fusion center and multi-hop aggregation scheme.

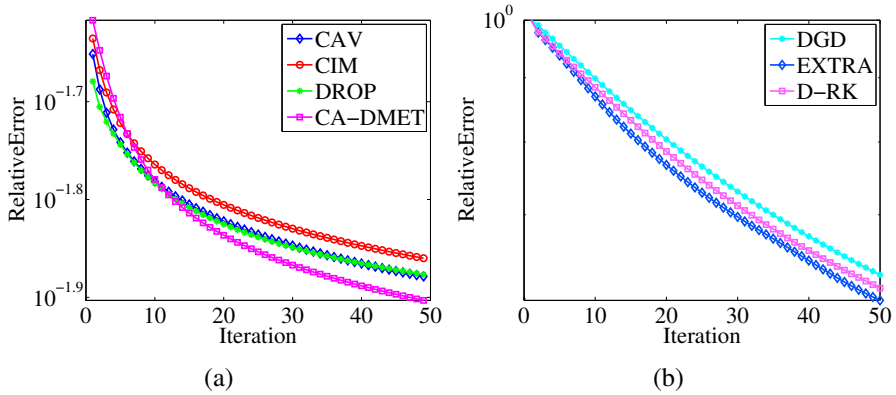
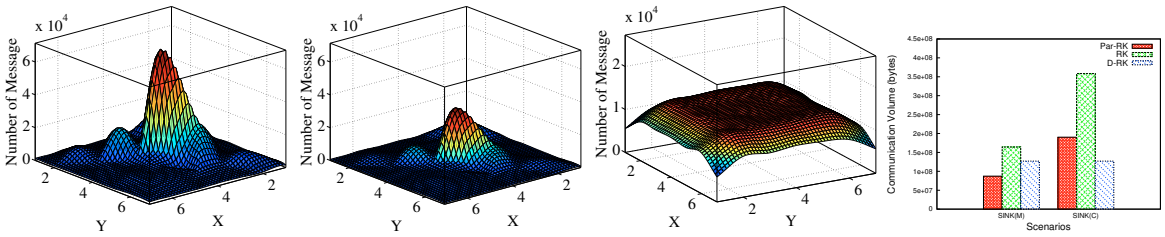


Figure (6.3) Comparison of (a) Par-RK and (b) D-RK with different parallel and distributed algorithms.

In this section, we compare the communication cost of the centralized algorithm with proposed distributed algorithms in terms of number of messages exchanged to reach the solution. Here for centralized and parallel case SINK(M) and SINK(C) refers to sink (fusion center) node placed in middle and at the corner respectively. From Fig. 6.4(a) we can see that communication cost in a centralized setup is high near the SINK as all the ray information is transferred over the network to sink before the computation. In this case, the volume of data is proportional to the number of earthquakes and also number of stations. Fig. 6.4(b) shows the communication pattern for Par-RK and from this we can see that the communication cost is lesser compared centralized scheme (RK). This is mainly because communication cost in Par-RK depends on number of it-

eration and typically with the semi-convergent property of iterative methods [35] the number of iteration is much less compared to the number of earthquake events.

In Fig. 6.4(c) we present the communication pattern of D-RK, which is flat compared to Par-RK and RK. Neighbor gossip helps us to balance the load in the network while avoiding other overheads such as routing, etc. It should be noted that due to the slower convergence of D-RK compared to Par-RK, a larger volume of packets will be exchanged in the entire process. This is verified from Fig. 6.4(d) where we compare volume (bytes) transferred in all the three settings. Fig. 6.4(d) shows that in case of SINK(M) volume of bytes transferred Par-RK is lower than D-RK, which is due to the slower convergence as mentioned earlier. However, placing SINK(C) at the corner increases the communication cost of the Par-RK, and this is due to packet loss caused by increased congestion. It should be noted that D-RK has no effect on the placement of SINK node as it communicates only with the neighbors..

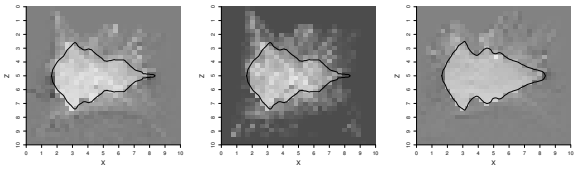


(a) RK (M) (b) Par-RK (M) (c) D-RK (d) Communication Volume

Figure (6.4) Communication Cost

In the next set of experiments, loss tolerance and robustness of proposed algorithms are evaluated. The algorithm runs with the same configuration for a packet loss ratio of 20% in the emulator. Fig. 6.5 gives part of the 2D slice rendered along Y axes with packet loss. We can see that in Par-RK and D-RK with 20% packet loss there is no significance difference in terms of the magma area outline when compared to the results with no packet loss Fig. 6.5(c). Since the computation is distributed, and all the nodes are involved in slowness calculation, the proposed algorithm is tolerant to a severe packet loss. A closer look at the result tells that packet loss in Par-RK has a slightly larger effect compared to D-RK. This can be seen in the visualization result where D-RK can get a

sharper image than Par-RK. This result can be attributed to the truly distributed communication as opposed to fusion center in Par-RK. Loss of packet near fusion center has a profound effect on the tomography result, whereas D-RK can tolerate such single point failures.



(a) Par-RK 20% Loss (b) D-RK 20% Loss (c) No Loss

Figure (6.5) Robustness of distributed algorithms in terms of packet loss.

