UNIVERSITY OF CALIFORNIA
Santa Barbara

Distributed Smoothing Based on Relative Measurements

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

by

William Joshua Anthony Russell

Committee in Charge:

Professor João P. Hespanha, Chair

Professor Bassam Bamieh

Professor Francesco Bullo

Professor Katie Byl

June 2012

The dissertation of
William Joshua Anthony Russell is approved:

Professor Bassam Bamieh

Professor Francesco Bullo

Professor Katie Byl

Professor João P. Hespanha, Committee Chair

June 2012

Distributed Smoothing Based on Relative Measurements

by

William Joshua Anthony Russell

For Diana.

Acknowledgements

This material is based upon work supported by the Institute for Collaborative Biotechnologies through grant W911NF-09-D-0001 from the U.S. Army Research Office.

Curriculum Vitæ

William Joshua Anthony Russell

Education

June 2011	MA in Economics, University of California, Santa Barbara
June 2008	MS in Electrical and Computer Engineering, University of California, Santa Barbara
June 2006	BS in Electrical Engineering, University of Washington, Seattle

Experience

2007 - 2012	Graduate Stdent Researcher, University of California, Santa Barbara
2008 - 2008	Intern, Army Research Laboratory, Adelphi, MD
2006 - 2007	Teaching Assistant, University of California, Santa Barbara
2006 - 2006	Intern, Lockheed Martin Corporation, Santa Maria, CA
2005 - 2006	Intern, Tethers Unlimited Inc., Seattle, WA
2005 - 2006	Commodore, Washington Yacht Club, Seattle, WA

Publications

W.J. Russell, P. Barooah, D.J. Klein, J.P. Hespanha, "Approximate Distributed Kalman Filtering and Fixed-Interval Smoothing for Cooperative Multi-agent Localization", to appear in a book published by iConcept Press.

W.J. Russell, D.J. Klein, J.P. Hespanha, "Optimal Estimation on the Graph Cycle Space", *IEEE Transactions on Signal Processing,* June 2011.

W.J. Russell, D.J. Klein, J.P. Hespanha, "Optimal Estimation on the Graph Cycle Space", In *Proceedings of the American Controls Conference,* June 2010.

P. Barooah, W.J. Russell, J.P. Hespanha, "Approximate Distributed Kalman Filtering for Cooperative Multi-Agent Localization", In *Proceedings of the International Conference on Distributed Computing in Sensor Systems,* June 2010.

S. Gebre, K. Johnson, W.J. Russell, C. Sun, "Design and Construction of a Reflow Soldering Oven", *University of Washington Electrical Engineering Technical Report,* June 2006.

Abstract

Distributed Smoothing Based on Relative Measurements

by

William Joshua Anthony Russell

This work focuses on the problem of estimating the locations of mobile agents by agent displacement measurements with agent relative position measurements. The problem of distributed Kalman smoothing for this application is reformulated as a parameter estimation problem. The graph structure underlying the reformulated problem makes it computable in a distributed manner using iterative methods of solving linear equations.

The first part of this dissertation presents a smoothing algorithm that computes an approximation of the centralized optimal estimates. The algorithm is distributed in the sense that each agent can estimate its own position by communication only with nearby agents. With finite memory and limited number of iterations before new measurements are obtained, the algorithm produces an approximation of the Kalman smoother estimates. As the memory of each agent and the number of iterations between each time step are increased, the approximation improves. The error covariances of the location estimates produced by

the proposed algorithm are significantly lower than what is possible if inter-agent relative position measurements are not available.

The second part of this dissertation presents an algorithm that reduces communication for a class of distributed estimation problems, which includes the aforementioned smoothing algorithm. Herein, the agents are viewed as nodes in a graph and the relative measurements between agents are viewed as the graph's edges. The algorithm exploits the existence of cycles in the graph to compute the best linear state estimates. For large graphs, the algorithm significantly reduces the total number of message exchanges that are needed to obtain an optimal estimate. The algorithm is guaranteed to converge for planar graphs and provide explicit formulas for its convergence rate for regular lattices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, large scale wireless sensor networks (WSNs) have become more prevalent in military and consumer applications. As the name implies, a WSNs is a network consisting of a group of *sensor nodes* that communicate wirelessly. At the most basic level, sensor nodes are simple self-contained devices consisting of a sensor, a computer or microprocessor, and a wireless communication device. However, a sensor node can be a very complex machine such as an unmanned aerial vehicle (UAV) or autonomous robot with many on-board sensors. WSNs are of considerable interest due to their multitude of uses including localization, target tracking, and environmental monitoring. A WSN can conceivably contain thousands of nodes and can be used to monitor vast geographic regions. Sensor nodes can be made cheap enough that they are considered disposable. In this configuration, they are especially useful in monitoring hazardous environments,

as they can be deployed via aircraft and discarded once their mission is complete or their battery is depleted.

One thing common to all sensor nodes is that they collect local data in the form of stochastic measurements. These measurements are of some environmental state that is critical to the WSN's mission. Stochasticity in the measurements is considered to be exogenous, meaning that it is natural and unavoidable. In raw form, the measurements provide a noisy estimate of the environmental state. Although measurement noise is unavoidable, estimation techniques can be used to reduce the effects of measurement noise. By exploiting redundancies of the information contained in the measurements, estimation techniques predict the underlying network states measurably better (typically lower error variance) then the measurements themselves.

In small networks, it may be possible to compute state estimates at a single *master node* within the network. This is referred to as a *centralized estimation* method. There are several drawbacks with such centralized methods including network congestion, lack of robustness and poor scalability. In order to compute an estimate at one master node in the network, all of the network's measurements must be collected at the master node. This leads to high network traffic in the areas surrounding the master node. With such methods, security is impaired due to the fact that centralized algorithms have a single point of failure. If the master

node becomes disabled, then the entire WSN can be rendered inoperable. Also, with centralized estimation methods, the effort required to compute estimates for the network can quickly grow to an unmanageable size.

To avoid the drawbacks of centralized estimation, one must look toward *distributed estimation* methods. In a distributed algorithm, sensor nodes rely only upon local measurements and communication with nearby sensor nodes to compute estimates of the network's state. In this manner, computation and communication is distributed throughout the network. This reduces communication bottlenecks and eliminates a single point of failure in the network. Furthermore, by using only local information, the complexity of the problem at any given node remains small, even as the size of the network grows very large. Such algorithms typically approximate or converge to some existing central optimal estimation method.

There have been many approaches to the distributed estimation problem for different applications. Among the simplest distributed estimation problems is the linear consensus problem addressed in [15, 20, 45, 51, 52], in which the state to be estimated is observable to all sensors in the network. More challenging problems such as distributed Kalman filtering [2, 10, 11, 33, 43, 46] and distributed Kalman smoothing [6, 7, 12] have been used for estimation of a dynamic states. More challenging still, are the problem of track-to-track fusion

[13, 32], in which highly correlated tracks are fused into an estimate, and the problem of localization from time-difference-of-arrival measurements [18], where one must estimate the intersection of hyperbolic functions.

This work focuses on a particular problem known as estimation from relative measurements (ERM). In this problem, each sensor node has associated with it, a local state. For example, the local state could be a the node's geographic position, or its velocity relative to other sensor nodes, or even the time measured by an internal clock. The sensor nodes cannot measure their state directly, but they can make relative difference measurements of their state with respect to the state of other nearby sensor nodes. Each sensor must then use these relative difference measurements along with information collected by neighboring nodes to estimate its own state.

As a conceptual example of an ERM problem, consider the problem of estimating voltages in an vast electrical network with a sensor node located at each node in the electrical circuit. Each sensor node is tasked with monitoring the voltage (with respect to a network ground) at its own location in the circuit. The nodes are generally far from grounds in the circuit and have no way to measure their voltage directly. However, each sensor node is aware of the impedances for branches of the circuit that it is connected to. Each sensor can measure the current traveling through these incident branches and thus it

has the ability to measure the potential difference between its own circuit node and the circuit nodes of its neighbors. By communicating with only neighboring nodes, each sensor node must estimate its own voltage with respect to a grounding node. In this example, the local state is voltage with respect to ground and the measurements are the voltages in each branch of the circuit.

Another example of an ERM problem is that of sensor localization. Suppose that one wanted to construct a network of sensors for environmental monitoring of a remote jungle area. To collect information accurately, the position of each sensor node within the environment must be known. Because the environment that needs monitored is dangerous, the nodes cannot be placed by hand, so they are dropped out of the back of an airplane. Some sensors have an obstructed view of the sky and therefore they cannot ascertain their positions from GPS. The nodes are equipped with sensors that allow them to make noisy range and bearing measurements to other nearby sensor nodes. These range and bearing measurements form a network of two-dimensional relative position measurements that can be used to localize the sensor nodes in the environment.

In this work several assumptions that are common in the estimation literature are adopted. First, it is assumed that each measurement is corrupted by some additive Gaussian noise error. It is assumed that these errors are independent of each other and independent of the underlying measurement parameters.

Furthermore, it is assumed that the covariances of these errors is known. When all the measurements and error covariances are available at a single node, the best linear unbiased estimate (BLUE) of the relative differences can be calculated. The BLUE is optimal in the sense that it is the estimate with minimum mean squared error of all estimates that have zero expected error [1]. For the ERM problem, the BLUE is only unique up to an additive constant (i.e. one could add a constant to every sensor node's estimate and the result would also be optimal). Because of this, it is always assumed that there exists at least one *reference node* within each independent section of the network and all estimates are relative to the state of the reference node(s) rendering a unique BLUE. The reference node is similar to a ground from the conceptual example above. With regard to electrical networks, it is common to speak of the voltage of a node. Voltage is a measurement of potential energy relative to a reference node that is deemed to be the circuit's ground.

This thesis is concerned with distributed estimation from relative measurements (DERM). The work herein assumes that there is a WSN in which sensor nodes are tasked with estimating some local state from relative difference measurements. All of the methods presented are iterative and require that nodes communicate with a small set of neighbors. The methods presented converge linearly to the BLUE and are thus approximations of the optimal estimate. These

distributed methods are optimal in the sense that, given enough time, one can obtain estimates that are arbitrarily close to the BLUE. One of the groups to throughly investigate distributed methods for the estimation from relative measurements problem were Barooah and Hespanha. Their work throughly discusses the DERM problem for a network of stationary sensors [4]. This thesis builds on this work and takes the DERM problem in two new directions.

1.1 Organization and Contributions

This thesis is organized as follows:

Chapter 2 reviews prior results for the DERM problem. This chapter will formally introduce the DERM problem for a WSN with stationary nodes. It discusses how a system of relative measurements can be represented using graph theory. In doing this, each sensor node is represented by a node in a graph and each pair of nodes that share a relative measurement are connected by an edge in the graph. The system of relative measurements is then conveniently represented by the graph's incidence matrix. The BLUE is then shown to be the weighted least squares solution for a system of equations involving the graph's incidence matrix. Barooah and Hespanha's iterative distributed estimation algorithm is introduced. This algorithm is important because all of the methods

in the remainder of the thesis were inspired by it. In this algorithm, the nodes of the network use local information to compute an estimate of the network's state. Under minimal assumptions, the algorithm is shown to converge to the BLUE.

Chapter 3 focuses on the problem of estimating the locations of mobile agents by fusing inter-agent relative position measurements with measurements of the agents' displacements over time. Due to communication constraints, the location estimates cannot be computed by a central node. An algorithm that approximates the centralized minimum mean squared error estimate is explored. In this algorithm, communication only occurs between agents in a local neighborhood. The problems of distributed Kalman filtering and fixed-interval smoothing are reformulated as parameter estimation problems. The underlying graph structure of the reformulated problem is compatible with existing distributed iterative linear equation solvers. When computation and memory are unbounded, the algorithm renders the minimum mean squared error estimates. With finite memory and a limited number of iterations before new measurements are obtained, the algorithm produces an approximation of the optimal estimates. As the memory of each agent and the number of iterations between each time step are increased, this approximation improves. For a lattice graph, the convergence rate of the algorithm is presented and compared to the error introduced

by agents with finite memory. Simulations are presented that show that even with limited communication and computation, the algorithm provides estimates that are close to the centralized optimal. The algorithm produces estimates that greatly reduce error covariance when compared with dead reckoning.

Chapter 4 proposes a new algorithm for the DERM problem. This algorithm exploits the existence of cycles in the graph to compute the BLUE of the network states. This method is inspired by the fact that, in the absence of noise, the sum of relative difference measurements around a cycle is zero. Such measurements are consistent and, borrowing from the conceptual example above, such measurements are called a *tension set*. When measurements are noisy, the sum of measurements around a cycle is typically not zero; this sum is called discrepancy. The discrepancy of a cycle can be divvied and added to the measurements of the cycle to form a tension set. A method that "corrects" the measurements around the cycles to make them tension sets (i.e., consistent) is provided. This correction is constructed so that the (new) consistent measurements correspond precisely to the BLUE. The BLUE of any node's state can be obtained from the optimal tension set by adding the tensions along a path from a reference node to the node in question. For large graphs, the new algorithm significantly reduces the total number of message exchanges that are needed to obtain an optimal estimate. It is then shown that the new algorithm is guaran-

teed to converge to the BLUE for planar graphs and provide explicit formulas for its convergence rate for regular lattices.

The following notation is used in this chapter to improve readability. Matrices are represented by uppercase variables, vectors are represented by bold lowercase variables, and scalars are represented by italicized lowercase variables. The transpose operator is denoted with T, for example $\mathbf{z}^\mathsf{T}$ is the transpose of $\mathbf{z}$. In all cases, i, j, and k are used as index variables. The size-d identity matrix is denoted by I_d. The "tilde" operator is used to denote a noisy measurement, for example $\tilde{\mathbf{z}}$ is a measurement of $\mathbf{z}$. The "hat" operator is used to indicate an estimate, for example $\hat{\mathbf{z}}$ is an estimate of $\mathbf{z}$. An asterisk superscript is used to denote an optimal estimate: $\hat{\mathbf{z}}^*$ is the optimal estimate of $\mathbf{z}$.

Chapter 2

The Estimation from Relative Measurements Problem

The estimation from relative measurements (ERM) framework has been proposed as a solution to several real world estimation problems. This chapter uses one of these, sensor localization, to motivate ERM and to introduce solution methods. However, the results presented here are applicable to any ERM problem. Many of the ideas in this chapter come from the work of Barooah and Hespanha; a thorough treatment of this subject matter is provided in [4]. This chapter will introduce the ERM problem and it will provide a centralized solution for such problems. This chapter will then introduce Barooah and Hespanha's iterative distributed algorithm for the solution of the ERM problem and show that this algorithm converges with minimal assumptions.

2.1 Problem Formulation

Consider a group of unmanned ground sensors, referred to as *agents*, strewn randomly in some environment. The agents have been tasked with environmental monitoring; to accurately perform their duties, each sensor must localize itself within the environment. That is to say, it must determine its own d-dimensional position. Due to exogenous circumstances very few (but at least one) of the Agents have access to GPS data. However, all of the agents are equipped with devices that allow them to make noisy measurements of the relative positions of nearby agents. It is assumed that measurement noises are Gaussian and uncorrelated with each other as well as with the agents' states. Furthermore, the agents are aware of the statistical properties of their respective sensors and therefore know the variance of the measurement errors. Additionally, each agent has been furnished with radio equipment that lets them communicate with its neighboring agents. Using only local measurements and communicating with neighboring agents, each agent must estimate its own position.

To start, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the directed graph associated with the measurements of the ERM problem. Each agent has associated with it a node in $\mathcal{V}$, the set of nodes in $\mathcal{G}$. When two agents are connected via a relative measurement, there is a corresponding edge in the edge set, $\mathcal{E}$. That is to say, two agents

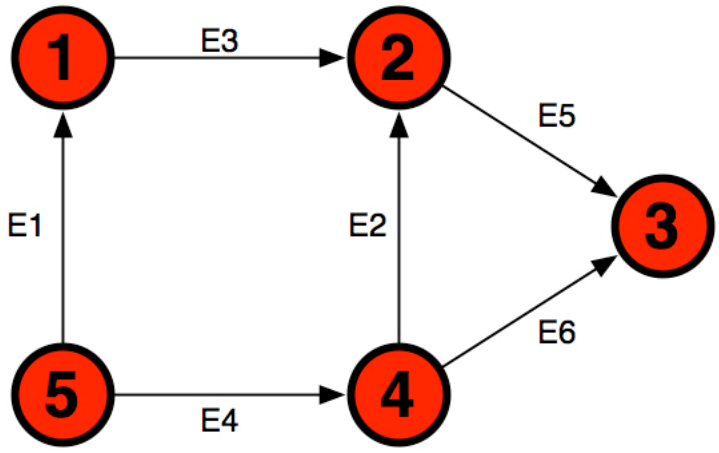
connected in the network by a measurement have corresponding nodes in that are connected by an edge in $\mathcal{G}$. By convention, the direction of an edge goes from the node that appears with the plus sign to the node that appears with the minus sign in the relative difference. Let n and m be the cardinality of $\mathcal{V}$ and the cardinality of $\mathcal{E}$ respectively. The measurement graph can be represented compactly by its incidence matrix $B_{\mathrm{G}} = [b_{i,j}]$, $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$,

$$b_{i,j} \equiv \begin{cases} 1 & \text{if edge } j \text{ leaves node } i \\ -1 & \text{if edge } j \text{ enters node } i \\ 0 & \text{otherwise.} \end{cases}$$

For an example incidence matrix, see Figure 2.1.