CHAPTER 7

ASYNCHRONOUS DECENTRALIZED ALGORITHMS VIA RANDOMIZATION

7.1 Introduction

Many real-world applications arising in domains such as distributed control [6], large-scale machine learning [7], wired and wireless networks [8], can be formulated as distributed linear least-squares over a large network. These problems have data naturally distributed among various nodes e.g peer-to-peer or sensor network. Today, due to the large volume of the data, we face a challenge to process it in real-time. Therefore, we are particularly interested in solving such problems in a distributed way, with each node handling its local component. This gives rise to novel techniques commonly referred to as *decentralized* algorithm that do not require a fusion center, cluster heads, or multi-hop communication.

Designing a decentralized algorithm presents three main difficulties. First, the underlying hardware in the distributed networks are heterogeneous and often unreliable. Individual sensors may fail at any time, and the communication network that connects them could be highly unstable. Second, the on-board processor often has limited memory, computational power and energy that restricts us from using basic linear algebra tools like SVD, matrix-matrix or matrix-vector multiplications for local computation. Third, due to the large number of nodes and the volatility of the network, any reliance on central coordinator will limit the systems scalability and the performance. The decentralized algorithm must be fault-tolerant as node and link failures are a common occurrence in such systems [54].

From the recent trends in big data optimization, we see a renewed interest in randomized (stochastic) algorithms both for computation [9] and communication [10]. Today, stochastic methods can solve data-intensive problems on a inexpensive hardware at a faster rate compared to the deterministic methods [11]. For instance, Randomized Kaczmarz has linear convergence rate and outperforms the traditional the conjugate gradient method in some cases [12]. Even in network

communications, randomized methods such as gossip protocols are emerging as a new communication paradigm for decentralized systems [13]. These methods guarantee convergence in expectation (probabilistic) similar to stochastic methods; achieve high stability under disruptions, and scale gracefully to a large number of nodes. In comparison, traditional communication techniques have certain guarantees, but are unstable or fail to make progress during periods of even modest disruption.

Motivated by these trends, we present an asynchronous algorithm that combines two randomized techniques, i) Randomized Kaczmarz (RK) [12] (computation) with ii) Asynchronous Random Gossip [10] (communication), in order to solve a large system of linear equation distributed over a network (3.9). We call this asynchronous method *Gossip Randomized Kaczmarz* (GRK). To our best knowledge, our work is the first attempt to solve Eq. (3.9) using stochastic methods both for computation and communication. Our contributions are threefold. First, we present a novel asynchronous algorithm and provide a theoretical convergence. Second, we evaluate the performance of GRK on a real sensor testbed and show that when the processors are not homogeneous the proposed method converges faster in terms of time than its synchronous counterpart [24], [25]. The empirical results also show that GRK has linear convergence for the consistent system and is robust to network failures. Third, we demonstrate the practicality of the GRK by applying it to a real-world problem arising from seismic imaging [55].

7.2 Gossip Randomized Kaczmarz

We consider a network of N nodes connected to form an arbitrary topology given by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V} = \{1, \cdots, N\}$ and edge set $\mathcal{E}$ that contains set of links in the network. We have $\{i, j\} \in \mathcal{E}$ if node i and node j can communicate with each other. $\mathcal{N}_i$ denotes the neighbor set of node i. Let $x \in \mathbb{R}^n$ in Eq. (3.9) be a column vector, and $x_i^k \in \mathbb{R}^n$ be the local copy held privately by node i and the superscript k denotes the iteration number, which is also proportional to total messages exchanged. Also, let $x_{i_{(j)}}$ denote the j-th component of vector x_i and $A^\intercal$ the transpose of A.

We reformulate Eq. (3.9) as a decentralized optimization problem where the networks objec-

tive is to solve the following minimization problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) = \frac{1}{2N}\sum_{i=1}^{N} \|A_i x - b_i\|_2^2 \tag{7.1}$$

We assume that the function $f_i(x) = \frac{1}{2}\sum_{i=1}^{N} \|A_i x - b_i\|_2^2$ is privately known only to node i and the local system is large and over-determined ($m_i \lll n$). Now due to the large sub-system, it becomes prohibitive to compute the full-gradient $\nabla f_i(x)$. Therefore, at each node we use stochastic gradient locally given by $\partial f_i^{\ell} = (\langle a_j, x_i\rangle - b_j)a_j/\|a_j\|_2^2$, where j is a set of ℓ random rows uniformly chosen from $\{1, 2, \cdots, m_i\}$ with replacement. The goal now is to solve Eq. (7.1) using an algorithm that is asynchronous and distributed.

Remark 3 *The stochastic gradient ∂f_i^{ℓ} is equivalent to a projection onto ℓ random hyperplanes as in Randomized Kaczmarz. The step 5 of RK is equivalent to stochastic gradient update $x^{k+1} = x^k - \alpha\partial f_i^{\ell}(x^k)$ for $\alpha = \ell = 1$.*

Asynchronous algorithms are easier to analyze using a single virtual clock as given in [10]. In this model, each node ticks according to a Poisson clock and the virtual clock is incremented when a local clock ticks. Thus the virtual clock ticks according to Poisson process with rate N. Let Z^k denote the k-th tick (iteration) of the virtual clock for a local tick at node p. Let node $q \in \mathcal{N}_p$ be the random node chosen at k-th iteration. Let $x_p^{(k-1)}$ denote local copy of x held by node p during the iteration $k-1$ i.e. immediately before Z_k.