Denote by $B_{\mathcal{G}}$ the Kronecker product of B_{G} and an identity matrix of size d, by $\tilde{\mathbf{z}}$ the vector of stacked measurements, by $\mathbf{w}$ the vector of measurement noise, and by $\mathbf{x}_{\mathcal{G}}$ the vector of agent positions (note that $\tilde{\mathbf{z}}$ and $\mathbf{w}$ are of length dm and $\mathbf{x}_{\mathcal{G}}$ is of length dn), then:

$$\tilde{\mathbf{z}} = B_{\mathcal{G}}^{\mathsf{T}} \mathbf{x}_{\mathcal{G}} + \mathbf{w}. \tag{2.1}$$



$$B_{\mathrm{G}} = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 2.1: An example graph shown along with its incidence matrix, B_{G}.

In this notation, $B_{\mathcal{G}}$ is subscripted by $\mathcal{G}$ to indicate that it is an incidence matrix for the whole measurement graph.

Because measurements are taken as relative differences in positions, the nullspace of $B_{\mathcal{G}}$ is not empty. Any $\mathbf{x}_{\mathcal{G}}$ that solves (2.1) is unique only up to an additive constant (i.e. all sensor nodes could be shifted a fixed distance and (2.1) would still hold). To remove this degree of freedom, it is assumed that at least one agent per disconnected subgraph of $\mathcal{G}$ has access to GPS. The set of agents with GPS are known as *reference agents* and their corresponding nodes are called *reference nodes*. Reference nodes are important as they globally anchor

the position estimates computed by at other nodes. Denote by $\mathcal{R} \in \mathcal{V}$ the set of reference nodes of the measurement graph. The position vector, $\mathbf{x}_{\mathcal{G}}$, can be split into two vectors $\mathbf{x}_{\mathcal{R}}$ and $\mathbf{x}$ containing, respectively, the positions of the reference nodes and the positions of the nodes that need to be estimated. Likewise, $B_{\mathcal{G}}$ can be split into two matrices, $B_{\mathcal{R}}$ and B containing, respectively, the rows of $B_{\mathcal{G}}$ corresponding to reference nodes and the rows of $B_{\mathcal{G}}$ corresponding to nodes whose state needs to be estimated. Moving the reference positions (and the corresponding columns of $B_{\mathcal{G}}^{\mathsf{T}}$) to the left-hand side of (2.1), results in

$$\tilde{\mathbf{z}} - B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} = B^{\mathsf{T}}\mathbf{x} + \mathbf{w}, \tag{2.2}$$

an overdetermined stochastic linear equation. Such equations and their solutions have been explored by engineers for several centuries. In the next subsection, the well known method of weighted least squares is used to solve this system of equations.

2.2 The Best Linear Unbiased Estimate

When presented with an overdetermined system of equations, such as (2.2), where one wishes to estimate the states, $\mathbf{x}$, one often considers the best linear unbiased estimate (BLUE). A linear estimator is one where the estimate is a

linear combination of dependent variables, $(\tilde{\mathbf{z}} - B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}})$. An estimator is unbiased if its expected error is zero. Among all linear unbiased estimators, the BLUE has the minimum mean squared error. When B is full row-rank, the BLUE is unique and coincides with the maximum likely estimator for the case of Gaussian noise. The goal of the estimators presented here and in the remainder of this thesis is to compute the BLUE, $\hat{\mathbf{x}}^*$, of the unknown positions, $\mathbf{x}$.

Given the system of equations in (2.2), one can compute the BLUE using the method of weighted least squares. Namely, the BLUE is the solution to the linear equation

$$BP^{-1}B^{\mathsf{T}}\hat{\mathbf{x}} = BP^{-1}(\tilde{\mathbf{z}} - B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}}), \tag{2.3}$$

where P denotes the dm x dm matrix of measurement noise covariance,

$$P \equiv \begin{bmatrix} P_1 & 0 & \dots & 0 \\ 0 & P_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P_m \end{bmatrix}, \quad P_j \equiv E\left[\mathbf{w}_j \mathbf{w}_j^{\mathsf{T}}\right], \quad j \in \{1, \dots, m\}. \tag{2.4}$$

For this equation to have a unique solution, $BP^{-1}B^{\mathsf{T}}$ must be invertible. Because P is a positive definite covariance matrix, it is full rank. Therefore, $\text{rank}(BP^{-1}B^{\mathsf{T}}) = \text{rank}(BB^{\mathsf{T}})$, which is invertible if and only if rows of B are

linearly independent (i.e., B is full row-rank). When each disconnected subgraph of $\mathcal{G}$ has at least one ground node, the row independence requirement on B is met. Recall that B was originally derived from B_{G}, the incidence matrix of the communication graph. By definition, B_{G} is rank deficient. The cardinality of this deficiency (i.e. the nullility of B_{G}) is equal to the number of independent subgraphs of $\mathcal{G}$ [50]. Removing at least one row of B_{G} for each independent subgraph of $\mathcal{G}$ results in a full row-rank matrix that called the *grounded incidence matrix*. Since the Kronecker product of two full rank matrices is full rank [8] and B is the Kronecker product of a grounded incidence matrix and an identity matrix one can conclude that if there is at least one reference node for each independent subgraph of $\mathcal{G}$, B is full row-rank. When B is full row-rank, the weighted least squares equation in (2.3) has a unique solution, which is the Best Linear Unbiased Estimator (BLUE) for the unknown positions $\mathbf{x}$ given the measurements $\tilde{\mathbf{z}}$ and reference states $\mathbf{x}_{\mathcal{R}}$:

$$\hat{\mathbf{x}}^* \equiv (BP^{-1}B^{\mathsf{T}})^{-1}BP^{-1}(\tilde{\mathbf{z}} - B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}}). \tag{2.5}$$

2.3 The Distributed Jacobi Iterative Algorithm

With the BLUE defined, a distributed iterative algorithm that converges to the BLUE can be introduced. When it was originally introduced in [4], this method was called the Static Communication Graph (SCG) algorithm. Herein it is referred to as the *Jacobi Algorithm*, as it was renamed in subsequent publications. Because the Jacobi Algorithm serves as important background material for the remainder of this thesis, it is explored here in detail. As with the other methods discussed in this work, the Jacobi Algorithm solves a multitude of ERM problems. For simplicity, the aforementioned agent localization problem is used to motivate the algorithm.

The Jacobi Algorithm is an iterative solution to the ERM problem. The algorithm is distributed in the sense that each agent can approximate the BLUE of its own position using only local measurements and communicating only with its neighbors in the measurements graph. As such, the amount of computation needed at each agent is limited by the number of relative measurements incident on the node. This is particularly useful as the method is still tractable in very large graphs. In this algorithm, the estimates produced at each iteration are unbiased and as the number of iterations increases, the estimates converge to the BLUE.

Another key attribute of the Jacobi Algorithm (one which is inherited by algorithms discussed later) is robustness to asynchronous communications and temporary sensor failure [4]. Such failures are common due to packet dropout, radio congestion, and hostile jamming. It was shown that as long as a agent is not permanently disabled, the Jacobi Algorithm provides estimates that converge to the BLUE. This robustness is especially important for the type of asynchronous communications that are common in WSNs.

In the Jacobi Algorithm, each agent acts autonomously and is aware of a small portion of the network called a *local subgraph*. Denote the *neighbor set* of Agent i (i.e., nodes connected to the i^{th} node, v_i, via an edge) by $\mathcal{N}_i$. Agent i's

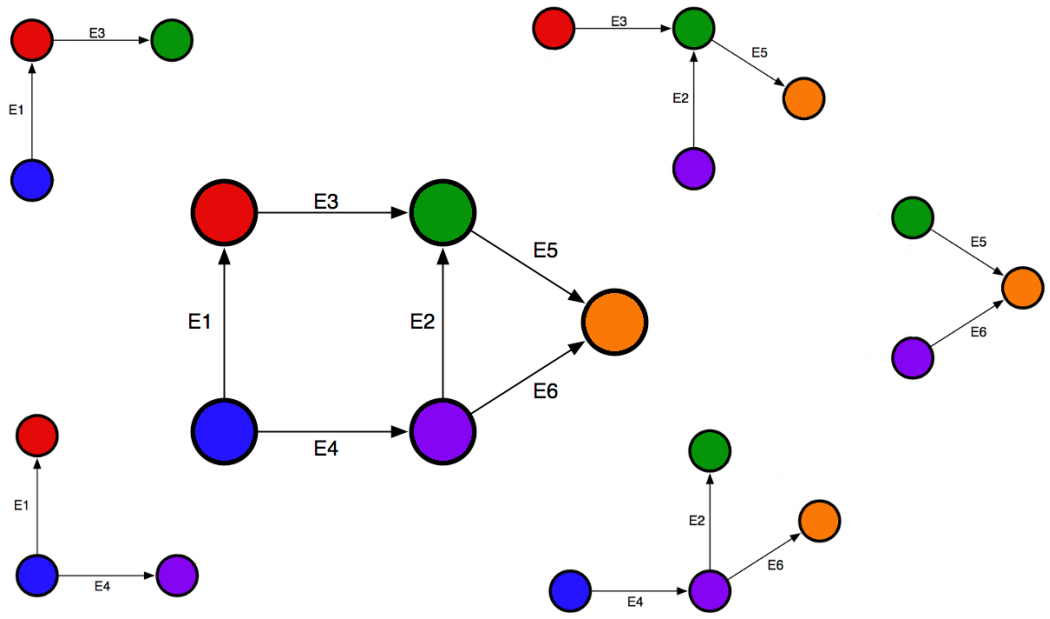


Figure 2.2: The local subgraphs for the the graph from Figure 2.1.

local subgraph, $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, is a subgraph of $\mathcal{G}$ containing v_i, all edges incident on v_i, and all nodes in $\mathcal{N}_i$. Note that $\mathcal{G}_i$ is a star-graph centered at node i. Figure 2.2 provides examples of $\mathcal{G}_i$ for the simple network depicted in 2.1. From this figure, it is easy to see that for $i \in \{1, 2, \ldots n\}$, $\cup_{i=1}^{n} \mathcal{G}_i = \mathcal{G}$.

The Jacobi Algorithm can be described as follows. Prior to the first iteration, all agents that have obtained relative measurements share these measurements and their associated covariances with the neighbor that corresponds to the relative measurement. Agents with unknown positions initialize their own estimates to some arbitrary initial estimates. The Jacobi Algorithm commences by iterating the following steps:

1. Agents transmit their current position estimates to neighboring agents.

2. Subsequently, agents receive position estimates from their neighbors.

3. Agents update their estimates by assuming that their neighbors are reference nodes and computing the BLUE for their local subgraph.

The beauty in the Jacobi Algorithm lies in its simplicity. Each agent is aware of only a small portion of the sensor network and carries out only a small optimization problem. However, it will be shown that the algorithm converges to the BLUE. To analyze this algorithm, a few mathematical elements must be introduced. The generalized incidence matrix $B_{\mathcal{G}}$ plays a key role in the

analysis of the Jacobi Algorithm. $B_{\mathcal{G}}$ is a block matrix consisting of an n by m array of square d-sized blocks. $B_{i,j}$ is the submatrix in the i^{th} block-row and j^{th} block column of $B_{\mathcal{G}}$. $B_{i,j}$ is non-zero only if edge e_j is incident on node v_i. Denote the i^{th} block-row and j^{th} block-column by $B_{i,:}$ and $B_{:,j}$, respectively. There are exactly two non-zero blocks per block-column, $B_{:,j}$. Also, nodes i and i' are neighbors if and only if the inner product of their respective block-rows is non-trivial (i.e. $B_{i,:}B_{i',:}^{\top} \neq 0$). As defined in (2.4), P is a block diagonal covariance matrix where j^{th} diagonal block, $P_{j,j}$, is the covariance of $\tilde{\mathbf{z}}_j$, the j^{th} measurement.

Recall that the set of Agent i's neighboring nodes and the set of reference nodes are denoted $\mathcal{N}_i$ and $\mathcal{R}$, respectively. Denote by $\mathcal{N}_i \setminus \mathcal{R}$, the set of nodes neighboring Agent i that are not in the set of reference nodes. The measurements of Agent i's subgraph can be written

$$\tilde{\mathbf{z}}_j = B_{i',j}^{\top}\hat{\mathbf{x}}_{i'} + B_{r,j}^{\top}\mathbf{x}_r + B_{i,j}^{\top}\hat{\mathbf{x}}_i + \mathbf{w}_j, \tag{2.6}$$

for i' such that $\mathrm{v}_{i'} \in \mathcal{N}_i \setminus \mathcal{R}$, for r such that $\mathrm{v}_r \in \mathcal{N}_i \cap \mathcal{R}$, and for j such that $\mathrm{e}_j \in \mathcal{E}_i$.

In the Jacobi Algorithm, Agent i treats neighboring nodes' position estimates as reference and updates its own position estimate to be the BLUE of $\hat{\mathbf{x}}_i$

in (2.6)

$$\hat{\mathbf{x}}_i^+ = (B_{i,j}P_{j,j}^{-1}B_{i,j}^\top)^{-1}B_{i,j}P_{j,j}^{-1}(\tilde{\mathbf{z}}_j - B_{i',j}^\top\hat{\mathbf{x}}_{i'} - B_{r,j}^\top\mathbf{x}_r). \tag{2.7}$$

Due to the sparsity in $B_{\mathcal{G}}$, this equation can be rewritten to

$$\hat{\mathbf{x}}_i^+ = (B_{i,:}P^{-1}B_{i,:}^\top)^{-1}B_{i,:}P^{-1}(\tilde{\mathbf{z}} - B_{-i,:}^\top\hat{\mathbf{x}}_{-i} - B_r^\top\mathbf{x}_r), \tag{2.8}$$

for $-i$ such that $\mathrm{v}_{-i} \in \mathcal{V} \setminus (\mathcal{R} \cup \mathrm{v}_i)$. That is to say, $-i$ denotes the indices for all non-reference nodes except node i. Indexing the nodes with unknown states from 1 to $\bar{n}$ and consolidating the unknown estimates into a single vector, $\hat{\mathbf{x}}$, (2.8) is rewritten

$$\hat{\mathbf{x}}^+ = D^{-1}A\hat{\mathbf{x}} + D^{-1}BP^{-1}(\tilde{\mathbf{z}} - B_{\mathcal{R}}^\top\mathbf{x}_{\mathcal{R}}), \tag{2.9}$$

where

$$D \equiv \begin{bmatrix} B_{1,:}P^{-1}B_{1,:}^\top & 0 & 0 & \dots & 0 \\ 0 & B_{2,:}P^{-1}B_{2,:}^\top & 0 & \dots & 0 \\ 0 & 0 & B_{3,:}P^{-1}B_{3,:}^\top & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & B_{\bar{n},:}P^{-1}B_{\bar{n},:}^\top \end{bmatrix} \tag{2.10}$$

and

$$A \equiv -\begin{bmatrix} 0 & B_{1,:}P^{-1}B_{2,:}^\top & B_{1,:}P^{-1}B_{3,:}^\top & \dots & B_{1,:}P^{-1}B_{\bar{n},:}^\top \\ B_{2,:}P^{-1}B_{1,:}^\top & 0 & B_{2,:}P^{-1}B_{3,:}^\top & \dots & B_{2,:}P^{-1}B_{\bar{n},:}^\top \\ B_{3,:}P^{-1}B_{1,:}^\top & B_{3,:}P^{-1}B_{2,:}^\top & 0 & \dots & B_{3,:}P^{-1}B_{\bar{n},:}^\top \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{\bar{n},:}P^{-1}B_{1,:}^\top & B_{\bar{n},:}P^{-1}B_{2,:}^\top & B_{n,:}P^{-1}B_{3,:}^\top & \dots & 0 \end{bmatrix}. \tag{2.11}$$

Note that in (2.9) the matrix B has not been changed. B is still a submatrix of $B_{\mathcal{G}}$ containing only rows corresponding to nodes with unknown state,

$$B^\top \equiv \begin{bmatrix} B_{1,:}^\top & B_{2,:}^\top & B_{3,:}^\top & \dots & B_{\bar{n},:}^\top \end{bmatrix}. \tag{2.12}$$

From (2.9), the Jacobi Algorithm can represented by a discrete-time linear system. Convergence of the Jacobi Algorithm is dependent only on $\rho(D^{-1}A)$, the spectral radius of $D^{-1}A$. The Jacobi Algorithm converges if and only if $\rho(D^{-1}A) < 1$. Careful observation reveals that $D - A = BP^{-1}B^\top$. Note that P and D are positive definite covariance matrices and by assumption B is full row-rank. As such, $BP^{-1}B^\top$ is positive definite; equivalently $\rho(D - A) > 0$. D is also positive definite and $\rho(D) > 0$. Given these inequalities, the following

hold

$$\rho(D^{-1}(D-A)) > 0 \tag{2.13}$$

$$\rho(\mathrm{I} - D^{-1}A) > 0 \tag{2.14}$$

$$1 > \rho(D^{-1}A). \tag{2.15}$$

When the Jacobi Algorithm converges, the following equations hold

$$\hat{\mathbf{x}} = D^{-1}A\hat{\mathbf{x}} + D^{-1}BP^{-1}(\tilde{\mathbf{z}} - B_r^\top \mathbf{x}_r) \tag{2.16}$$

$$D\hat{\mathbf{x}} = A\hat{\mathbf{x}} + BP^{-1}(\tilde{\mathbf{z}} - B_r^\top \mathbf{x}_r) \tag{2.17}$$

$$(D-A)\hat{\mathbf{x}} = BP^{-1}(\tilde{\mathbf{z}} - B_r^\top \mathbf{x}_r) \tag{2.18}$$

$$\hat{\mathbf{x}} = (D-A)^{-1}BP^{-1}(\tilde{\mathbf{z}} - B_r^\top \mathbf{x}_r) = \hat{\mathbf{x}}^*. \tag{2.19}$$

Combining these results, if B is full row-rank, then the Jacobi Algorithm converges to the BLUE.

2.4 Conclusion

This chapter used position estimation of a static wireless network to introduce the problem of estimation from relative measurements. It showed how

the ERM problem could be modeled using standard graph theoretic concepts. After modeling the graph as a problem, the best linear unbiased estimate was derived. The BLUE is the position estimate with minimum mean squared error. Next, came a review of the Jacobi algorithm, first introduced by Barooah and Hespanha. Finally, this chapter showed that if the sensor network has properly placed reference nodes, then the Jacobi algorithm converges to the BLUE.

Chapter 3

Distributed Approximate Kalman Smoothing

Mobile autonomous agents such as unmanned ground robots and unmanned aerial vehicles that are equipped with on-board sensing, actuation, computation and communication capabilities hold great promise for applications such as surveillance, disaster relief, and scientific exploration. Irrespective of the application, their successful use generally requires that the agents be able to obtain accurate estimates of their positions. Typically, position information is provided by GPS, but in many scenarios GPS may be available only intermittently, or sometimes not available at all. These situations include underwater operation, presence of urban canyons, or hostile jamming. In such situations, localization is typically performed by accumulating over time displacement measurements, which can be obtained from IMUs (inertial measurement units) or/and vision

sensors [9, 31, 34]. Since these measurements are noisy, their integration over time leads to high rates of error growth [25, 34, 36].

When multiple agents operate cooperatively, it is possible to reduce localization errors if measurements regarding *relative positions* between pairs of agents are available. Such measurements can be obtained by vision-based sensors such as cameras and LIDARs, or RF sensors through AoA and TDoA measurements. These measurements, although noisy, furnish information about the agent's locations in addition to the information provided by displacement measurements. The problem of fusing measurements regarding relative positions between mobile agents for estimating their locations is commonly known as *cooperative localization* in the robotics literature [23, 29, 40].

The typical approach for cooperative localization is to use an extended Kalman filter, to fuse both odometry data and robot-to-robot relative distance measurements [29, 42]. However, computing the Kalman filter estimates requires that all the measurements (inter-agent as well as displacement measurements) are available to a central processor. Such a centralized approach requires routing data though an ad-hoc network of mobile agents, which is a difficult problem [30]. Therefore, a *distributed scheme* that allows agents to estimate their positions accurately through local computation and communication instead of relying on a central processor is preferable. In the sensor networks and control lit-

erature, the problem of distributed Kalman filtering has drawn significant attention [2, 10, 33, 46]. The papers [2, 3], in particular, present distributed Kalman filtering techniques for multi-agent localization. The paper [53] addresses cooperative localization in mobile sensor networks with intermittent communication, in which an agent updates its prediction based on the agents it encounters, but does not use inter-agent relative position measurements.

Within the realm of filtering there exists a class of non-causal filters called *smoothers*; smoothers based on Kalman filters are commonly referred to as *fixed-interval smoothers* or *Kalman smoothers*. Fixed-interval smoothers can be used to post-precess data and provide parameter estimates that have less error than estimates found using a Kalman filter. The added accuracy in smoothed estimates comes at the cost of additional computation and delay in receiving said estimates. Smoothers are useful in problems such as surveillance and mapping, where an objective is not time sensitive.

In a discrete time system where state estimates are to be computed from a stream of measurements, fixed interval smoothing is used to estimate the network state over a fixed window of time. The Kalman filter is a particular fixed-interval smoother that is focused on computing optimal estimates over a window of time that contains a single time instant coinciding with the time of the last measurement received. In this chapter the terms Kalman smoothing and fixed-

interval smoothing are used interchangeably. Furthermore, the results presented for these smoothers hold for the specific case of the Kalman filter.

This chapter approaches the problem of distributed Kalman smoothing for cooperative localization, by reformulating it as a parameter estimation problem. The Kalman smoother computes the linear minimum mean-squared error (LMMSE) estimates of the states of a linear system given the measurements. It is a classical result that under infinite prior covariance, the LMMSE is the BLUE (best linear unbiased estimator) [27]. For the problem at hand, the BLUE estimator has a convenient structure that can be described in terms of a graph consisting of nodes (agent locations) and edges (relative measurements). This structure can be exploited to distribute the computations by using parallel iterative methods of solving linear equations. The proposed method therefore is designed to compute the BLUE estimates using iterative techniques, and is based on earlier work on localization in static sensor networks [4]. If the agents had infinite memory and could run infinitely many iterations before their positions change, then the estimates produced by the proposed algorithm are equal to the centralized BLUE estimates, and therefore the same as the Kalman smoother estimates. Due to memory and time constraints, the proposed algorithm uses only a subset of all the past measurements and runs only a few (as little as one) iterations. This makes the method an approximation of the Kalman smoother.

Fewer the number of iterations and smaller the size of past measurements retained, the easier it is to implement the algorithm in a distributed setting. Simulations show the approximations are remarkably close to the centralized Kalman smoother/BLUE estimates even when a very small amount of past data is used and only one iteration is executed between successive motion updates.

The rest of the chapter is organized as follows. Section 3.1 describes the estimation problem precisely. Section 3.2 describes centralized Kalman smoothing for cooperative localization and its reformulation as a problem of parameter estimation in measurement graphs. Section 3.3 describes the proposed algorithm and an extensions to target tracking is given in Section 3.4. Convergence results are presented in Section 3.5. Section 3.6 discusses simulation results and Section 3.7 provides a brief conclusion.

3.1 Problem description

Consider a group of η mobile agents that need to estimate their own positions with respect to a geostationary coordinate frame. Denote time by a discrete index $k = 0, 1, 2, \ldots$ The noisy measurement of the displacement of Agent i obtained by its on-board sensors during the k-th time interval is denoted by $u_i(k)$,

so that

$$u_i(k) = x_i(k+1) - x_i(k) - w_i(k), \tag{3.1}$$

where $x_i(k)$ is the position of Agent i at time k, and $\{w_i(k)\}$ is a zero-mean noise with the property that $E[w_i(k)w_{i'}(k')] = 0$ unless $i = i', k = k'$. It is assumed that certain pairs of agents, say i, i', can also obtain noisy inter-agent measurements

$$y_{ii'}(k) = x_i(k) - x_{i'}(k) + v_{ii'}(k), \tag{3.2}$$

where $v_{ii'}(k)$ is zero-mean measurement noise with $E[v_{ii'}(k)v_{jj'}(k')] = 0$ unless $(ii') = (jj'), k = k'$. It is assumed that the agents are equipped with compasses, so that all these measurements are expressed in a common Cartesian reference frame. Also, occasionally some of the nodes have noisy absolute position measurements (e.g., obtained from GPS) of the form

$$y_i(k) = x_i(k) + v_{ii'}(k), \tag{3.3}$$

The goal is to combine the agent-to-agent relative position measurements with agent displacement measurements to obtain estimates of their locations

that are more accurate than what is possible from the agent displacement measurements alone (i.e. *dead reckoning*). In addition, the computation should be distributed so that every agent can compute its own position estimate by communication with a small subset of the other agents, called *neighbors*. It is assumed that if two agents can obtain each others' relative position measurement at time index k, then they can also exchange information through wireless communication during the interval between instants k and $k+1$.

3.2 Kalman Smoothing vs. BLUE

The displacement measurements (3.1) allow us to write the following process model for the i-th vehicle:

$$x_i(k+1) = x_i(k) + u_i(k) + w_i(k),$$

where $u_i(k)$ is now viewed as a known input. One can stack the states $x_i(k), i = 1, \ldots, \eta$ into a tall vector $\mathbf{x}(k)$ and write the system dynamics

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{u}(k) + \mathbf{w}(k), \quad \mathbf{y}(k) = C(k)\mathbf{x}(k) + \mathbf{v}(k)$$

where $C(k)$ is appropriately defined so that the entries of $\mathbf{y}(k)$ are the inter-agent absolute and relative position measurements (3.2)-(3.3). When the control input and the measurements $\{\mathbf{u}(k)\}, \{\mathbf{y}(k)\}, k \in \{0, 1, \ldots, \bar{k}\}$ are made available to a central processor, along with the initial conditions $\hat{\mathbf{x}}(0|-1)$ and $\Sigma(0|-1) = \mathrm{Cov}(\hat{\mathbf{x}}(0|-1) - \mathbf{x}(0), \hat{\mathbf{x}}(0|-1) - \mathbf{x}(0))$, and the noise covariances $Q(k) := \mathrm{Cov}(\mathbf{w}(k), \mathbf{w}(k))$, $R(k) := \mathrm{Cov}(\mathbf{v}(k), \mathbf{v}(k))$, a Kalman smoother can be used to compute estimate $\hat{\mathbf{x}}^{\mathrm{Kal}}(k|\bar{k}) = E^*(\hat{x}(k)|\{u\}, \{y\})$ of the state $\mathbf{x}(k)$, where $E^*(X|Y)$ denotes the LMMSE estimate of a r.v. X conditioned to the r.v Y [39].

To distribute the computations of the Kalman smoother, the problem is reformulated into an equivalent, deterministic parameter estimation problem. The estimation problem is associated with a measurement graph $\mathcal{G}(\bar{k}) = (\mathcal{V}(\bar{k}), \mathcal{E}(\bar{k}))$ constructed as follows

1. The positions $x_i(k)$ of each agent $i \in \{1, \ldots, \eta\}$ at each time step $k \in \{0, 1, \ldots, \bar{k}\}$ is associated with one node in $\mathcal{V}(\bar{k})$ and an additional "reference" node x_0 that is associated with the origin of an inertial coordinate system is included.

2. Each measurement is associated with a graph edge: agent displacement measurements (3.1) are associated with edges between nodes corresponding to the same agent at different time steps; inter-agent measurements (3.2)

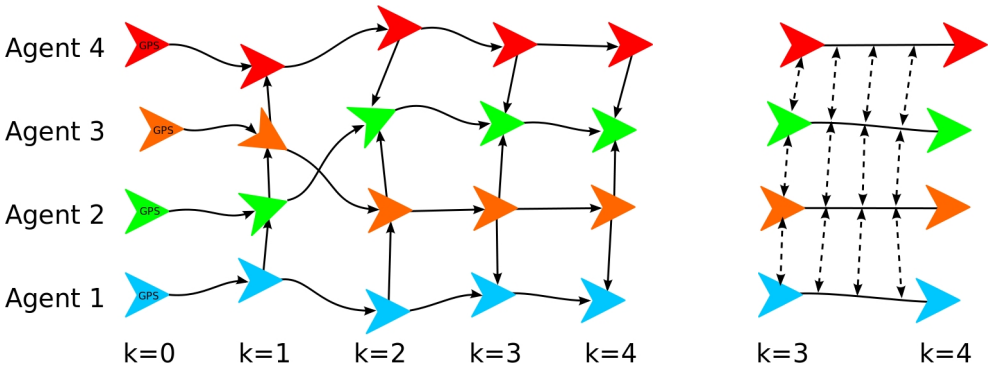
are associated with edges between nodes corresponding to different agents at the same time step; and absolute displacement measurements (3.3) are associated with edges between an agent at a given time instant and the reference node.

All measurements mentioned above are of the type

$$\tilde{z}_{\mathrm{e}} = x_i - x_{i'} + \varepsilon_{\mathrm{e}} \tag{3.4}$$

where i and i' are node indices for the measurement graph $\mathcal{G}(\bar{k})$ and $\mathrm{e} = (i, i')$ is an edge (an ordered pair of nodes) between nodes. In particular, an edge exists between a node i and node i' if and only if a relative measurement of the form (3.4) is available between the two nodes. Since a measurement of $x_i - x_{i'}$ is different from that of $x_{i'} - x_i$, the edges in the measurement graph are directed, but the edge directions are arbitrary and simply have to be consistent with (3.4). Figure 3.1 shows an example of a measurement graph.

Presented here is a brief review the BLUE (best linear unbiased estimator) for relative measurements for graphs that do not change with time. The BLUE is optimal (i.e. minimal-error variance) among all linear unbiased estimators. Consider a measurement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes in $\mathcal{V}$ correspond to variables and edges in $\mathcal{E}$ correspond to relative measurement of the form (3.4).



(a) Measurement graph $\mathcal{G}(4)$.

(b) Communication between $k = 3$ and $k = 4$.

Figure 3.1: (a) An example of a measurement graph generated as a result of the motion of a group of four mobile agents. The graph shown here is $\mathcal{G}(4)$, i.e., the snapshot at the 4^{th} time instant. The unknown variables at current time $k = 4$ are the positions $x_i(k)$, $i \in \{1, \ldots, 4\}$, at the time instants $k \in \{1, \ldots, 4\}$. (b) The communication during the time interval between $k = 3$ and $k = 4$. In the situations shown in the figure, 4 rounds of communication occur between a pair of agents in this time interval.

Let $\mathcal{V}_{\mathcal{R}}$ denote the reference node and $n = |\mathcal{V} \setminus \mathcal{V}_{\mathcal{R}}|$ be the number of unknown variables that are to be estimated.

Denote by $\mathbf{x}$, the vector obtained by stacking together the unknown variables, by $\mathbf{x}_{\mathcal{R}}$, the vector obtained by stacking together the reference node's state, by $\tilde{\mathbf{z}}$, the vector obtained by stacking together measurements $u_i(k)$ and $y_{ii'}(k)$, by $\boldsymbol{\epsilon}$, the vector obtained by stacking together noises $w_i(k)$ and $v_{ii'}(k)$, and by P, the matrix of error covariance. The system of relative measurements can then be written as

$$\tilde{\mathbf{z}} = B^{\top}\mathbf{x} + B_{\mathcal{R}}^{\top}\mathbf{x}_{\mathcal{R}} + \boldsymbol{\epsilon}, \tag{3.5}$$

where $[B^\top B_{\mathcal{R}}^\top]^\top$ is a generalized incidence matrix for $\mathcal{G}$. As described in Chapter 2, given a measurement graph with $\bar{n}$ unknown variables, the BLUE, $\hat{\mathbf{x}}^*$ is given by the solution of a system of linear equations

$$\mathcal{L}\mathbf{x} = \mathbf{b}, \tag{3.6}$$

where $\mathcal{L} = BP^{-1}B^\top$ and $\mathbf{b} = BP^{-1}(\tilde{\mathbf{z}} - B_{\mathcal{R}}^\top \mathbf{x}_{\mathcal{R}})$. The matrix $\mathcal{L}$ is invertible (so that the BLUE, $\hat{\mathbf{x}}^*$, exists and is unique) if and only if for every node, there is an undirected path between the node and the reference node [4] (note that this condition is satisfied by assumption). Under this condition, the covariance matrix of the estimation error $\Sigma := \mathrm{Cov}(\hat{\mathbf{x}}^*, \hat{\mathbf{x}}^*)$ is given by

$$\Sigma = \mathcal{L}^{-1}.$$

If all of the measurements corresponding to the edges in $\mathcal{G}(\bar{k})$ are available to a central processor at time $\bar{k}$, the processor can compute the BLUE of all the node variables in the graph $\mathcal{G}(\bar{k})$ (which correspond to the present as well as past positions of the agents) by solving (3.6). The resulting estimate is denoted by $\hat{\mathbf{x}}^{\mathrm{BLUE}}(\bar{k})$.

The following result, which follows from standard results in estimation theory shows that under uninformative prior, the blue estimate is equivalent to the Kalman smoother estimates.

Lemma 1 *If* $\Sigma^{-1}(0|-1) = 0$, *then* $\left[\hat{x}^{\mathrm{Kal}}(0|\bar{k}) \;\; \hat{x}^{\mathrm{Kal}}(1|\bar{k}) \;\; \ldots \;\; \hat{x}^{\mathrm{Kal}}(\bar{k}|\bar{k})\right]^{\top} = \hat{\mathbf{x}}^{\mathrm{BLUE}}(\bar{k})$. □

The next section utilizes the BLUE formulation to devise distributed algorithms that compute these optimal position estimates.

3.3 Distributed Dynamic Localization

This section presents an iterative distributed algorithm to obtain estimates of the positions mobile agents that are close to the centralized BLUE described in the previous section. Here, distributed is used to mean an iterative algorithm by which every agent estimates its own position based on local sensing and communication with its neighboring agents.

3.3.1 Infinite Memory and Bandwidth

The algorithm is first described assuming that every agent can store and broadcast an unbounded amount of data. This assumption is later relaxed.

For every agent, i, let $\mathcal{V}_i(\bar{k})$ contain all the nodes that correspond to the positions of Agent i and the positions of the agents with whom i has had relative measurements, up to and including time $\bar{k}$, those agents are referred to as the *neighbors of Agent i*. Let $\mathcal{E}_i(\bar{k})$ be the subset of edges in $\mathcal{G}(\bar{k})$ that are incident on nodes that correspond to i's current or past positions. By assumption, the relative measurements and their error covariances $\tilde{z}_{\mathrm{e}}, P_{\mathrm{e}}, \mathrm{e} \in \mathcal{E}_i(\bar{k})$ are available to Agent i at time $\bar{k}$. The *local subgraph* of Agent i at time $\bar{k}$ is defined as $\mathcal{G}_i(\bar{k}) = (\mathcal{V}_i(\bar{k}), \mathcal{E}_i(\bar{k}))$. Figure 3.2 shows the subgraph of Agent 1 at time $\bar{k} = 3$ corresponding to the measurement graph shown in Figure 3.1.

The nodes in a local subgraph $\mathcal{G}_i(\bar{k})$ are divided into two categories - the *internal nodes* $\mathcal{V}_i(\bar{k})^{\mathrm{int}}$ and the *boundary nodes* $\mathcal{V}_i(\bar{k})^{\mathrm{bdy}}$. The internal nodes are the nodes that correspond to the positions of the agent up to and including time $\bar{k}$. The boundary nodes consist of the nodes in the local subgraph that correspond to the positions of the neighboring agents. Thus, $\mathcal{V}_i(\bar{k}) = \mathcal{V}_i(\bar{k})^{\mathrm{int}} \cup \mathcal{V}_i(\bar{k})^{\mathrm{bdy}}$ and $\mathcal{V}_i(\bar{k})^{\mathrm{int}} \cap \mathcal{V}_i(\bar{k})^{\mathrm{bdy}} = \emptyset$.