Consider nodes p and q at k-th iteration. Gossip Randomized Kaczmarz (GRK) at k-th iteration follows two main steps:

- Exchange the vector x_p^k and x_q^k to perform a gossip update give by $\bar{x}_{pq}^k = ((x_p^k + x_q^k)/2$
- Update the current iterate $\bar{x}_{pq}^k$ in node p and q by adjusting it along the negative direction of $\partial f_p(\bar{x}_{pq}^k)$ and $\partial f_q(\bar{x}_{pq}^k)$ respectively. Mathematically, the iterates evolve according to,

$$x_i^{k+1} = \begin{cases} \bar{x}_i^k - \partial f_i(\bar{x}_i^k) & \text{if } i \in \{p, q\} \\ x_i^k & \text{otherwise,} \end{cases} \tag{7.2}$$

Now for N nodes over a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, Gossip Randomized Kaczmarz is given by the Algorithm 7

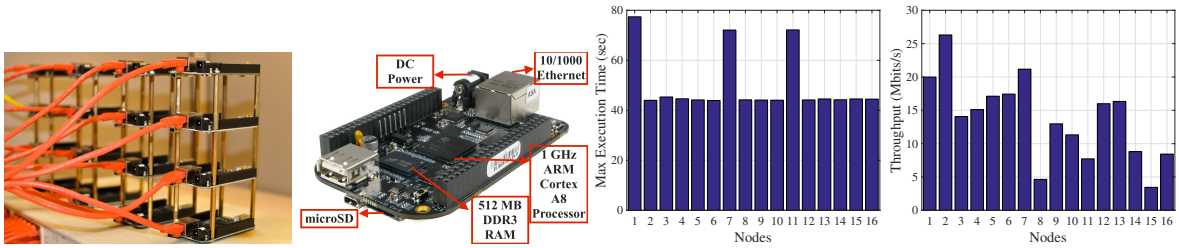
Algorithm 7 Gossip Randomized Kaczmarz

set $x_p^0 \in \mathbb{R}^n$ to an arbitrary value $\forall p \in \mathcal{V}$.
for $k \leftarrow 0$ until convergence or max iteration **do**
    Pick uniformly at random a node $p \in V$
    Node p now selects node $q \in \mathcal{N}_p$
    Node p and q exchange x_p^k and x_q^k
    Node p sets: $\bar{x}_{pq}^k \leftarrow (x_p^k + x_q^k)/2$
    Node q sets: $\bar{x}_{pq}^k \leftarrow (x_q^k + x_p^k)/2$
    Node p sets: $x_p^{k+1} \leftarrow \bar{x}_{pq}^k - \partial f_p(\bar{x}_{pq}^k)$
    Node q sets: $x_q^{k+1} \leftarrow \bar{x}_{pq}^k - \partial f_q(\bar{x}_{pq}^k)$
end

7.3 Evaluation

We provide empirical results studying the performance of asynchronous gossip randomized Kaczmarz using simulated datasets on a real sensor testbed. We first validate our theoretical results such as consensus and convergence. We later compare the performance of asynchronous method GRK with its synchronous counterpart DGD [24] and EXTRA [25]. Our results indicate that the proposed method performs similar to its synchronous counterpart in all cases and even outperforms them on a grid topology with respect to execution time and number of message transmitted. We also demonstrate the robustness of GRK by simulating link and node failure.

The experimental testbed consists of 16-BeagbleBone Black (BBB) that are connected using a switch forming a cluster. Fig. 7.1(a) shows the testbed setup where BBB's are stacked in a rack fashion. Each BBB comes with Angstrom OS (linux) and has 512MB DDR3 RAM, 16GB flash storage, 1 GHz ARM Cortex A8 processor (Fig. 7.1(b)). They are credit card sized, low-power computing units that are under $50. Unlike, High Performance Computing Cluster, the cluster developed using BBB allows us make a fair evaluation of the decentralized algorithms. This cluster also gives us the flexibility to configure the topology based on the physical environment and simulate link and node failures.



(a) (b) (c) (d)

Figure (7.1) (a) Testbed consisting of 16 Beaglebone black connected using a switch. (b) Beagle-Bone black hardware details (c) CPU benchmark using Sysbench to evaluate the execution time across cluster nodes. (c) Network benchmark using iPerf to evaluate the throughput across nodes. We see that though each node has the same hardware, we see a significant difference both in terms of execution time and throughput.

Before performing the experiments, we evaluate the performance of our cluster using standard benchmarking tools such as *sysbench* [1] and *iPerf* [2]. Sysbench CPU test uses a standard prime number generation algorithm to evaluate the execution time. Fig 7.1(c) shows the execution time taken across 16 nodes. Although several nodes took similar time to execute, some nodes (e.g. 1,7,11) took slightly longer time. The exact cause of this behavior is unknown, however, we suspect the unstable power via usb would have effected the cpu clock. We also evaluate the network throughput (Mbits/sec) using iPerf tool and from Fig. 7.1(c) we see that it varies significantly across the nodes. The cluster has variations both in computation and communication time and exhibits the heterogeneity and unreliability of the network in general. These characteristics of the cluster makes it suitable to evaluate the decentralized methods.