To explain the algorithm, imagine first that the agents have stopped moving at time $\bar{k}$. A distributed algorithm for the localization of static agents that is based on the Jacobi iterative method of solving linear equations [17] was reviewed in Chapter 2. Now, a generalization of that algorithm for the estimation problem

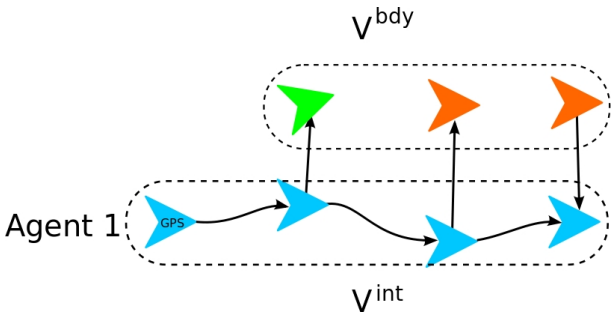


Figure 3.2: The subgraph $\mathcal{G}_1(3)$ of Agent 1 at time 3, for the measurement graph shown in Figure 3.1. The labels v^{int} and v^{bdy} refer to the internal and boundary nodes.

at hand is described. This algorithm will compute BLUE estimates of the entire position history $x_i(k)$, $k \in \{0, 1, \ldots, \bar{k}\}$ of each agent $i \in \{1, \ldots, \eta\}$.

Fixed-time Mobile Agent Iterative Algorithm

Consider the set of past positions of Agent i until time $\bar{k}$, i.e., $\{x_v, v \in \mathcal{V}_i(\bar{k})^{\text{int}} \setminus \mathcal{V}_r\}$. The vector of the node variables in this set is denoted by $\mathbf{x}_i(\bar{k})$. The set of unknown node positions at time $\bar{k}$ is $\cup_i \mathbf{x}_i(\bar{k})$. Let $\hat{\mathbf{x}}_i^\tau(\bar{k})$ be the estimate of $\mathbf{x}_i(\bar{k})$ obtained by Agent i at the end of the τ^{th} iteration. The estimates are obtained and improved using the following distributed algorithm, starting with a dead reckoned initial condition:

At the τ^{th} iteration, every agent, i does the following.

1. It broadcasts its current estimate $\hat{\mathbf{x}}_i^\tau(\bar{k})$ to all neighboring agents. Consequently, it also receives the current estimates $\hat{\mathbf{x}}_{i'}^\tau(\bar{k})$ from each of its neighboring agents, i'.

2. It (Agent i) regards the boundary nodes $\mathcal{V}_i(\bar{k})^{\text{bdy}}$ as reference nodes in its local subgraph $\mathcal{G}_i(\bar{k})$ sets the values of the corresponding node variables to be equal to the estimates that it has received from the corresponding neighbor. With this assignment of reference node variables and with the relative measurements $\{\tilde{z}_\text{e}, \text{e} \in \mathcal{E}_i(\bar{k})\}$, Agent i then sets up the system of linear equations (3.6) for its local subgraph $\mathcal{G}_i(\bar{k})$, and solves these equations to obtain an updated estimate of $\hat{\mathbf{x}}_i^{\tau+1}(\bar{k})$ of its "internal" node variables, $\{x_v, v \in \mathcal{V}_i(\bar{k})^{\text{int}} \setminus \mathcal{V}_r\}$. □

The last step can be restated as, each agent treats neighbors' estimates as references and computes the BLUE for its subgraph, thus computing estimates only for its own nodes.

The following result about the behavior of the estimates follows from the convergence property of the Jacobi algorithm (see [4]).

Proposition 1 *The estimates of all the node variables* $\mathbf{x}(\bar{k})$ *(i.e., all agents' past positions up to time* $\bar{k}$*) converge to their centralized optimal estimates:* $\hat{\mathbf{x}}_i^\tau(\bar{k}) \to \hat{\mathbf{x}}_i^*(\bar{k})$ *as* $\tau \to \infty$. □

The description above implicitly assumes that the iterations are executed synchronously, i.e., there is a common iteration counter τ among all the agents. However, the result in Proposition 1 holds even if communication and computation is performed in an asynchronous way, where every agent has its own iteration counter τ. This follows from results in asynchronous iterations [4].

3.3.2 Finite Memory and Bandwidth

The algorithm for localization of mobile agents with finite memory and finite bandwidth is now presented. To ease later discussion, this algorithm is called the Mobile Agent Iterative Algorithm (or MAI algorithm).

Mobile Agent Iterative Algorithm

In the description below, κ is a fixed positive integer that denotes the "size" of the subgraph of $\mathcal{G}(\bar{k})$ that every agent maintains in its local memory at time $\bar{k}$. The parameter κ is essentially fixed ahead of time and provided to all agents; its value is determined by the memory available to each agent's local processor. The maximum number of iterations carried out by an agent during the interval between times $\bar{k}$ and $\bar{k}+1$, is denoted by $\bar{\tau}$ and depends on the maximum number of communication rounds between Agent i and its neighbors during this interval.

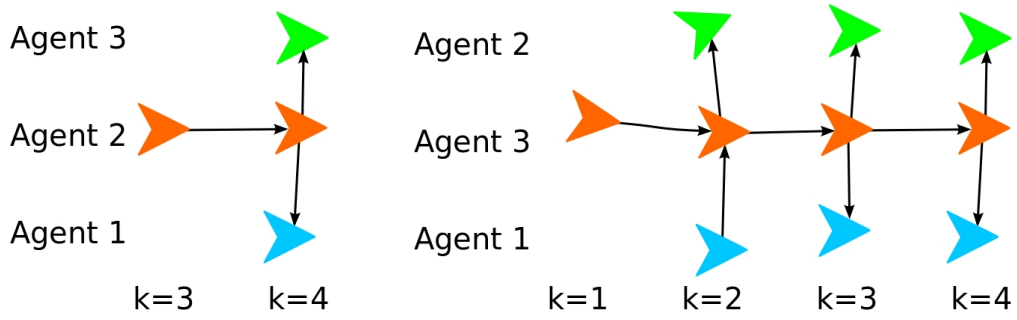
Let $\mathcal{G}(\bar{k})^{\kappa} = (\mathcal{V}(\bar{k})^{\kappa}, \mathcal{E}(\bar{k})^{\kappa})$ be the subgraph containing the nodes, $\mathcal{V}(\bar{k})^{\kappa}$, associated with all agent positions from time max $(\bar{k} - \kappa, 0)$ until time $\bar{k}$ and containing edges, $\mathcal{E}(\bar{k})^{\kappa}$, corresponding to relative measurements between two nodes in $\mathcal{G}(\bar{k})^{\kappa}$. More simply, $\mathcal{G}(\bar{k})^{\kappa}$ is the subgraph containing all nodes and edges corresponding to positions of agents and relative measurements at the current and previous κ time instants. In this case, $\hat{\mathbf{x}}_i^{\tau}(\bar{k})$ is a vector of the estimates of the positions of Agent i from time max $(\bar{k}-\kappa, 0)$ to time $\bar{k}$, obtained in the τ^{th} iteration. During the interval between time indices $\bar{k}$ and $\bar{k}+1$, each agent, i updates the estimate of $\mathbf{x}_i(\bar{k})$ in the following way.

1. initialization: $\hat{x}_i^{(0)}(\bar{k}) = \hat{x}_i^{(\bar{\tau})}(\bar{k}-1) + u_i(\bar{k}-1)$.

2. collect inter-agent measurements that occurred at time $\bar{k}$, i.e., obtain $y_{ii'}(\bar{k})$ for neighbor agents, i'.

3. iterative update: Agent i treats its neighbors estimates, as well as its oldest estimate as reference variables and iteratively updates its position by the algorithm described in the previous section. Specifically, it runs the Jacobi algorithm for the subgraph $\mathcal{G}_i(\bar{k})^{\kappa}$. □

Because each agent treats its oldest estimate as a reference, these estimates are fixed and cannot change. Due to truncation, older measurements are no longer used in computing estimates but, the information that is contained in



(a) $\kappa = 1$ (b) $\kappa = 3$

Figure 3.3: Truncated subgraphs, $\mathcal{G}_3(4)$ of Agent 3 at time 4 for the measurement graph shown in Figure 3.1.

these measurements still has an affect on the current estimates (a la Kalman filtering). The penalty for truncating the measurement graph prior to the BLUE being reached is that some error is fixed into future estimates. Later it will be shown that even with a short history of data, this error tends to be small.

The MAI algorithm continues as long as the agents continue to move. Figure 3.3 shows an example of the local subgraphs used by an agent (agent 3 in Figure 3.1) for two cases, $\kappa = 1$ and $\kappa = 3$. Note that the proposed algorithm is particularly simple when $\kappa = 1$, since in that case the iterative update is simply the solution to the following equation:

$$S_i(\bar{k}-1)\hat{x}_i^{\tau}(\bar{k}) = Q_i^{-1}(\bar{k}-1)\Big(\hat{x}_i^{\bar{\tau}}(\bar{k}-1) + u_i(\bar{k}-1)\Big) + \sum_{i' \in \mathcal{N}_i(\bar{k})} R_{ii'}^{-1}(\bar{k})\Big(\hat{x}_{i'}^{\tau-1}(\bar{k}) + y_{ii'}(\bar{k})\Big),$$

where $Q_i(\bar{k}) := c\text{Cov}(w_i(\bar{k}), w_i^\top(\bar{k}))$ and $R_{ii'}(\bar{k}) := \text{Cov}(v_{ii'}(\bar{k}), v_{ii'}^\top(\bar{k}))$ are the error covariances in the displacement measurement $u_i(\bar{k}-1)$ and inter-agent relative measurement $y_{ii'}(\bar{k})$, respectively, and $S_i(\bar{k}) := Q_i^{-1}(\bar{k}) + \sum_{i' \in \mathcal{N}_i(\bar{k})} R_{ii'}^{-1}(\bar{k})$. When all the measurement error covariances are equal, the update is simply:

$$\hat{x}_i^\tau(\bar{k}) = \frac{1}{\mid \mathcal{N}_i(\bar{k}) + 1 \mid} \big(\hat{x}_i^{\bar{\tau}}(\bar{k}-1) + u_i(\bar{k}-1) + \sum_{i' \in \mathcal{N}_i(\bar{k})} (\hat{x}_i'^{\tau-1}(\bar{k}) + y_{ii'}(\bar{k})) \big)$$

When $\kappa = \infty$, the MAI algorithm is simply the Jacobi iterations to compute the BLUE estimates of all the node variables in the graph $\mathcal{G}(\bar{k})$, i.e., the past and present positions of the agents. In this case, Proposition 1 guarantees that the estimates computed converge to the BLUE estimates as $\bar{\tau} \to \infty$. When the MAI algorithm is implemented with small values of κ, after a certain number of time steps, measurements from times earlier than κ steps into the past are no longer directly used. However, the information from past measurements is still used indirectly, since it affect the values of the reference variables used by the agents for their local subgraphs.

With finite κ, the estimates typically do not reach their centralized optimal because, while the measurement before time $k - \kappa$ are not completely forgotten (as explained above) they are not incorporated in the optimal fashion. A further reduction in accuracy comes from the fact that in practice $\bar{\tau}$ may not be large

enough to get close to convergence even in the truncated local subgraphs. The MAI algorithm therefore produces an approximation of the centralized optimal estimates; the approximation becomes more accurate as $\bar{\tau}$ and κ increase.

Communication and Computation Costs

The amount of data an agent needs to store and broadcast increases as the "size" of the truncated local subgraph that the agent keeps in local memory increases, and therefore, on κ. When the neighbors of an agent do not change with time, the number of nodes in the local truncated subgraph of an agent at any given time is $\kappa+\eta_{\text{nbr}}\kappa$, where η_{nbr} is the number of neighbors of the agent. In this case, the number of edges in the truncated local subgraph is at most $\kappa+\kappa\eta_{\text{nbr}}$ (the first term is the number of inertial measurements and the second term is the number of relative measurements between the agent and its neighboring agents that appear as edges in the subgraph). Therefore, an agent needs a local memory large enough to store $d[2\kappa(1+\eta_{\text{nbr}})+\kappa\eta_{\text{nbr}}]$ floating-point numbers, where $d = 2$ or 3 depending on whether positions are 2 or 3 dimensional vectors. An agent has to broadcast the current estimates of its interior node variables, i.e., $d\kappa$ numbers, at every iteration. Thus, the communication, computation and memory requirements of the MAI algorithm are quite low for small values of κ. It

is assumed that the agents can obtain the error covariances of the measurements on the edges that are incident on themselves, so these need not be transmitted.

3.4 Target Tracking

The problem formulation in Section 3.1 assumes that all agents compute temporal displacement measurements. In some cases, such as target tracking, there may be agents external to the network, that do not provide such measurements to the network. Such agents are referred to as *target agents* and the remaining agents will be called *cooperative agents*. This section shows that by assigning cooperative agents to estimate track of target agents' states, it is possible to incorporate target agents with linear dynamics into the estimation framework introduced above.

3.4.1 Extension to General Linear Dynamics

Recall from the distributed algorithms presented in Section 3.3 that, at each iteration, each agent computes the BLUE of its local subgraph. Previously, only absolute and relative position measurements were considered, for which the local BLUE is solvable if every node in the subgraph is connected to at least one reference node.

Consider now a single target agent with with state $\mathbf{x}_{\mathrm{T}}(k)$, that moves according to the linear model

$$\mathbf{x}_{\mathrm{T}}(k+1) = F(k)\mathbf{x}_{\mathrm{T}}(k) + G(k)\mathbf{u}_{\mathrm{T}}(k) + \mathbf{w}_{\mathrm{T}}(k), \tag{3.7}$$

where $\mathrm{Cov}(\mathbf{w}_{\mathrm{T}}(k), \mathbf{w}_{\mathrm{T}}(k)) = Q_{\mathrm{T}}(k)$ is the target model noise. Now, the target's state is allowed to be multidimensional (i.e. position and velocity).

Along with the target model, measurements of the target are made relative to a reference state, $\mathbf{x}_{\mathcal{R}}(k)$,

$$\mathbf{y}_{\mathrm{T}}(k) = H_{\mathcal{R}}(k)\mathbf{x}_{\mathcal{R}}(k) + H_{\mathrm{T}}(k)\mathbf{x}_{\mathrm{T}}(k) + \mathbf{v}_{\mathrm{T}}(k). \tag{3.8}$$

Here $\mathrm{Cov}(\mathbf{v}_{\mathrm{T}}(k), \mathbf{v}_{\mathrm{T}}(k)) = R_{\mathrm{T}}(k)$ is observation noise and $H(k) = [H(k)_{\mathcal{R}}\ H(k)_{\mathrm{T}}]$ is an observation matrix that is appropriately defined so entries of $\mathbf{y}_{\mathrm{T}}(k)$ are the measurements of the target's position relative to the reference state. The noise vectors, $\mathbf{w}_{\mathrm{T}}(k)$ and $\mathbf{v}_{\mathrm{T}}(k)$ are assumed to be uncorrelated with time, uncorrelated with the network's state, and uncorrelated with the *model input* vector, $\mathbf{u}_{\mathrm{T}}(k)$.

As was done previously, the model equations (3.7), are rearranged to make this problem fit the relative measurement framework

$$G(k)\mathbf{u}_{\mathrm{T}}(k) = F(k)\mathbf{x}_{\mathrm{T}}(k) - \mathbf{x}_{\mathrm{T}}(k+1) - \mathbf{w}_{\mathrm{T}}(k).$$

Also, subtracting the reference state from both sides of the measurements in (3.8) results in

$$\mathbf{y}_{\mathrm{T}}(k) - H_{\mathcal{R}}(k)\mathbf{x}_{\mathcal{R}}(k) = H_{\mathrm{T}}(k)\mathbf{x}_{\mathrm{T}}(k) + \mathbf{v}_{\mathrm{T}}(k).$$

Now stack the target measurements and modeled dynamics into a single vector

$$\begin{bmatrix} \mathbf{y}_{\mathrm{T}}(1) \\ G(1)\mathbf{u}_{\mathrm{T}}(1) \\ \mathbf{y}_{\mathrm{T}}(2) \\ G(2)\mathbf{u}_{\mathrm{T}}(2) \\ \vdots \\ G(\bar{k}-1)\mathbf{u}_{\mathrm{T}}(\bar{k}-1) \\ \mathbf{y}_{\mathrm{T}}(\bar{k}) \end{bmatrix} - \begin{bmatrix} H_{\mathcal{R}}(1) & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & H_{\mathcal{R}}(2) & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & H_{\mathcal{R}}(\bar{k}) \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}}(1) \\ \mathbf{x}_{\mathcal{R}}(2) \\ \vdots \\ \mathbf{x}_{\mathcal{R}}(\bar{k}-1) \\ \mathbf{x}_{\mathcal{R}}(\bar{k}) \end{bmatrix} =$$

$$\begin{bmatrix} H_{\mathrm{T}}(1) & 0 & \cdots & 0 & 0 \\ F(1) & -\mathrm{I} & \cdots & 0 & 0 \\ 0 & H_{\mathrm{T}}(2) & \cdots & 0 & 0 \\ 0 & F(2) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & F(\bar{k}-1) & -\mathrm{I} \\ 0 & 0 & \cdots & 0 & H_{\mathrm{T}}(\bar{k}) \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathrm{T}}(1) \\ \mathbf{x}_{\mathrm{T}}(2) \\ \vdots \\ \mathbf{x}_{\mathrm{T}}(\bar{k}-1) \\ \mathbf{x}_{\mathrm{T}}(\bar{k}) \end{bmatrix} + \begin{bmatrix} -\mathbf{w}_{\mathrm{T}}(1) \\ \mathbf{v}_{\mathrm{T}}(1) \\ -\mathbf{w}_{\mathrm{T}}(2) \\ \mathbf{v}_{\mathrm{T}}(2) \\ \vdots \\ \mathbf{v}_{\mathrm{T}}(\bar{k}-1) \\ -\mathbf{w}_{\mathrm{T}}(\bar{k}) \end{bmatrix}.$$

For simplicity, this equation is rewritten as

$$\tilde{\mathbf{z}}_{\mathrm{T}} - H_{\mathcal{R}}\mathbf{x}_{\mathcal{R}} = H_{\mathrm{T}}\mathbf{x}_{\mathrm{T}} + \boldsymbol{\epsilon}_{\mathrm{T}},$$

where $\mathbf{x}_{\mathrm{T}}$ denotes the target state for all $k \in \left\{1, 2, \ldots, \bar{k}\right\}$, $\tilde{\mathbf{z}}_{\mathrm{T}}$ denotes measurements of linear combinations of the target state, and $\boldsymbol{\epsilon}_{\mathrm{T}}$ denotes the noise associated with these measurements. Denote by P_{T} the covariance of $\boldsymbol{\epsilon}_{\mathrm{T}}$.