The communication scheme in GRK was implemented using a UDP protocol. The connection less protocol reduces the communication overhead and allows scalability over different topology. In GRK, a node wakes up at random and sends its current iterate to one of its neighbor. It then starts a timer and waits for neighbors response until timeout. During this process the gossip is called successful (SUCCESS) only if the node receives the response (iterate) from the neighbor within the timeout. Otherwise the gossip is termed as unsuccessful (FAIL). A FAIL can happen if

[1] `https://launchpad.net/sysbench`
[2] `https://iperf.fr/`

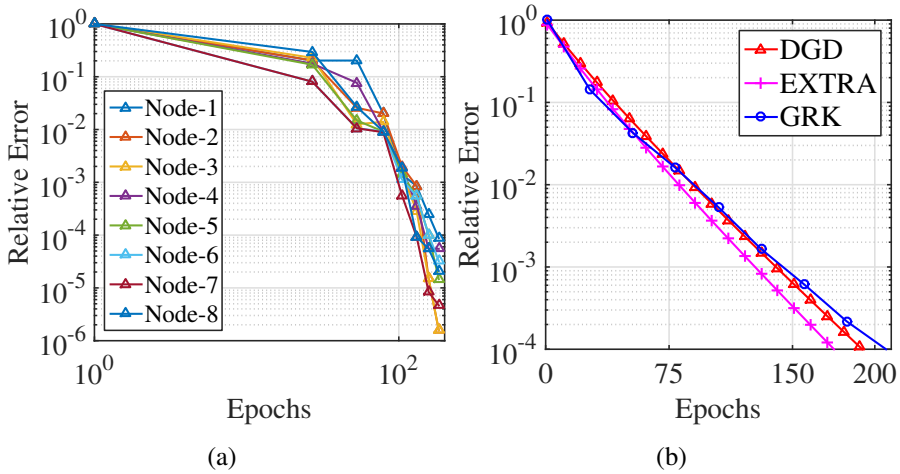


Figure (7.2) (a) Illustrates the convergence of eight random nodes to a consensus value. (b) Comparing relative error of GRK with its synchronous counterpart DGD and EXTRA. Graph shows that DGD, EXTRA and GRK took 191, 176 and 210 epochs to reach a threshold $\delta < 10^{-4}$ respectively. Although GRK took more number of epochs, the total execution time for GRK (220 sec) was less than DGD (261 sec) and EXTRA (255 sec).

the solicited neighbor is either in a compute state (Randomized Kaczmarz) or is communicating with some other neighbor (BUSY). Unreliable link or node failure can also cause the gossip to fail. The randomness in wake up time is simulated using a random sleep after each SUCCESS. In case of synchronous decentralized method such as EXTRA and DGD, we use a more reliable TCP connection that guarantees successful communication and reduces synchronization overhead. Unlike GRK, EXTRA and DGD communicates with all its neighbor at each round. By default all the experiments are performed over a grid topology unless specified. Note that achieving synchronization in a actual wireless network is non-trivial and involves significant overhead.

We generated dense dataset of size $A \in \mathbb{R}^{8000\times256}$, $b \in \mathbb{R}^{8000}$ following a standard normal distribution. We then split $\{A, b\}$ into 16 equal blocks row-wise and assign them to each node. We denote $x^* = A^\dagger b$ as the ground truth and $\bar{x}^k = \frac{1}{N}\sum_{i=1}^{N} x_i^k$ as the mean value of the network at each epoch k, where one epoch in GRK corresponds to nodes waking up at least once. Performance of the algorithms are compared using the relative error ($\delta = |\bar{x}^k - x^*|/\|x^*\|$). We use $\delta < 10^{-4}$ as the stopping criteria throughout unless mentioned.

Here we validate our theoretical result of node consensus. Fig. 7.2(a)) shows the relative error

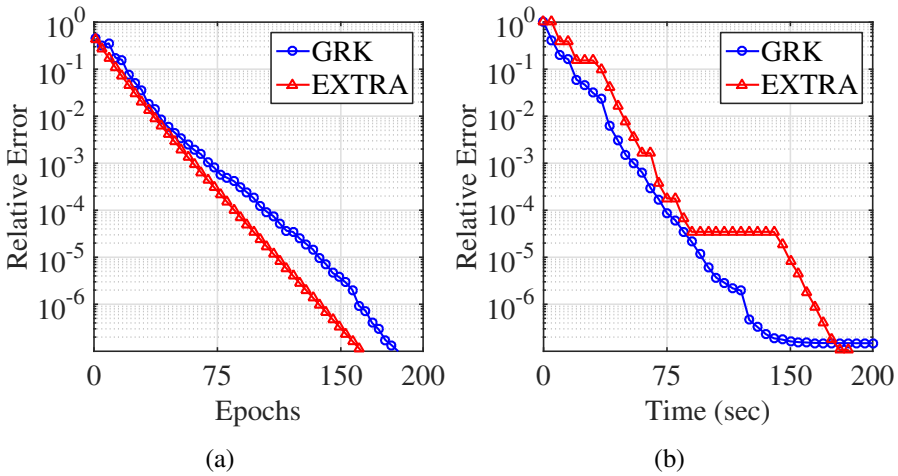
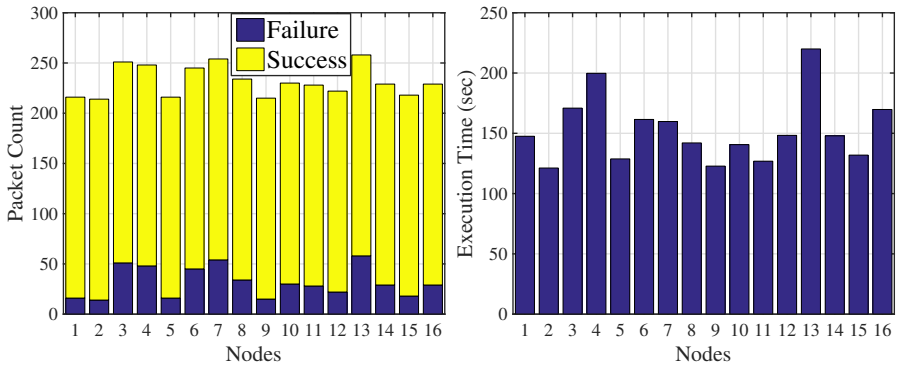


Figure (7.3) Comparison of GRK (asynchronous) with EXTRA (synchronous) with respect to (a) Epochs (b) Execution time. Number of epochs is more in case of GRK, however, it takes less time due to asynchronous communication.

of eight random nodes and we see that that the relative error at each node decreases and the iterates reach to a consensus. Next we compare the performance of GRK with DGD and EXTRA. From Fig. 7.2(b) we see that the GRK has a linear convergence like EXTRA and DGD. We also observe GRK takes 210 epochs, more than DGD (191) and EXTRA (176), however, the total execution time of GRK (220 sec) is less compared to DGD (261 sec) and EXTRA (255 sec). The slower convergence in terms of epochs is due to slower mixing of the value. The speedup in the execution time is due to reduced communication overhead and synchronization. In the next section we will compare this characteristic in detail.

To further investigate the performance of synchronous vs asynchronous in terms of execution time, we carry out another experiment. Since EXTRA and DGD have similar communication characteristic here we compare GRK only with EXTRA. From Fig. 7.3(a) we again see that GRK takes more epochs to reach the same error threshold, but takes lesser time than EXTRA to finish the epochs Fig. 7.3(b). In synchronous algorithm such as EXTRA, the execution time is governed by the slowest node and in Fig. 7.3(b) we clearly see that at around 100th second, EXTRA briefly stops for about 40 seconds before it resumes. Further investigation of the logs revealed that one particular node (10) was not able to receive an update from its neighbor node (11) and blocked the



(a) (b)

Figure (7.4) (a) Unsuccessful Gossip (FAIL) across all the nodes for 200 epochs (Success). (b) Per Node Execution time (sec). The variations in execution time is due to the heterogeneity property of the cluster.

	GRID		KN	
	epoch	mesg	epoch	mesg
EXTRA	176	11264	30	**6144**
DGD	191	12224	320	81920
GRK	210	**7414**	187	8912

Table (7.1) Effect of topology on decentralized algorithms

progress of all the other nodes. Because of asynchronous characteristic of GRK it does not wait on all the nodes across the network making the algorithm robust.

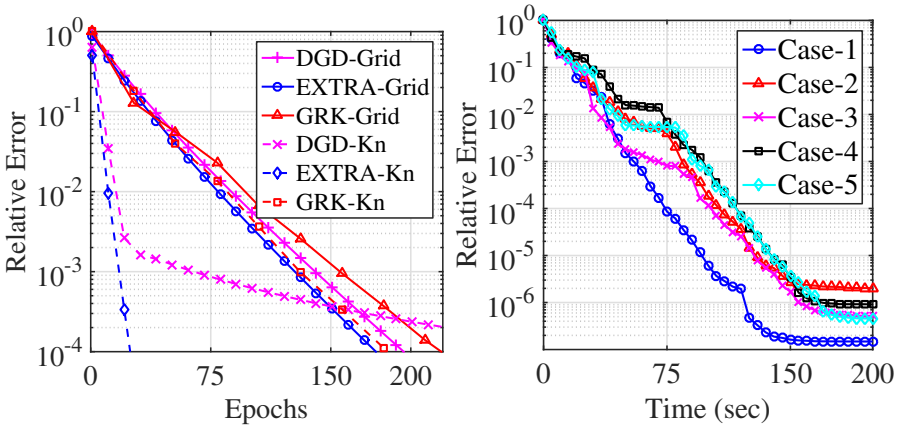
Communication in asynchronous algorithm plays a important role in evaluating the performance of the algorithm. Inefficient protocol implementation can lead to large number of packet loss (FAIL). Fig. 7.4(a) shows the number of Success vs Fail across all the nodes. From the graph we observe that to complete 200 epochs, GRK has around 10-15% of FAIL. As mentioned earlier, gossip FAIL can be due to nodes BUSY state or a faulty link. This is the limitation of the GRK algorithm, where it requires two nodes to be partially synchronized. In future, we plan to explore other gossip methods such as broadcast [53] or one sided gossip, which eliminates the need for partial synchronization. In Fig 7.4(b) we plot the per node execution time in seconds. We observe that, each node takes different execution time to reach the same solution. This can be attributed to the heterogeneity property of the cluster as shown in benchmarking.