Following the conventions established earlier, the BLUE of target state given $\tilde{\mathbf{z}}_{\mathrm{T}}$ is defined to be

$$\hat{\mathbf{x}}_{\mathrm{T}}^{*} \equiv (H_{\mathrm{T}} P_{\mathrm{T}}^{-1} H_{\mathrm{T}}^{\top})^{-1} H_{\mathrm{T}} P_{\mathrm{T}}^{-1} (\tilde{\mathbf{z}}_{\mathrm{T}} - H_{\mathcal{R}} \mathbf{x}_{\mathcal{R}}). \tag{3.9}$$

For the BLUE to be feasible, conditions for which (3.9) has a solution must be established.

Theorem 1 *[Solvability of the BLUE with Modeled Dynamics]*
The BLUE, (3.9)*, is solvable if and only if the stochastic linear system given by* (3.7) *and* (3.8) *is observable at each time* k*.*

Proof: For the BLUE to exist $H_{\mathrm{T}} P_{\mathrm{T}}^{-1} H_{\mathrm{T}}^{\top}$ must be invertible. For $\bar{k} > 1$, H_{T} has more rows than columns. Furthermore, P_{T} is a symmetric positive definite matrix. Therefore, $H_{\mathrm{T}} P_{\mathrm{T}}^{-1} H_{\mathrm{T}}^{\top}$ is invertible if and only if H_{T} is full column rank. It is shown here that this is the case if and only if the linear system described by (3.7) and (3.8) is observable at time $\bar{k}$.

First, it is shown that if H_{T} is full column rank, then the system is observable at time $\bar{k}$. Assume for contra-position that the system is not observable at time $\bar{k}$, then there exists a vector $\mathbf{x}_{\mathrm{T}}(1) \neq 0$ such that

$$\begin{bmatrix} H_{\mathrm{T}}(1) \\ H_{\mathrm{T}}(2)F(1) \\ \vdots \\ H_{\mathrm{T}}(\bar{k})F(\bar{k}-1)\dots F(1) \end{bmatrix} \mathbf{x}_{\mathrm{T}}(1) = 0.$$

Furthermore,

$$H_{\mathrm{T}} \begin{bmatrix} \mathrm{I} \\ F(1) \\ \vdots \\ F(\bar{k}-2)F(\bar{k}-3)\dots F(1) \\ F(\bar{k}-1)F(\bar{k}-2)\dots F(1) \end{bmatrix} \mathbf{x}_{\mathrm{T}}(1) = \begin{bmatrix} H_{\mathrm{T}}(1) \\ 0 \\ H_{\mathrm{T}}(2)F(1) \\ \vdots \\ 0 \\ H_{\mathrm{T}}(\bar{k})F(\bar{k}-1)\dots F(1) \end{bmatrix} \mathbf{x}_{\mathrm{T}}(1) = 0$$

As H_{T} was multiplied by a non-trivial matrix and the result was zero, H_{T} is not full rank. Contra-position leads to the conclusion that if H_{T} is full rank then (3.7) and (3.8) is observable at time $\bar{k}$.

Now, it is shown that if the system in (3.7) and (3.8) is observable at time $\bar{k}$ then H_{T} is full column rank. Assume for contradiction that the system is observable at time k and H_{T} is not full column rank. Because H_{T} is not full column rank, there exists vector, $\mathbf{x}_{\mathrm{T}} = [\mathbf{x}_{\mathrm{T}}(1)^{\top}\mathbf{x}_{\mathrm{T}}(2)^{\top}\dots\mathbf{x}_{\mathrm{T}}(\bar{k})^{\top}]^{\top}$, such that $H_{\mathrm{T}}\mathbf{x}_{\mathrm{T}} = 0$. Expanding on this,

$$H_{\mathrm{T}}\mathbf{x}_{\mathrm{T}} = \begin{bmatrix} H_{\mathrm{T}}(1)\mathbf{x}_{\mathrm{T}}(1) \\ \mathbf{x}_{\mathrm{T}}(2) - F(1)\mathbf{x}_{\mathrm{T}}(1) \\ \vdots \\ \mathbf{x}_{\mathrm{T}}(\bar{k}) - F(k-1)\mathbf{x}_{\mathrm{T}}(\bar{k}-1) \\ H(\bar{k}) \end{bmatrix} = 0.$$

Consider only even rows of $H_{\mathrm{T}}\mathbf{x}_{\mathrm{T}}$, for $k \in \left\{2, 3, \dots, \bar{k}\right\}$

$$\begin{aligned} x_{\mathrm{T}}(k) &= F(k-1)x_{\mathrm{T}}(k-1) \\ &= F(k-1)F(k-2)\dots F(1)x_{\mathrm{T}}(1). \end{aligned} \tag{3.10}$$

Substitution (3.10) into the odd rows of $H_{\mathrm{T}}\mathbf{x}_{\mathrm{T}}$,

$$\begin{bmatrix} H_{\mathrm{T}}(1)\mathbf{x}_{\mathrm{T}}(1) \\ H_{\mathrm{T}}(2)\mathbf{x}_{\mathrm{T}}(2) \\ \vdots \\ H_{\mathrm{T}}(\bar{k})\mathbf{x}_{\mathrm{T}}(\bar{k}) \end{bmatrix} = \begin{bmatrix} H_{\mathrm{T}}(1)\mathbf{x}_{\mathrm{T}}(1) \\ H_{\mathrm{T}}(2)F(1)\mathbf{x}_{\mathrm{T}}(1) \\ \vdots \\ H_{\mathrm{T}}(\bar{k})F(\bar{k}-1)\dots F(1)\mathbf{x}_{\mathrm{T}}(1) \end{bmatrix} = \mathcal{O}(\bar{k})\mathbf{x}_{\mathrm{T}}(1) = 0,$$

where $\mathcal{O}(\bar{k})$ is the observability matrix at time $\bar{k}$. This contradicts the assumption that (3.7) and (3.8) is observable at time $\bar{k}$. ■

Note that the inclusion of modeled dynamics can be extended from target agents to all agents. So long as the measurements and models of each agents local BLUE are observable, the local BLUEs are solvable and the larger iterative Jacobi procedure is tractable.

3.4.2 General Partitioning of the Measurement Graph

In the target tracking problem, it is assumed that the target agent is non-cooperative. Because of this, a cooperative agent must take responsibility for estimating the target's state, that agent is called the *tracking agent*. One naïve way to accomplish this is to have the tracking agent perform two optimizations per iteration, one for itself and one for the target. This would work, but there is a simpler method that speeds convergence.

In Section 3.2 a graph was used to represent the distributed estimation problem. In Section 3.3.1 the graph's vertex set was partitioned into η independent subsets $\mathcal{V}_i^{\text{int}}, i \in \{1, 2, \dots \eta\}$, one for each agent. Each agent then considered only a subgraph consisting of their nodes, all neighboring nodes, and the edges connecting said nodes. An iterative Jacobi algorithm was employed to compute the BLUE of the system's parameters. In this algorithm, each agent received their neighbors position estimates and computed the BLUE of their local subgraph assuming their neighbors estimates were noiseless. This case employed a partitioning of the vertex set that was intuitive because each subset contained nodes belonging to a single agent.

Now, it is shown that the convergence results for distributed algorithms in Section 3.3 are preserved under an arbitrary partitioning of the vertex set. Assume that the vertex set $\mathcal{V}$ is partitioned into μ independent subsets, $\mathcal{V}_i, i \in \{1, 2, \dots \mu\}$. To analyze the block-Jacobi method, the nodes of the graph are first indexed so that nodes belonging to a given partition are indexed sequentially.

With this indexing, the system of equations in (3.5) can be written

$$\tilde{\mathbf{z}} - B_{\mathcal{R}}^{\top}\mathbf{x}_{\mathcal{R}} = \begin{bmatrix} B_1^{\top} & B_2^{\top} & \dots & B_{\mu}^{\top} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{\mu} \end{bmatrix} + \boldsymbol{\epsilon},$$

where the vector $\mathbf{x}_i$ contains all of the node variables in the partition i, $B = \begin{bmatrix} B_1^{\top} & B_2^{\top} & \dots & B_{\mu}^{\top} \end{bmatrix}^{\top}$ denotes the generalized incidence matrix for the unknown states, and $\boldsymbol{\epsilon}$ the noise vector. The generalized Laplacian matrix, $\mathcal{L}$, is a block matrix defined to be

$$\mathcal{L} \equiv BP^{-1}B^{\top} = \begin{bmatrix} B_1P^{-1}B_1^{\top} & B_1P^{-1}B_2^{\top} & \dots & B_1P^{-1}B_{\mu}^{\top} \\ B_2P^{-1}B_1^{\top} & B_2P^{-1}B_2^{\top} & \dots & B_2P^{-1}B_{\mu}^{\top} \\ \vdots & \vdots & \ddots & \vdots \\ B_{\mu}P^{-1}B_1^{\top} & B_{\mu}P^{-1}B_2^{\top} & \dots & B_{\mu}P^{-1}B_{\mu}^{\top} \end{bmatrix}$$

where P denotes the covariance of $\boldsymbol{\epsilon}$. The generalized Laplacian matrix can be split into $\mathcal{L} = D - A$, where the generalized degree matrix is

$$D = \begin{bmatrix} B_1 P^{-1} B_1^\top & 0 & \dots & 0 \\ 0 & B_2 P^{-1} B_2^\top & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_\mu P^{-1} B_\mu^\top \end{bmatrix},$$

and the generalized adjacency matrix is

$$A = \begin{bmatrix} 0 & -B_1 P^{-1} B_2^\top & \dots & -B_1 P^{-1} B_\mu^\top \\ -B_2 P^{-1} B_1^\top & 0 & \dots & -B_2 P^{-1} B_\mu^\top \\ \vdots & \vdots & \ddots & \vdots \\ -B_\mu P^{-1} B_1^\top & -B_\mu P^{-1} B_2^\top & \dots & 0 \end{bmatrix}.$$

As previously discussed, the BLUE is the unique solution for (3.6), which can be rewritten as $D\mathbf{x} = A\mathbf{x} + \mathbf{b}$, which leads to the following block-Jacobi iterative method for solving it:

$$\hat{\mathbf{x}}(\tau + 1) = D^{-1} A \hat{\mathbf{x}}(\tau) + D^{-1}\mathbf{b} \tag{3.11}$$

where A is a sparse matrix . In particular, the $\{i, i'\}$ block element of A is non-zero if and only if $\mathcal{V}_i$ and $\mathcal{V}_{i'}$ contain nodes that are neighbors in $\mathcal{G}$ (i.e. $B_i B_{i'}^\top \neq 0$). This sparsity makes it possible to compute the updates in (3.11) in a distributed manner; each $\mathcal{V}_i$ only needs to communicate with its neighboring graph partitions. The following theorem is a convergence result for the block-Jacobi algorithm under arbitrary partitioning.

Theorem 2 *[Convergence of Block-Jacobi Iteration Under General Graph Partitioning]*

Given an arbitrary partitioning of the graph's node set, $\mathcal{V}$, if B is full row rank and $B_i B_i^\top$ is invertible for each partition, then the block-Jacobi algorithm, (3.15)*, converges.*

Proof: The block-Jacobi algorithm, (3.15), converges if and only if $\rho(D^{-1}A) < 1$. If B is full row rank, then $\mathcal{L}$ is positive definite. This comes about because P is positive definite (it is a covariance matrix) and $\mathcal{L}$ is an inner product of B weighted by P^{-1}. Further more, if $B_i B_i^\top$ is invertible for every partition, then each diagonal block of D is positive definite, rendering D positive definite. Finally, with $\mathcal{L}$ and D positive definite, the following inequalities

hold

$$\mathcal{L} = D - A > 0$$

$$\rho(D^{-1}\mathcal{L}) = \rho(I - D^{-1}A) > 0$$

$$1 > \rho(D^{-1}A).$$

■

It should be noted that when $B_i B_i^\top$ is invertible the BLUE for the i^{th} subgraph is computable. This condition is true when considering only relative difference measurements, if every node is connected to at least one reference node. More generally, this condition is true when the conditions in Theorem 1 are met.

From this result, it is evident that the vertex set for a tracking agent can be extended to include the nodes belonging to the target. The tracking agent then implements the MAI algorithm and simultaneously estimates its own state, as well as the states belonging to the target agent. By doing the computation concurrently the tracking agent reduces communication and convergence time for the algorithm.

The idea of partitioning the vertex set for use in distributed estimation has two extremes. In the most distributed case, each agent has only one node in

its vertex set. This was presented in [4] . In the least distributed case, there is no partitioning of the vertex set and state estimation is done in a centralized manner. The fact that any partitioning of the vertex set leads to an optimal result is useful because one can cluster nodes together on any level.

3.5 Analysis

This section provides analysis for the convergence rate of the MAI algorithm. First the convergence properties of the algorithm with infinite memory (i.e. estimating all unknown positions) are considered. Then the convergence properties of the algorithm with finite memory are examined. As with algorithms of this sort, convergence rates are highly dependent on topology. As such, a network that resembles a rectangular lattice is considered. For a square lattice graph, and uniform measurement covariances, tight bounds for the infinite memory case are provided. For the case of finite memory, the trade-off between memory length and number of iterations at each step is shown. Even under modest computational requirements (i.e. few iterations and a short memory length), the MAI algorithm provides estimates that are much better than non-cooperative estimates; commonly referred to as *dead reckoning*.

3.5.1 Convergence Rate on a Grid Network

Consider again a network of η agents that make d-dimensional synchronous inter-agent measurements. Furthermore, assume that the inter-agent measurements at a particular time make a chain graph and that the neighbor relations are time invariant. Suppose that the agent's positions at time $k = 0$ are known exactly from GPS but that the agents lose GPS and time elapses until some time, $\bar{k}$. The corresponding measurement graph is an η by $\bar{k}+1$ grid, rendering the graph shown in Figure 3.4.

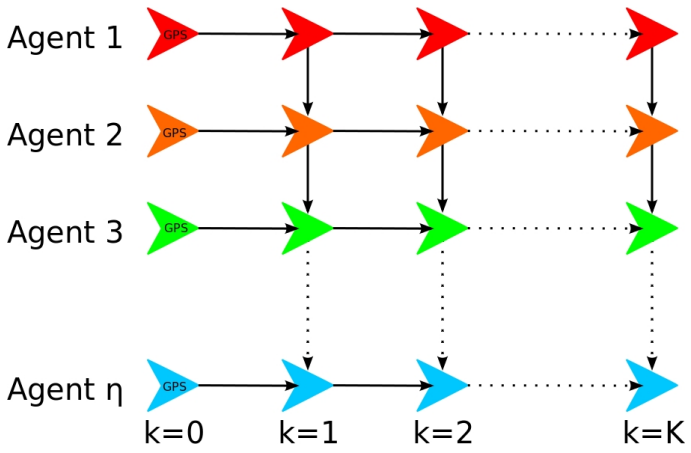


Figure 3.4: Agents states as a grid.

To determine the convergence rate of the MAI algorithm for agents with infinite memory, the convergence rate of the BLUE algorithm on the graph shown in Figure 3.4 must be determined. To start, the displacement measurement of an agent given in (3.1) is reintroduced. Denote by u_i the $d\bar{k}$-length vector of

ordered displacement measurements for Agent i up to time $\bar{k}$,

$$\mathbf{u}_i = \left[\mathbf{u}_i(0)^\top, \mathbf{u}_i(1)^\top, \ldots, \mathbf{u}_i(\bar{k})^\top\right]^\top.$$

Similarly, denote by $\mathbf{y}_{ii'}$ the $d\bar{k}$-length vector of ordered relative position measurements between Agent i and Agent i' up to time $\bar{k}$,

$$\mathbf{y}_{ii'} = \left[\mathbf{y}_{ii'}(0)^\top, \mathbf{y}_{ii'}(1)^\top, \ldots, \mathbf{y}_{ii'}(\bar{k})^\top\right]^\top.$$

Denote by $\mathbf{x}_{\mathcal{R}i}$ the position of Agent i at $k = 0$ (the GPS fix) and denote by $\mathbf{x}_i$ the $d\bar{k}$-length vector of Agent i's unknown positions ordered chronologically. Furthermore, set the notation for error vectors to match the notation for measurements; $\mathbf{u}_i$ has error $\mathbf{w}_i$ and $\mathbf{y}_{ii'}$ has error $\mathbf{v}_{ii'}$. Denote by Q_i and $R_{ii'}$ the covariance matrices of $\mathbf{w}_i$ and $\mathbf{v}_{ii'}$, respectively.