The convergence rate of decentralized methods rely heavily on the property of the mixing ma-

trix, which in turn depends on the spectral properties of the underlying topology. In this section, we report the performance of EXTRA, DGD and GRK on two different topologies i.e Grid and a complete graph (Kn). From Fig 7.5(a) we see that EXTRA converges extremely fast (30 epochs) on a Kn, whereas DGD takes around 320 epochs slower than on the grid. This is similar to the observation made by the authors in [25]. We see that GRK is also effected by the topology, however the effect is not significant compared to EXTRA and DGD. Notice that for a grid topology GRK requires fewer transmissions for reaching the solution. This is again because of the fact that the EXTRA/DGD communicates with all its neighbors, whereas, GRK exchanges only with one random neighbor. In Kn, due to the faster convergence of EXTRA, it requires less messages to be transmitted. Another interesting phenomena to notice is that, although GRK requires less epochs in Kn (187) compared to the Grid (210), it transmits more messages in Kn. Our investigation showed that GRK in Kn had more FAIL due to increase in its neighbor. From this we can conclude that asynchronous decentralized algorithms are more suitable for sparse topology whereas, synchronous methods are proffered on a reliable dense network.

Remark 4 *In a grid topology, although GRK takes more epochs, it outperforms EXTRA and DGD in term of execution time and message transmitted. This makes it suitable for decentralized systems such as sensor networks.*

The GRK algorithm must be fault-tolerant as the node or link failures in decentralized systems are norms rather than the exceptions. Here we validate the performance of the algorithm by simulating the node and link failure. We run the algorithm under five different cases. We run each experiment for 200 seconds and start the node failure after 50 seconds from start. **Case 1)** No failure **Case 2)** 25% nodes fail for 10% time **Case 3)** 25% nodes fail for 15% time **Case 4)** 50% nodes fail for 10% time **Case 5)** 50% nodes fail for 15% time. Fig. 7.5(b) shows that the algorithm runs successfully irrespective of the percentage of node failure or the time. This characteristic is very important in case of unreliable system such as wireless sensor network. The convergence however is impacted due to the unreliability which is expected. In case 4 and 5 since 50% of the node failed, the update across the network stalled (flat line after 50 sec until 80 sec) for a while before it resumed. We did not observe this behavior with 25% node failures (case 2 - 3).



(a) (b)

Figure (7.5) (a) Performance of algorithms on a Grid and a complete graph (Kn) topology. (b) Performance of GRK under different cases of node and link failure. Case-1) No failure Case-2) 25% nodes fail for 10% time Case-3) 25% nodes fail for 15% time Case-4) 50% nodes fail for 10% time Case-5) 50% nodes fail for 15% time

CHAPTER 8

CONCLUSIONS

This thesis presents a suite of algorithms suitable for distributed/decentralized computing in WSN. As an application we present the real-time seismic imaging framework made possible via the proposed algorithm. We saw that today, traditional sensor that was only responsible for sensing is rapidly being replaced by next generation devices called as Internet Of Things. These are more sophisticated devices that not only sense but also performs computing not only using its own measurements but also using other neighboring devices. We also saw that the sensing capability of these devices has drastically increased and each device today can store several GB of data.

In the seismic application we showed that transferring few GB of data from each sensor to a central repository is prohibitive. To solve this data transfer, we first proposed a set of Parallel algorithms. Here the main theme is that the computation occurs within each sensor nodes and at every epoch, the partial solution is transferred to a central repository. The partial solutions are sparse and requires few KB compared to GB of data. At the central repository, these partial solutions are combined with partial solutions from other sensors. The combined solution is again sent to sensors which then uses it as the initial guess for next epoch. We showed that a convex combination of these partial solutions is sufficient to guarantee the overall convergence. Our results on real seismic data was promising and it confirmed that such techniques are indeed feasible for real world scenario.

We later studied several techniques to accelerate the convergence of the parallel algorithms. Among them was adaptive mesh refinement, where the preconditioning was done by carefully selecting the mesh structure etc. Adaptive mesh refinement was applied at each sensor which was inherently distributed. These preconditioned data helped the solution to converge faster than previous proposed scheme. Moreover, using this adaptive mesh refinement we could adaptively

zoom in on any image region to obtain higher resolution.

Although adaptive mesh was able to accelerate the convergence, we still used a central repository to merge the intermediate solution which was a bottleneck and prevented scaling. To overcome this, we explored gossip based methods to merge intermediate result. We restricted communication of the nodes only to its neighbors and performed the convex combination at each node. Through the concept of gossiping we could show that gossiping at every iteration followed by gradient descent at each node converged to the same solution as parallel algorithm. This was the first distributed algorithm proposed for solving seismic imaging. Our experiments in real-data also proved the above hypothesis.

Once we had gossip based setup working, we explored other alternatives to reduce the overall computation and communication load on sensor network. Earlier gossip method proposed was synchronous in nature i.e each sensor exchanged information with their neighbor at the same time. This required a centralized clock to coordinate this action. When network scaled obtaining this scheme to work was challenging. To avoid this, we explored other options such as asynchronous methods. In this method, each node woke up at random time and sent information to its neighbor. Once neighbor received sufficient information it combined it with available intermediate results. This process continued and we showed that this strategy indeed converged. This work was also tested on real testbed with seismic data and we obtained satisfactory result.

Asynchronous algorithms are very effective methods right now for seismic imaging.. Several other works on asynchronous methods such as adaptive delay, adaptive gradient etc have been proposed and widely used. We learned that the most effective way to perform analytics today in sensor network is to send all the data to cloud. But this architecture will soon fail as there are too many several devices that connect to internet. Another solution is to perform most of the heavy weight calculation at each node and send only small amount of information to central repository. This is both faster and efficient. Today, amazon echo and apple siri uses similar architecture. If the data being recorded are very high then decentralized methods are preferred to do computing. In future, we plan to explore several different techniques in each of these categories and build a computational framework for IoT.

REFERENCES

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *The First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[2] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Wireless sensor networks for structural health monitoring," in *Proc. 4th ACM conference on Embedded networked sensor systems (SenSys)*, Nov. 2006.

[3] H. B. Lim, Y. M. Teo, P. Mukherjee, V. T. Lam, W. F. Wong, and S. See, "Sensor Grid: Integration of Wireless Sensor Networks and the Grid," in *Local Computer Networks*, Nov. 2005.

[4] G. Kamath, L. Shi, and W.-Z. Song, "Component-Average Based Distributed Seismic Tomography in Sensor Networks," in *The 9th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2013, pp. 88–95. [Online]. Available: http://dx.doi.org/10.1109/DCOSS.2013.17

[5] W.-Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. Lahusen, "Air-dropped Sensor Network for Real-time High-fidelity Volcano Monitoring," in *The 7th Annual International Conference on Mobile Systems, Applications and Services (MobiSys)*, Jun. 2009.

[6] A. Nedić and A. Ozdaglar, "Distributed Subgradient Methods for Multi-Agent Optimization," *Automatic Control, IEEE Transactions on*, vol. 54, no. 1, pp. 48–61, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1109/tac.2008.2009515

[7] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Physica-Verlag HD, 2010, pp. 177–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-7908-2604-3_16

[8] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling," *Automatic Control, IEEE Transactions on*, vol. 57, no. 3, pp. 592–606, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1109/tac.2011.2161027

[9] M. W. Mahoney, "Randomized Algorithms for Matrices and Data," *Found. Trends Mach. Learn.*, vol. 3, no. 2, pp. 123–224, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1561/2200000035

[10] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006. [Online]. Available: http://dx.doi.org/10.1109/tit.2006.874516

[11] F. Niu, B. Recht, C. Rã©, and S. J. Wright, "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent," in *In NIPS*, 2011. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.229.3698

[12] T. Strohmer and R. Vershynin, "A randomized Kaczmarz algorithm with exponential convergence," *J. Fourier Anal. Appl*, vol. 15, pp. 262–278, 2009.

[13] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '03, 2003, pp. 482–491.

[14] G. Kamath, L. Shi, W.-Z. Song, and J. M. Lees, "Distributed travel-time seismic tomography in large-scale sensor networks," *Journal of Parallel and Distributed Computing*, vol. 89, 2016. [Online]. Available: http://sensorweb.engr.uga.edu/wp-content/uploads/2016/08/DropboxChooserAPI_KSSL-JPDC2015.pdf

[15] G. Kamath, P. Ramanan, and W.-Z. Song, "Distributed Randomized Kaczmarz and Applications to Seismic Imaging in Sensor Network," in *The 11th IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS), Fortaleza, Brazil*, 2015.

[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2003.

[17] M. T. Heath, E. Ng, and B. W. Peyton, "Parallel algorithms for sparse linear systems," *SIAM review*, vol. 33, no. 3, pp. 420–460, 1991.

[18] D. P. Bertsekas and J. N. Tsitsiklis, "Some aspects of parallel and distributed iterative algorithms a survey," *Automatica*, vol. 27, no. 1, pp. 3–21, 1991.

[19] J. M. Elble, N. V. Sahinidis, and P. Vouzis, "GPU computing with Kaczmarz's and other iterative algorithms for linear systems," *Parallel Computing*, vol. 36, pp. 215–231, Jun. 2010.

[20] Y. Censor, D. Gordon, and R. Gordon, "Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems." *Parallel Computing*, vol. 27, no. 6, pp. 777–808, 2001.

[21] D. Gordon and R. Gordon, "Component-averaged row projections: a robust, block-parallel scheme for sparse linear systems," *SIAM Journal on Scientific Computing*, vol. 27, pp. 1092–1117, 2005.

[22] G. Cimmino, "Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari, XVI (9) (1938) 326333," *La Ricerca Scientifica*, vol. 16, no. 9, pp. 326–333, 1938.

[23] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Fast Distributed Gradient Methods," Apr. 2014. [Online]. Available: http://arxiv.org/abs/1112.2972

[24] K. Yuan, Q. Ling, and W. Yin, "On the Convergence of Decentralized Gradient Descent," Feb. 2014. [Online]. Available: http://arxiv.org/abs/1310.7063

[25] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An Exact First-Order Algorithm for Decentralized Consensus Optimization," Nov. 2014. [Online]. Available: http://arxiv.org/abs/1404.6264

[26] E. Wei and A. Ozdaglar, "On the O(1/k) Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers," Jul. 2013. [Online]. Available: http://arxiv.org/abs/1307.8254

[27] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization," *Signal Processing, IEEE Transactions on*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014. [Online]. Available: http://dx.doi.org/10.1109/tsp.2014.2304432

[28] A. Nedic and A. Olshevsky, "Distributed Optimization Over Time-Varying Directed Graphs," *Automatic Control, IEEE Transactions on*, vol. 60, no. 3, pp. 601–615, Mar. 2015. [Online]. Available: http://dx.doi.org/10.1109/tac.2014.2364096

[29] N. Freris and A. Zouzias, "Fast distributed smoothing for network clock synchronization," 2012. [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=j38QfhkAAAAJ&citation_for_view=j38QfhkAAAAJ:W7OEmFMy1HYC

[30] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An Asynchronous Parallel Stochastic Coordinate Descent Algorithm," Nov. 2014. [Online]. Available: http://arxiv.org/abs/1311.1873

[31] G. Liu, R. Tan, R. Zhou, G. Xing, W. Song, and J. Lees, "Volcanic Earthquake Timing using Wireless Sensor Networks," in *The 12th ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*, 2013, pp. 91–102.