With these structures properly defined, the network of measurements can be written

$$\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_{12} \\ \mathbf{u}_2 \\ \mathbf{y}_{23} \\ \vdots \\ \mathbf{y}_{\eta\,\eta-1} \\ \mathbf{u}_\eta \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & B_{\mathrm{I}}^\top & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\mathrm{I}_{d\bar{k}} & 0 & \mathrm{I}_{d\bar{k}} & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{e}_1 & B_{\mathrm{I}}^\top & \cdots & 0 & 0 \\ 0 & 0 & 0 & -\mathrm{I}_{d\bar{k}} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \mathrm{I}_{d\bar{k}} \\ 0 & 0 & 0 & 0 & \cdots & \mathbf{e}_1 & B_{\mathrm{I}}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}1} \\ \mathbf{x}_1 \\ \mathbf{x}_{\mathcal{R}2} \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{\mathcal{R}\eta} \\ \mathbf{x}_\eta \end{bmatrix} + \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{v}_{12} \\ \mathbf{w}_2 \\ \mathbf{v}_{23} \\ \vdots \\ \mathbf{v}_{\eta\,\eta-1} \\ \mathbf{w}_\eta \end{bmatrix}, \quad (3.12)$$

where $\mathbf{e}_1$ denotes the Kronecker product of length $\bar{k}$ elementary vector and the size d identity matrix, $\mathbf{e}_1 = [\mathrm{I}_d, 0_d, \ldots, 0_d]^\top$, 0 denotes the appropriately sized zero matrix, and B_{I} denotes the Kronecker product of the incidence matrix of an agent's non-GPS nodes and the size d identity matrix. From the problem formulation, an agent's nodes form a chain graph; B_{I} is a square matrix containing

the last $d\bar{k}$ rows of an incidence matrix representing a chain graph. In general,

$$B_{\mathrm{I}}^{\top} = \begin{bmatrix} -\mathrm{I}_d & 0 & 0 & \cdots & 0 & 0 \\ \mathrm{I}_d & -\mathrm{I}_d & 0 & \cdots & 0 & 0 \\ 0 & \mathrm{I}_d & -\mathrm{I}_d & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\mathrm{I}_d & 0 \\ 0 & 0 & 0 & \cdots & \mathrm{I}_d & -\mathrm{I}_d \end{bmatrix}.$$

For the special case when $\bar{k} = 1$, $B_{\mathrm{I}}^{\top} = [-\mathrm{I}_d]$.

The Best Linear Unbiased Estimate

The measurement vector, presented in (3.12) is a linear combination of reference states, $x_{\mathcal{R}i}$, and unknown states, x_i, corrupted by noise. The first step in computing the BLUE is to transform (3.12) into a linear system of unknown states. This is accomplished by subtracting the reference states from both sides

of the measurement equation

$$\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_{12} \\ \mathbf{u}_2 \\ \mathbf{y}_{23} \\ \vdots \\ \mathbf{y}_{\eta\,\eta-1} \\ \mathbf{u}_\eta \end{bmatrix} - \begin{bmatrix} \mathbf{e}_1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & \mathbf{e}_1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \mathbf{e}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}1} \\ \mathbf{x}_{\mathcal{R}2} \\ \vdots \\ \mathbf{x}_{\mathcal{R}\eta} \end{bmatrix} = \begin{bmatrix} B_{\mathrm{I}}^\top & 0 & \cdots & 0 \\ -\mathrm{I}_{d\bar{k}} & \mathrm{I}_{d\bar{k}} & \cdots & 0 \\ 0 & B_{\mathrm{I}}^\top & \cdots & 0 \\ 0 & -\mathrm{I}_{d\bar{k}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{I}_{d\bar{k}} \\ 0 & 0 & \cdots & B_{\mathrm{I}}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_\eta \end{bmatrix} + \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{v}_{12} \\ \mathbf{w}_2 \\ \mathbf{v}_{23} \\ \vdots \\ \mathbf{v}_{\eta\,\eta-1} \\ \mathbf{w}_\eta \end{bmatrix}. \tag{3.13}$$

To simplify this equation the following is introduced

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{u}_1^\top & \mathbf{y}_{12}^\top & \mathbf{u}_2^\top & \mathbf{y}_{23}^\top & \cdots & \mathbf{y}_{\eta\,\eta-1}^\top & \mathbf{u}_\eta^\top \end{bmatrix}^\top,$$

$$\mathbf{x}_{\mathcal{R}} = \begin{bmatrix} \mathbf{x}_{\mathcal{R}1}^\top \mathbf{e}_1^\top & 0 & \mathbf{x}_{\mathcal{R}2}^\top \mathbf{e}_1^\top & 0 & \cdots & \mathbf{x}_{\mathcal{R}\eta}^\top \mathbf{e}_1^\top \end{bmatrix}^\top,$$

$$B = \begin{bmatrix} B_{\mathrm{I}} & -\mathrm{I}_{d\bar{k}} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathrm{I}_{d\bar{k}} & B_{\mathrm{I}} & -\mathrm{I}_{d\bar{k}} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mathrm{I}_{d\bar{k}} & B_{\mathrm{I}} \end{bmatrix},$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^\top & \mathbf{x}_2^\top & \cdots & \mathbf{x}_\eta^\top \end{bmatrix}^\top,$$

$$\boldsymbol{\epsilon} = \begin{bmatrix} \mathbf{w}_1^\top & \mathbf{v}_{12}^\top & \mathbf{w}_2^\top & \mathbf{v}_{23}^\top & \cdots & \mathbf{v}_{\eta\,\eta-1}^\top & \mathbf{w}_\eta \end{bmatrix}^\top,$$

$$P = \begin{bmatrix} Q_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & R_{12} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & Q_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & R_{23} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & R_{\eta-1\,\eta} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & Q_\eta \end{bmatrix}.$$

Now, (3.13) can be written

$$\tilde{\mathbf{z}} - \mathbf{x}_{\mathcal{R}} = B^\top \mathbf{x} + \boldsymbol{\epsilon}.$$

The BLUE, $\hat{x}^*$, of x is defined to be

$$\hat{\mathbf{x}}^* \equiv (BP^{-1}B^\top)^{-1}BP^{-1}(\tilde{\mathbf{z}} - \mathbf{x}_{\mathcal{R}}). \tag{3.14}$$

The Block-Jacobi Iterative Method

The algorithms presented in this chapter use the block-Jacobi iterative method (introduced in section 3.2) to compute the BLUE. Now the the block-

Jacobi method as it applies to computing (3.14) is examined. The BLUE, $\hat{x}^*$, is the unique solution to the linear equation $\mathcal{L}\mathbf{x} = \mathbf{b}$. To implement the block-Jacobi method the matrix $\mathcal{L} = D - A$ is split so that D contains the block diagonal elements of $\mathcal{L}$. Then $A\mathbf{x}$ is added to both sides of the linear equation to form an estimate update law in accordance with the block-Jacobi method:

$$\hat{\mathbf{x}}(\tau + 1) = D^{-1}A\hat{\mathbf{x}}(\tau) + D^{-1}\mathbf{b}. \tag{3.15}$$

Note that the update law is simply a discrete-time linear system. As such, convergence is determined solely by the eigenvalues of $D^{-1}A$.

To simplify further discussions, let the *network iteration matrix* be defined as

$$J \equiv D^{-1}A.$$

Denote by $\rho(J)$ the magnitude of the largest eigenvalue of J – commonly known as the spectral radius of J. Because J characterizes the dynamics of the Jacobi Algorithm, a discrete-time linear system, convergence of the algorithm is dependent on $\rho(J)$. It was shown in Theorem 2 that for this problem, the block-Jacobi method converges. Now it is shown to converge to the BLUE.

Theorem 3 *[Convergence to the BLUE]*

Given that the block-Jacobi iteration in (3.15) *converges,*

$$\lim_{\tau\to\infty} \hat{\mathbf{x}}(\tau) = \hat{\mathbf{x}}^*.$$

Proof: When (3.15) is convergent,

$$\hat{\mathbf{x}}(\infty) = D^{-1}A\hat{\mathbf{x}}(\infty) + D^{-1}\mathbf{b}.$$

From this

$$\begin{aligned}\hat{\mathbf{x}}(\infty) =& (D-A)^{-1}\mathbf{b}\\ =& (BP^{-1}B^\top)^{-1}BP^{-1}(\tilde{\mathbf{z}} - \mathbf{x}_\mathcal{R}) = \hat{\mathbf{x}}^*.\end{aligned}$$

■

Theorem 4 *[Rate of Convergence for Scalar Covariance Matrices]*

If the covariances, $P_{\{.\}}$, for all measurements in the network are equal, then

$$\rho(J) \leq \frac{2}{4 - 2cos(\frac{\pi}{2k+1})}.$$

Proof: If all measurements have scalar covariance matrix, $P_{\mathrm{s}} = \alpha \mathrm{I}$ for $\alpha \in \mathbb{R}$, the iteration matrix J can be written as

$$J = \begin{bmatrix} 0 & J_{\mathrm{e}} & 0 & \cdots & 0 & 0 & 0 \\ J_{\mathrm{c}} & 0 & J_{\mathrm{c}} & \cdots & 0 & 0 & 0 \\ 0 & J_{\mathrm{c}} & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & J_{\mathrm{c}} & 0 \\ 0 & 0 & 0 & \cdots & J_{\mathrm{c}} & 0 & J_{\mathrm{c}} \\ 0 & 0 & 0 & \cdots & 0 & J_{\mathrm{e}} & 0 \end{bmatrix},$$

where

$$\begin{aligned} J_{\mathrm{e}} =& (B_{\mathrm{I}} P_{\mathrm{s}}^{-1} B_{\mathrm{I}}^{\top} + P_{\mathrm{s}}^{-1})^{-1} P_{\mathrm{s}}^{-1} \\ =& (B_{\mathrm{I}} B_{\mathrm{I}}^{\top} + \mathrm{I})^{-1} \\ J_{\mathrm{c}} =& (B_{\mathrm{I}} P_{\mathrm{s}}^{-1} B_{\mathrm{I}}^{\top} + 2P_{\mathrm{s}}^{-1})^{-1} P_{\mathrm{s}}^{-1} \\ =& (B_{\mathrm{I}} B_{\mathrm{I}}^{\top} + 2\mathrm{I})^{-1}. \end{aligned}$$

From [16], for at least one block row, i, each eigenvalue of J, $\lambda_k(J)$, satisfies

$$\left(\| \left(J_{i,i} - \lambda_k(J) \mathrm{I} \right)^{-1} \|_p \right)^{-1} \leq \sum_{i' \neq i} \| J_{i,i'} \|_p. \tag{3.16}$$

Being that all of the diagonal blocks, $J_{i,i}$ are zero,

$$\left(\| \left(J_{i,i} - \lambda_k(J)\mathrm{I}\right)^{-1} \|_p\right)^{-1} = \|\lambda_k(J)\mathrm{I}\|_p,$$

and (3.16) reduces to

$$\|\lambda_k(J)\mathrm{I}\|_p \leq \sum_{i' \neq i} \|J_{i,i'}\|_p.$$

Using the 2-norm in the previous inequality and taking the maximum sum over block rows,

$$\rho(J) \leq \sum_{i' \neq i} \|J_{i,i'}\|_2 = \max_i \sum_{i' \neq i} \rho(J_{i,i'}). \tag{3.17}$$

Equality in (3.17) comes from the fact that $\|J_{i,i'}\|$ is symmetric.

Careful inspection reveals that

$$B_\mathrm{I} B_\mathrm{I}^\top = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}.$$

It was shown in [48], that the eigenvalues of $B_{\mathrm{I}}B_{\mathrm{I}}^{\top}$ are $\lambda_m = 2 - 2\cos\left(\frac{(2m-1)\pi}{2\bar{\tau}+1}\right)$ for $m \in \{1, 2, \ldots, \bar{\tau}\}$. Hence the eigenvalues of $(B_{\mathrm{I}}B_{\mathrm{I}}^{\top} + \mathrm{I})$ are $\lambda_m = 3 - 2\cos\left(\frac{(2m-1)\pi}{2\bar{\tau}+1}\right)$ for $m \in \{1, 2, \ldots, \bar{\tau}\}$. The smallest eigenvalue of $(B_{\mathrm{I}}B_{\mathrm{I}}^{\top} + \mathrm{I})$ is the inverse of the spectral radius of $J_{\mathrm{e}}P_{\mathrm{s}}^{-1}$, denoted $\rho_{\mathrm{e}} = \frac{1}{3-2\cos\left(\frac{\pi}{2\bar{\tau}+1}\right)}$. In a similar manner, it can be shown that the spectral radius of $J_{\mathrm{c}}P_{\mathrm{s}}^{-1}$ is $\rho_{\mathrm{c}} = \frac{1}{4-2\cos\left(\frac{\pi}{2\bar{\tau}+1}\right)}$.

Each block row of J features either one $J_{\mathrm{c}}P_{\mathrm{s}}^{-1}$ block or two $J_{\mathrm{e}}P_{\mathrm{s}}^{-1}$ blocks. By inspection, $\rho_{\mathrm{e}} < 2\rho_{\mathrm{c}}$, hence $\rho(J) \leq \frac{2}{4-2\cos(\frac{\pi}{2\bar{\tau}+1})}$. ■

There are a few things to note about this bound. First, the bound gets tighter as η gets large. Second, the bound is independent of the number of agents. Last, because $\frac{2}{4-2\cos(\frac{\pi}{2\bar{\tau}+1})} < 1$, the Jacobi algorithm always converges. Last, it is independent of the number of agents.

To illustrate the tightness of the bound, the following table shows the error of the upper bound for several values of $\bar{\tau}$ and η.

Table 3.1: Error of Upper Bounds Computed for $\rho(J)$

		$\bar{\tau}$			
		1	2	3	4
η	2	0	0	0	0
	3	.0893	.0601	.0384	.0257
	4	.0591	.0399	.0256	.0171
	5	.0431	.0295	.0190	.0128

3.5.2 Trade-off of Computation and Communication

Section 3.3.2 discussed how the MAI algorithm can be used to estimate the state of distributed dynamic systems. There are two main parameters affecting the performance of this algorithm, the amount of history retained for estimation, κ, and the number of iterations at each time step, $\bar{\tau}$. Here, performance refers to the covariance of estimate error. Higher performance relates to lower error covariance (i.e. estimates that are "tighter" to the optimal estimates). Both κ and $\bar{\tau}$ are positively correlated with performance; making these two values larger improves performance.

The first parameter, κ, is the amount of history considered when performing distributed estimation. When κ is finite it is no longer possible to obtain the global BLUE (the estimate that considers every measurement the network has ever produced). This is due fixing the oldest estimates in the network. As a set of nodes become the oldest nodes in the graph, they are deemed to be reference nodes and their corresponding position estimates are taken as truth. By doing this, whatever error remains in the estimates is fixed forever. It will be demonstrated that even for κ relatively small, the method produces estimates that are near optimal and generally much better than dead reckoning. Furthermore, increasing κ produces diminishing returns. When increasing κ by a fixed amount, there is a larger performance increase when κ is small.

As an iterative algorithm, if the MAI algorithm is convergent then each iteration reduces the estimate error. As the number of iterations grows, the estimates converge asymptotically to the BLUE. As such, increasing $\bar{\tau}$ will improve the estimate's performance. As with κ, this effect has diminishing returns. When increasing $\bar{\tau}$ by a fixed amount, there is a larger performance increase when $\bar{\tau}$ is small. Also like κ, $\bar{\tau}$ is limited by constraints within the network. There is a finite limit to the number of iterations that are possible (i.e. much information can be shared) in a single time step. The density of the network and the bandwidth of the network's communication channels greatly affect how much information can be shared at each time step.

A logical question to ask when implementing the MAI algorithm on a dynamic system is just how well the algorithm will perform. This is not an easy question to answer as it depends not only on κ and $\bar{\tau}$, but on the network's topology and the measurement and model fidelities as well. A lower bound for the estimate error covariance is the estimate error covariance of the BLUE. Note that the BLUE is equivalent to the estimate one obtains from the MAI algorithm if $\kappa = \infty$ and $\bar{\tau} = \infty$. One can bound upper bound the estimate error covariance with the covariance of dead-reckoning. For a given agent, the worst case scenario would be for each agent to estimate their positions solely from inertial measurements. Under the assumption that the inertial measurements have independent

error, one would determine an agent's current estimate error covariance by summing the covariances of inertial measurements from the agent's reference node to it's current node. Note that dead reckoning is equivalent to a MAI algorithm where $\bar{\tau} = 0$. In [5] Barooah and Hespanha showed that adding a measurement to the network where there was not one previously can only reduce the error covariance of the BLUE. That is to say, by including inter-agent measurements in the estimation scheme can only serve to improve the resulting position estimates.

Due to the topological dependence of the MAI algorithm, no general closed form solution for the estimate covariance can be provided. However, given a specific network it is possible to explicitly compute the estimate error covariances. To illustrate the utility of the MAI algorithm and also the trade-off between κ and $\bar{\tau}$, covariance results that were calculated for the scenario where 10 agents are moving together are presented. Initially, each agent knows it's position exactly. As the agents move along, they make relative position measurements to their nearest neighbors. As shown in Figure 3.4, the measurements graph evolves as a rectangular grid. The agents measurements (both inertial and inter-agent) are corrupted with error of unity covariance.

Figure 3.5 shows the trade-off between κ and $\bar{\tau}$. This figure depicts the agents' average error covariance norm of current position estimates after 50 time steps. With only one iteration per time step and only keeping two time steps in

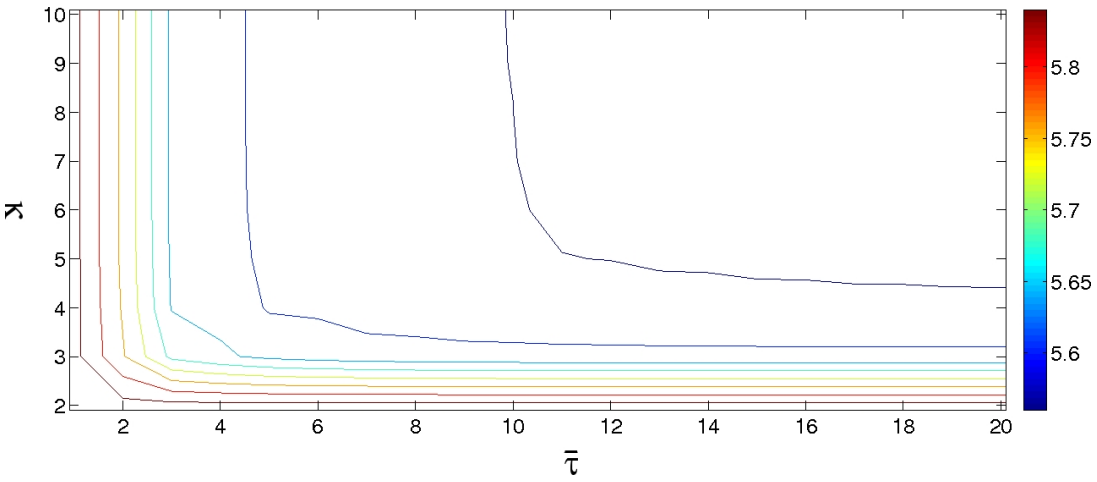


Figure 3.5: Average of the norm of agents' real time estimate errors at $k = 50$. These estimates use only past measurements. The level sets show the trade-offs for different values of κ and $\bar{\tau}$.

memory (the minimal parameters for the MAI algorithm to function), the agents' average error covariance norm is 5.86. For comparison, in an uncooperative environment (i.e. agents estimate positions using dead reckoning) the average error covariance norm would by 50. The same metric for BLUE estimation (i.e. a Kalman filter) is 5.55. Under very minimal assumptions, the algorithm provides estimates with performance close to that of Kalman filtering. If the algorithm parameters are increased so that $\kappa = 5$ and $\bar{\tau} = 5$, then the average error covariance norm shrinks to 5.59 a mere 0.7% higher than the that of Kalman filtering. Computationally, with these parameters the worst case is that an agent must solve 5 optimizations with 12 equations and 4 unknowns at each time step.

Figure 3.6 shows how the MAI algorithm performs as a smoother. This figure shows the agents' average error covariance norm for estimates at $k = 40$ using information gathered upto $k = 50$. These estimates approximate a fixed interval smoother. With only one iteration per time step and only keeping two time steps in memory, the agents' average error covariance norm is 4.85. For comparison, in an uncooperative environment the average error covariance norm would by 40. The same metric for BLUE estimation is 4.33. If the algorithm parameters are increased so that $\kappa = 5$ and $\bar{\tau} = 5$, then the average error covariance norm shrinks to 4.40 a mere 1.5% higher than the that of Kalman smoothing. Although these plots are for a particular scenario, the utility of

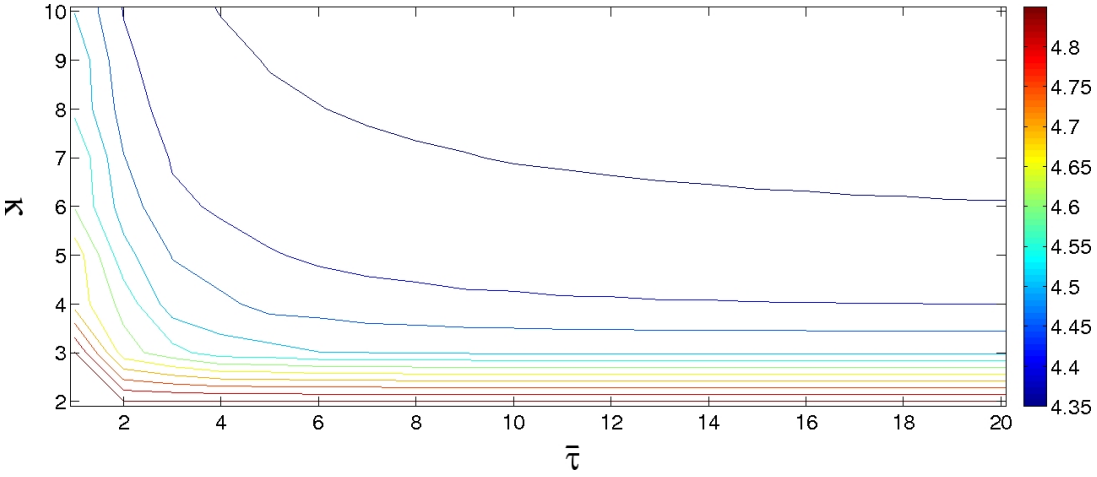


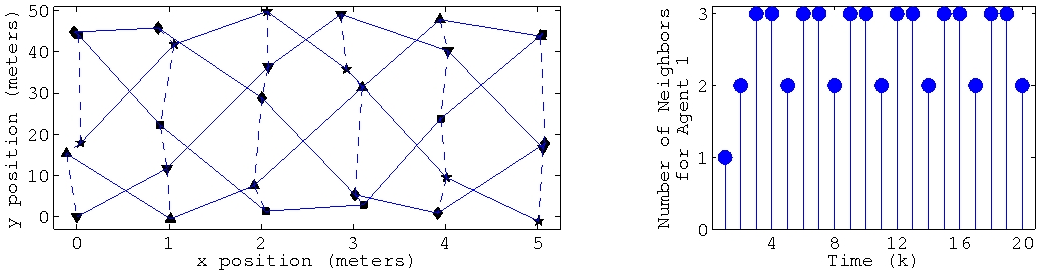
Figure 3.6: Average of the norm of agents' smoothed estimate errors at $k = 40$. These estimates use a window of measurements, both past and present to improve estimate performance. The level sets show the trade-offs for different values of κ and $\bar{\tau}$.

the MAI algorithm is apparent. Under very reasonable conditions, the method performs quite well.

3.6 Simulations

The MAI algorithm's performance is further illustrated by numerical simulations. All simulation results are shown for the case $\kappa = 1$. Five agents move in a zig-zag trajectory; the resulting measurement graph is shown in Figure 3.7(a). A time trace of the number of neighbors of Agent 1 is shown in Figure 3.7(b). The initial position of Agent 1 (bottom left node in Figure 3.7) is taken as the reference. Every measurement of $x_u - x_v$ is obtained from noisy measurements of the distance $\|x_u - x_v\|$ and the angle between x_u and x_v. The distance and angle measurements are corrupted with additive Gaussian noise, with $\sigma_r = 0.05m$ and $\sigma_\theta = 5^o$. The measurement error covariances are estimated from the range and bearing measurements and the parameters σ_r, σ_θ (as explained in [4]), which makes the covariances of the errors on relative position measurements on distinct edges distinct.

Covariances of agent position estimates produced by the MAI algorithm are estimated from 1000 Monte-Carlo runs. Figure 3.8 shows the covariance of the estimate of $x_5(k)$, the position of Agent 5, as a function of k. Agent 5 is the



(a) (b)

Figure 3.7: A snapshot of the measurement graph $\mathcal{G}(k)$ at time $k = 5$ created by the motion of 5 mobile agents, for which the simulations reported here are conducted.

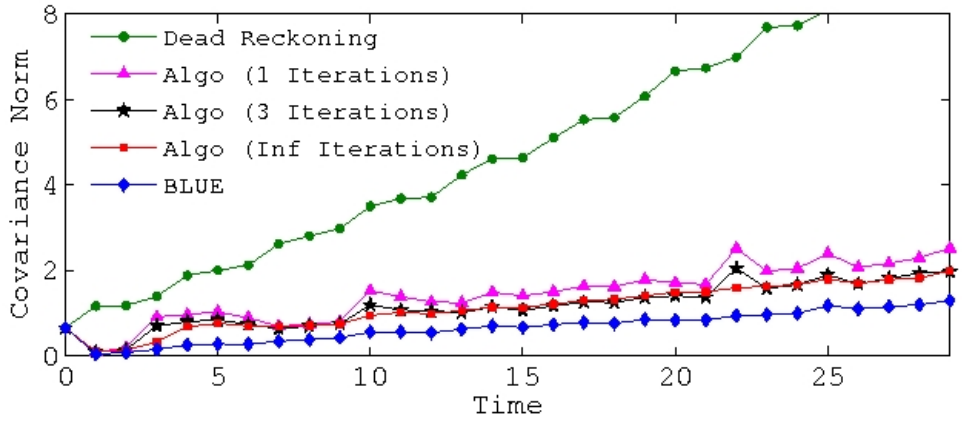


Figure 3.8: Covariance of the estimate of the current position of Agent 5 (of Figure 3.7) as a function of time. Agent 5 is the one farthest from Agent 1, whose initial position being the reference node. Dead reckoning provides an estimate of positions by summing optometry data. Estimates from the MAI algorithm are shown for $\bar{\tau} = 1, 3$, and ∞. BLUE refers to the centralized optimal.

one farthest away from Agent 1. The figure shows that the location estimates produced by the MAI algorithm are much more accurate than those produced by integrating the displacement measurements alone (dead reckoning). It is seen from the plot that the estimation error covariance of the algorithm (even with

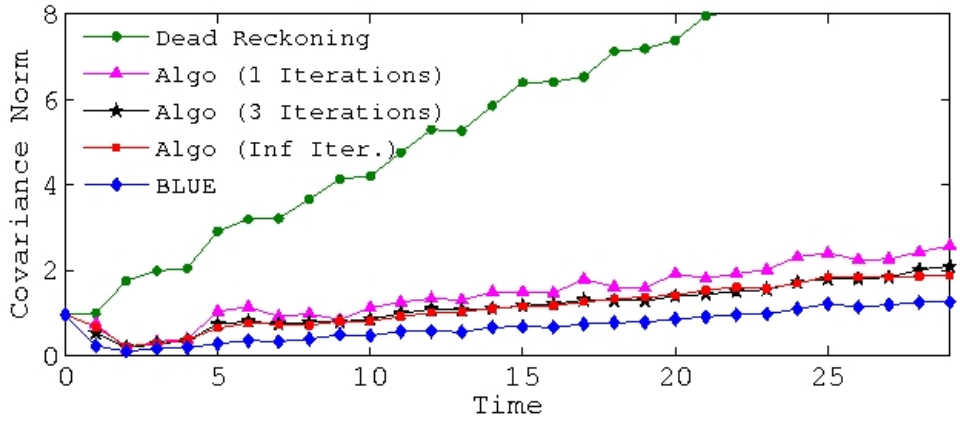


Figure 3.9: Covariance of the estimate of the current position of Agent 4 (of Figure 3.7) as a function of time.

$\kappa = 1$ and $\bar{\tau} = 1$) is close to that of the centralized optimal estimator (BLUE). Comparison among the plots for $\bar{\tau} = 1, 3$ and ∞ shows that, as expected, the estimation accuracy improves with increasing number of iterations between every time step. However, it is also seen that the improvement levels off quickly. In fact, even with the minimal possible number of iterations $\bar{\tau} = 1$, the estimation accuracy is quite close to the best possible (with $\bar{\tau} = \infty$). This property of the algorithm enhances its applicability since good estimates are obtained with little delay. Figure 3.9 plots these variables for the second agent.

3.7 Conclusion

This chapter presented a distributed algorithm for mobile agents to estimate their positions by fusing their own displacement measurements with inter-agent relative position measurements. The algorithm is distributed in the sense each agent can estimate its own position by communicating with nearby agents. The algorithm computes an approximation of the centralized optimal (Kalman smoother) estimates. The problem of distributed Kalman smoothing for this application is reformulated as a problem of computing the BLUE estimates. The graph structure of the BLUE estimation problem, opening the door for distributed estimation algorithms. With finite memory and limited number of iterations before new measurements are obtained, the algorithm produces an approximation of the Kalman smoother estimates. As the memory of each agent and the number of iterations between each time step are increased, the approximation improves.

This method was extended to include more general partitioning of the graph's node set. It was shown that by using a more general partitioning, the distributed estimation method can be extended to target tracking under a modest observability assumption. A closed form bound for convergence of the algorithm on a lattice with no constraint on memory was presented. Also, the trade off

between the number of iterations per time step and the amount of history retained during estimation was illustrated. It was shown that under very modest parameters, the algorithm vastly improves estimates when compared to dead reckoning. Furthermore, if the algorithm parameters are slightly improved performance of the algorithm nears that of Kalman Smoothing. Finally, simulations that illustrate the algorithm's performance under very restrictive conditions were presented. With a memory encompassing a single point in history, the algorithm greatly outperforms dead-reckoning.

Chapter 4

Optimal Estimation on the Graph Cycle Space

This chapter proposes a solution to the distributed estimation from relative measurements problem discussed in Chapter 2 and original explored in [4, 26, 37]. The relative difference measurement problem assumes that each sensor node (or agent) in a network has an associated state that cannot be measured absolutely (e.g. an agent's global position), but a agent can measure the relative difference between its state and its neighbor's states (e.g. an agent can measure the range and bearing to its neighbors). Barooah et al. explored several distributed algorithms for this problem that converge to the optimal minimum error variance solution [4]. Barooah's algorithms assume that agents have limited topological knowledge and communication abilities. In these algorithms, each agent computes an estimate of its own state synchronously using an affine transformation of neighboring state estimates. Methods that estimate the agent state directly

are referred to as *node estimation* methods. The node estimation methods introduced in [4] were shown to converge to the optimal estimate and were shown to be robust to interruptions in communication.

The work reported in the present chapter is motivated by the observation that, in the absence of noise, the sum of the relative difference measurements around any cycle of the graph should be zero. However, due to measurement noise, these sums are generally not equal to zero — these sums are referred to as discrepancies. This naturally leads to the desire to find estimates with no discrepancy that will sum to zero around the cycles of the graph. The graph cycle space has been used in the past to solve several problems. Notably, Kirchoff proposed that the redundancies present in cycles of electrical circuits could be used to determine voltages within the circuits [22]. More recently, in [41], Rockafellar discusses many properties of graph cycles and shows how the cycle space can be used to determine flows in a network. Piovan et al. explored the use of cycles in orientation localization from relative angle-of-arrival measurements in [38].

The Cycle Space Estimation algorithms introduced in this chapter consist of two steps: first, one computes a "corrected" version of the measurement set that has zero discrepancy and then one uses this corrected measurement set to estimate the state variables. The first step can be done either in a centralized or distributed fashion. Its decentralized version utilizes a Jacobi-like algorithm [35]

on a "cycles graph" whose nodes correspond to the cycles of the original graph. The second step can also be done in a centralized or distributed fashion. In the latter case, convergence is guaranteed within a finite number of steps (equal to the graph diameter) using the so-called flagged initialization algorithm [4].

For triangular, square, and hexagonal lattice graphs and certain classes of random graphs, the new distributed algorithms converge significantly faster than previous algorithms [4]. Our results show that, for large graphs, the Cycle Space Estimation algorithm provides significant gains in terms of the number of messages that are required to compute optimal estimates. This happens because the cycles graph often has much better convergence properties than the original graph. Monte Carlo simulations are used to show that this is the case for several classes of regular graphs, which include square, triangular, and hexagonal lattices of varying size. For square lattices, there are also explicit formulas to compute the convergence rate of the distributed algorithm.

This chapter expands upon results previously presented in [44]. Specifically, this chapter presents new convergence results for the distributed algorithm presented herein as well as expanded simulation results comparing this method to a preexisting distributed estimation algorithm.

This chapter proceeds in the following manner. In Section 4.1, the Centralized Optimal Cycle Space Estimate is introduced. The Distributed Optimal

Cycle Space Estimation algorithm is covered in Section 4.2. The convergence properties of the Distributed Optimal Cycle Space Estimation algorithm are discussed in Section 4.3. Simulation results are given in Section 4.4. Conclusions and future research goals are discussed in Section 4.5.

4.1 Centralized Cycle Space Estimation

To gain insight into the approach followed here, it is convenient to regard the states of the agents as voltages in an electric circuit that takes the shape of the measurement graph. The measurements would then correspond to tensions measured across the edges of the graph. When the graph has cycles, one should obtain zero by summing the tensions along a cycle (a la Kirchhoff's Voltage Law), but because of measurement errors this is generally not the case. In this light, one can view the estimation procedure as computing a set of voltages that are internally consistent, in the sense that they add up to zero around any cycle (i.e. the set satisfies KVL).

Let us define the *optimal tension estimate vector* as:

$$\hat{\mathbf{z}}^* \equiv \begin{bmatrix} B_{\mathcal{R}}^{\mathsf{T}} & B^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}} \\ \\ \hat{\mathbf{x}}^* \end{bmatrix}, \tag{4.1}$$

where $\hat{\mathbf{x}}^*$ is the minimum variance estimate for $\mathbf{x}$ from (3.14). The vector $\hat{\mathbf{z}}^*$ defined above is obtained by taking differences between all elements of the optimal estimate $\hat{\mathbf{x}}^*$ for which there are measurements available. That is, $\hat{\mathbf{z}}^*$ is the optimal estimate of differences in $\mathbf{x}$. In the absence of noise, $\hat{\mathbf{x}}^*$ would be equal to $\mathbf{x}$ and $\hat{\mathbf{z}}^*$ would be equal to the measurement vector, $\tilde{\mathbf{z}}$, which in turn is equal to the non-noisy state differences, $B^\mathsf{T}\mathbf{x}$.