[32] L. Geiger, "Probability method for the determination of earthquake epicenters from the arrival time only," *Bull.St.Louis.Univ*, vol. 8, pp. 60–71, 1912.

[33] R. Fischer and J. M. Lees, "Shortest path ray tracing with sparse graphs," *GEOPHYSICS*, vol. 58, no. 7, pp. 987–996, Jul. 1993.

[34] J. M. Lees and R. S. Crosson, "Bayesian Art versus Conjugate Gradient Methods in Tomographic Seismic Imaging: An Application at Mount St. Helens, Washington," *Institute of Mathematical Statistics*, vol. 20, pp. 186–208, 1991.

[35] P. C. Hansen, *Discrete Inverse Problems*. Society for Industrial and Applied Mathematics, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1137/1.9780898718836

[36] G. T. Herman, *Reconstruction from Projections: The Fundamentals of Computerized Tomography*. Academic Press, 1980.

[37] S. Kaczmarz, "Angenäherte Auflösung von Systemen linearer Gleichungen," *Bulletin International de l'Académie Polonaise des Sciences et des Lettres*, vol. 35, pp. 355–357, 1937.

[38] P. P. B. Eggermont, G. T. Herman, and A. Lent, "Iterative algorithms for large partitioned linear systems, with applications to image reconstruction," *Linear Algebra and its Applications*, vol. 40, pp. 37–67, Oct. 1981. [Online]. Available: http://dx.doi.org/10.1016/0024-3795(81)90139-7

[39] M. R. Trummer, "Reconstructing pictures from projections: On the convergence of the ART algorithm with relaxation," vol. 26, no. 3, pp. 189–195, 1981. [Online]. Available: http://dx.doi.org/10.1007/bf02243477

[40] D. Needell, "Randomized Kaczmarz solver for noisy linear systems," *BIT*, vol. 50, no. 2, pp. 395–403, 2010.

[41] Y. Censor, T. Elfving, and G. T. Herman, "On diagonally-relaxed orthogonal projection methods," *SIAM J. Sci. Comput*, vol. 30, no. 1, pp. 473–504, Jun. 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.333.960

[42] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484–512, Mar. 1984.

[43] A. Michelini, “An adaptive-grid formalism for traveltime tomography,” *Geophysical Journal International*, vol. 121, no. 2, pp. 489–510, May 1995. [Online]. Available: http://dx.doi.org/10.1111/j.1365-246x.1995.tb05728.x

[44] A. L. Vesnaver, “Irregular grids in seismic tomography and minimum-time ray tracing,” *Geophysical Journal International*, vol. 126, no. 1, pp. 147–165, Jul. 1996.

[45] A. Curtis and R. Snieder, “Reconditioning inverse problems using the genetic algorithm and revised parameterization,” *Geophysics*, vol. 62, no. 5, pp. 1524–1532, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1190/1.1444255

[46] W. Spakman and H. Bijwaard, “Optimization of Cell Parameterizations for Tomographic Inverse Problems,” *Pure and Applied Geophysics*, vol. 158, no. 8, pp. 1401+, 2001.

[47] M. Bertero, C. D. Mol, and E. R. Pike, “Linear inverse problems with discrete data. I. General formulation and singular system analysis,” vol. 1, no. 4, pp. 301–330, 1985. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/1/4/004

[48] I. Gargantini, “An Effective Way to Represent Quadtrees,” *Commun. ACM*, vol. 25, no. 12, pp. 905–910, Dec. 1982.

[49] J. Ahrenholz, T. Goff, and B. Adamson, “Integration of the CORE and EMANE Network Emulators,” in *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, 2011, pp. 1870–1875.

[50] R. Van Renesse, K. P. Birman, and W. Vogels, “Astrolabe: A Robust and Scalable Technology For Distributed . . .” in *ACM TRANSACTIONS ON COMPUTER SYSTEMS*, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.6072

[51] H. Straková, W. N. Gansterer, and T. Zemen, “Distributed QR factorization based on randomized algorithms,” in *PPAM’11 Proceedings of the 9th international conference on Parallel Processing and Applied Mathematics - Volume Part I*, 2012, pp. 235–244.

[52] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks," 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3915

[53] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast Gossip Algorithms for Consensus," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2748–2761, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1109/tsp.2009.2016247

[54] G. Kamath, P. Agnihotri, M. Valero, K. Sarker, and W.-Z. Song, "Pushing analytics to the edge," in *2016 IEEE Global Communications Conference: Selected Areas in Communications: Internet of Things (Globecom2016 SAC IoT)*, Washington, USA, 2016. [Online]. Available: http://sensorweb.engr.uga.edu/wp-content/uploads/2016/09/DropboxChooserAPI_KAVSS-GLOBOCOM2016.pdf

[55] W. Menke, *Geophysical data analysis discrete inverse theory*. Elsevier/Academic Press, 2012. [Online]. Available: http://www.worldcat.org/isbn/9780123971609