While it appears from (4.1) that one needs to first compute the optimal estimate $\hat{\mathbf{x}}^*$ to compute the optimal tension estimate $\hat{\mathbf{z}}^*$, it will be demonstrated shortly that it is possible to compute $\hat{\mathbf{z}}^*$ directly from the measurements $\tilde{\mathbf{z}}$, and that it is then straightforward to obtain $\hat{\mathbf{x}}^*$ from $\hat{\mathbf{z}}^*$. It will also be shown that this indirect procedure can be advantageous from the perspective of reducing communication and computation when implemented as a distributed algorithm.

The remainder of this section is split into three parts. The first subsection introduces the cycle space of a graph. The next subsection details how the centralized optimal tension estimate is computed. The last subsection provides a method for computing the BLUE from the optimal tension estimate.

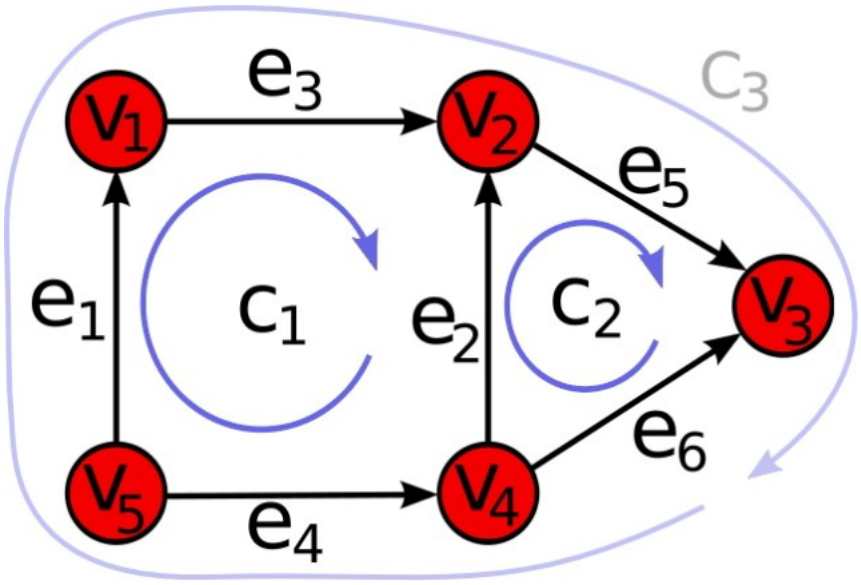
4.1.1 Cycle Space

A *simple cycle* of $\mathcal{G}$ is a sequence of alternating nodes and edges that starts and ends at the same node and has only this one repeated node. The direction

of a cycle is given innately by the order in which the nodes of the cycle are visited. A simple cycle need not follow the direction of its edges; a cycle may traverse an edge forward or backward. By indexing the edges with the natural from numbers 1 to m, a *simple cycle vector* can be assigned to each simple cycle, which is defined to be the m-length vector $\mathbf{c} = (c_1, \ldots, c_m)^\mathsf{T}$, where,

$$c_j \equiv \begin{cases} 1 & \text{if cycle traverses edge j forward} \\ -1 & \text{if cycle traverses edge j backward} \\ 0 & \text{otherwise.} \end{cases}$$

The *cycle space*, $\mathcal{C}$, of $\mathcal{G}$ is then defined to be the subspace of $\mathbb{R}^m$ generated by the simple cycle vectors associated with the simple cycles in $\mathcal{G}$. Subsequently, C will be used to denote the Kronecker product of a matrix whose columns are simple cycle vectors that form a basis for the cycle space and the identity matrix of size d, the dimension of the measurements. Considering that every column of $B_\mathcal{G}$ corresponds to an edge in $\mathcal{G}$, C is a basis for the nullspace of $B_\mathcal{G}$ [41]. The cycle space matrix, C, has height dm and width dr, where r is the dimension of $\mathcal{C}$. For an example cycle space matrix, see Figure 4.1.



$$C = \begin{bmatrix} 1 & -1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -1 \end{bmatrix}^{\mathsf{T}}$$

Figure 4.1: An example graph shown along with a cycle space matrix, C. Note that $\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2$ is a cycle that is not in the basis, C.

4.1.2 The Centralized Optimal Tension Estimate

Optimal tension estimation uses $\tilde{\mathbf{z}}$ and C to compute $\hat{\mathbf{z}}^*$. The premise of optimal tension estimation is that one can add a correction term to the measurement vector, $\tilde{\mathbf{z}}$, to form the optimal tension estimate vector, $\hat{\mathbf{z}}^*$. This idea is formalized in the following result, which is a main contribution in this work.

Theorem 5 *[The Optimal Tension Estimate]*

The optimal tension estimate from (4.1)*,*

$$\hat{\boldsymbol{z}}^* = \begin{bmatrix} B_{\mathcal{R}}^T & B^T \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\mathcal{R}} \\ \\ \hat{\boldsymbol{x}}^* \end{bmatrix},$$

is the solution to the quadratic programming problem:

$$\hat{\boldsymbol{z}}^* = \underset{\hat{\boldsymbol{z}}}{\operatorname{argmin}}[(\tilde{\boldsymbol{z}} - \hat{\boldsymbol{z}})^T P^{-1} (\tilde{\boldsymbol{z}} - \hat{\boldsymbol{z}})] \qquad \text{s.t. } C^T \hat{\boldsymbol{z}} = 0 \tag{4.2}$$

$$= \begin{cases} \tilde{\boldsymbol{z}} - PC(C^T PC)^{-1} C^T \tilde{\boldsymbol{z}} & \text{if } r \neq 0 \\ \\ \tilde{\boldsymbol{z}} & \text{if } r = 0. \end{cases} \tag{4.3}$$

Proof: When $r = 0$ ($\mathcal{G}$ has no cycles), C^{T} is trivial and the constraint in (4.2) is satisfied for any $\hat{\mathbf{z}}$. Consequentially, $\hat{\mathbf{z}}^* = \tilde{\mathbf{z}}$. Furthermore, with no cycles in the graph, the BLUE of a particular agent is simply the summation of measurements along the unique path from the reference agent to the agent in question. As such, the difference in the BLUEs of two neighboring agents is simply the measurement connecting these agents. Hence, the optimal tension vector for which (4.1) holds is the measurement vector, $\tilde{\mathbf{z}}$.

When $r \neq 0$, the Lagrangian for (4.2) is:

$$L(\hat{\mathbf{z}}, \mathbf{\Lambda}) = (\tilde{\mathbf{z}} - \hat{\mathbf{z}})^{\mathsf{T}} P^{-1} (\tilde{\mathbf{z}} - \hat{\mathbf{z}}) + \mathbf{\Lambda}^{\mathsf{T}} C^{\mathsf{T}} \hat{\mathbf{z}}.$$

Taking the partial derivatives of the Lagrangian equal to zero:

$$\begin{aligned} \frac{\partial L}{\partial \hat{\mathbf{z}}} &= 2P^{-1}(\hat{\mathbf{z}} - \tilde{\mathbf{z}}) + C\mathbf{\Lambda} = 0 \\ \frac{\partial L}{\partial \mathbf{\Lambda}} &= C^{\mathsf{T}} \hat{\mathbf{z}} = 0, \end{aligned}$$

which can be written in the form:

$$\begin{bmatrix} 2P^{-1} & C \\ C^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{z}} \\ \mathbf{\Lambda} \end{bmatrix} = \begin{bmatrix} 2P^{-1}\tilde{\mathbf{z}} \\ 0 \end{bmatrix}. \tag{4.4}$$

Using the Schur Complement (see [28]), (4.4) can be solved to find $\hat{\mathbf{z}}^*$:

$$\begin{aligned} \hat{\mathbf{z}}^* &= \left[\tfrac{1}{2} P - \tfrac{1}{2} PC(C^{\mathsf{T}} PC)^{-1} C^{\mathsf{T}} P \right] 2P^{-1} \tilde{\mathbf{z}} \\ &= [I - PC(C^{\mathsf{T}} PC)^{-1} C^{\mathsf{T}}] \tilde{\mathbf{z}}. \end{aligned} \tag{4.5}$$

Lemma 2 (in the Appendix) is used to show that (4.5) is related to the BLUE by (4.1). Lemma 2 states that for two full column rank matrices, α and β^{T}, the

image of α is the nullspace of β if and only if

$$\alpha(\alpha^{\mathsf{T}}\alpha)^{-1}\alpha^{\mathsf{T}} = I - \beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta.$$

Consider the matrices $\beta = \text{image}(BP^{-\frac{1}{2}})$ and $\alpha = \text{image}(P^{\frac{1}{2}}C)$. Note that $\text{null}(\beta) = \text{image}(\alpha)$. Considering Lemma 2 and the fact that $C^{\mathsf{T}}B_{\mathcal{R}}^{\mathsf{T}} = 0$, the following equalities hold:

$$\begin{aligned}
0 &= [\alpha(\alpha^{\mathsf{T}}\alpha)^{-1}C^{\mathsf{T}}P^{\frac{1}{2}}]P^{-\frac{1}{2}}B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} \\
&= [\alpha(\alpha^{\mathsf{T}}\alpha)^{-1}\alpha^{\mathsf{T}}]P^{-\frac{1}{2}}B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} \\
&= [I - \beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta]P^{-\frac{1}{2}}B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} \\
&= P^{\frac{1}{2}}[I - \beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta]P^{-\frac{1}{2}}B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} \\
&= [I - P^{\frac{1}{2}}\beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta P^{-\frac{1}{2}}]B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} \\
&= [I - B^{\mathsf{T}}(BP^{-1}B^{\mathsf{T}})^{-1}BP^{-1}]B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}}. \qquad (4.6)
\end{aligned}$$

Consider again Lemma 2:

$$
\begin{aligned}
I - \alpha(\alpha^{\mathsf{T}}\alpha)^{-1}\alpha^{\mathsf{T}} &= \beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta \\
I - P^{\frac{1}{2}}\alpha(\alpha^{\mathsf{T}}\alpha)^{-1}\alpha^{\mathsf{T}}P^{-\frac{1}{2}} &= P^{\frac{1}{2}}\beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta P^{-\frac{1}{2}} \\
I - PC(C^{\mathsf{T}}PC)^{-1}C^{\mathsf{T}} &= B^{\mathsf{T}}(BP^{-1}B^{\mathsf{T}})^{-1}BP^{-1} \\
[I - PC(C^{\mathsf{T}}PC)^{-1}C^{\mathsf{T}}]\tilde{\mathbf{z}} &= B^{\mathsf{T}}(BP^{-1}B^{\mathsf{T}})^{-1}BP^{-1}\tilde{\mathbf{z}}. \qquad (4.7)
\end{aligned}
$$

Now, add (4.6) to (4.7):

$$
\begin{aligned}
[I - PC(C^{\mathsf{T}}PC)^{-1}C^{\mathsf{T}}]\tilde{\mathbf{z}} =& B^{\mathsf{T}}(BP^{-1}B^{\mathsf{T}})^{-1}BP^{-1}\tilde{\mathbf{z}} \\
&+ [I - B^{\mathsf{T}}(BP^{-1}B^{\mathsf{T}})^{-1}BP^{-1}]B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} \\
=& B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} + B^{\mathsf{T}}(BP^{-1}B^{\mathsf{T}})^{-1}BP^{-1}(\tilde{\mathbf{z}} - B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}}) \\
=& B_{\mathcal{R}}^{\mathsf{T}}\mathbf{x}_{\mathcal{R}} + B^{\mathsf{T}}\hat{\mathbf{x}}^{*} \equiv \hat{\mathbf{z}}^{*}.
\end{aligned}
$$

■

In light of Theorem 5 the following interpretation of $\hat{\mathbf{z}}^{*}$ can be made: The tension vector corresponding to the BLUE, $\hat{\mathbf{x}}^{*}$, is the tensions vector with the least square difference from the measurement vector, $\tilde{\mathbf{z}}$. That is to say, it is the tension vector with minimum squared edge modification.

4.1.3 The Centralized Optimal State Estimate

As expressed in (4.1), the optimal tension estimate, $\hat{\mathbf{z}}^*$, is a vector of the relative difference in BLUEs, $\hat{\mathbf{x}}^*$. Because the optimal tension estimates are consistent (i.e. they sum to zero around cycles), the sum of optimal tension estimates between any two points in the graph is path independent. As such, the optimal estimate of any agent's state can be found by summing the optimal tension estimates along a path from a reference agent to the agent.

To represent a path between Agent i and Agent i', define a *path vector* to be the m-length row-vector $\mathbf{h}_{i,i'} = (h_1, \ldots, h_m)^\mathsf{T}$, where,

$$h_j \equiv \begin{cases} 1 & \text{if the path traverses edge j forward} \\ -1 & \text{if the path traverses edge j backward} \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{T} \in \mathcal{G}$ be the subgraph corresponding a minimal spanning forest that consists of trees rooted at the nodes corresponding to reference agents. That is, $\mathcal{T}$ is a graph where all nodes are weakly connected (i.e. connected without respect to edge direction) to the reference node and the number of edges in the graph is minimized [50]. The *path vector matrix* of graph $\mathcal{T}$ is then defined to be an n by

m matrix where the i^{th} row is a path vector from the reference agent to Agent i:

$$H_{\mathcal{T}} \equiv \begin{bmatrix} \mathbf{h}_{1,\mathcal{R}} \\ \vdots \\ \mathbf{h}_{n,\mathcal{R}} \end{bmatrix}.$$

Finally, $\hat{\mathbf{x}^*}$ can be computed from $\hat{\mathbf{z}}^*$ using,

$$\hat{\mathbf{x}^*} = \begin{bmatrix} \mathbf{x}_{\mathcal{R}} \\ \vdots \\ \mathbf{x}_{\mathcal{R}} \end{bmatrix} + H_{\mathcal{T}} \hat{\mathbf{z}}^*.$$

The preceding equation relates that the optimal estimate of an agent's state is simply the state of the reference agent plus the sum of optimal tension estimates along a path connecting the agent to the reference agent.

4.2 Distributed Cycle Space Estimation

In a large network it is often impractical to collect all of the measurements at one agent, compute the optimal estimate at that agent, and then redistribute the estimate to every other agent. Instead, it is desirable to find an estimate of the state in a distributed manner.

This section presents the Jacobi Cycle Space Estimation (JCSE) algorithm, a distributed algorithm that uses the cycles of the graph to solve the estimation from relative measurements problem. JCSE uses two levels of computation in series to estimate the graph states, $\mathbf{x}$. The first level in the JCSE algorithm iteratively computes a tension estimate, $\hat{\mathbf{z}}_j$, for all edges of the graph, $\mathrm{e}_j \in \mathcal{E}$. The second level in the JCSE algorithm uses the tension estimates from the first level to compute state estimates, $\hat{\mathbf{x}}_i$, for all nodes in the graph, $\mathrm{v}_i \in \mathcal{V}$.

In the first level of the JCSE algorithm, each edge, e_j, has associated with it a tension estimate, $\hat{\mathbf{z}}_j$, that is initialized $\tilde{\mathbf{z}}_j$, the measurement corresponding to the edge j. Also, each simple cycle, $\mathbf{c}_k \in \mathcal{C}$, has a d-length vector iteration variable, $\hat{\mathbf{y}}_k$. At each iteration $\hat{\mathbf{y}}_k$ is computed to be the value that, when added to the tension estimates of edges in $\mathbf{c}_k$, will make these tension estimates consistent. An edge may exist in multiple cycles (causing the cycles to be adjacent) and these cycles generally will not agree on what value must be added to the respective measurement to make it consistent. Because of this, the cycles iteratively refine their iteration variables to account for adjacent cycles' iteration variables. While this iteration is proceeding, the tension estimate for an edge is continually recomputed to be a weighted sum of iteration variables for cycles that the edge is a part of added to the measurement associated with the the edge.

The second level of the JCSE algorithm uses the tension estimates, $\hat{\mathbf{z}}_j$, to compute state estimates, $\hat{\mathbf{x}}_i$. This comes about naturally because $\hat{\mathbf{z}}_j$ are estimates of differences in two states. As such, if the tension estimates along a path from a node, v_i, to any reference node are available, then the sum of these tension estimates provides $\hat{\mathbf{x}}_i$, the estimate of the state of v_i.

To improve readability, the remainder of this section is divided into three parts. First several structures including the weighted cycle degree matrix, the weight cycle adjacency matrix, and the cycle Laplacian are defined. Of these matrices, the first two are used in the JCSE algorithm and the last is used to illustrate how the first two matrices relate to the cycle space matrix, C. The second subsection presents a distributed tension estimation algorithm. The last subsection presents a distributed algorithm that estimates the agent states from the distributed tension estimates.

4.2.1 Cycle Laplacian

Motivated by (4.3) *cycle Laplacian matrix* is defined as,

$$\mathcal{L}_{\mathrm{c}} \equiv C^{\mathsf{T}} P C. \tag{4.8}$$

While $\mathcal{L}_{\mathrm{c}}$ is generally not doubly stochastic, it is called the Laplacian because it characterizes the weighted cycle degree and the weighted cycle adjacency, which are defined as follows: let $\mathcal{B}_k \subset \mathcal{E}$ be the set of edges in $\mathbf{c}_k \in \mathcal{C}$, the cycle corresponding to the k^{th} column of the cycle space basis matrix C. Define the *weighted cycle degree matrix*, D_k, of this cycle as the sum of the error covariance matrices, P_j as defined in (2.4), of measurements corresponding to edges $j \in \mathcal{B}_k$, i.e.

$$D_k \equiv \sum_{j \in \mathcal{B}_k} P_j.$$

Define the *weighted cycle adjacency matrix*, $A_{k,k'}$ between two cycles, $\mathbf{c}_k$ and $\mathbf{c}'_k$, as the net sum of error covariance matrices, P_j, corresponding to measurements that exist in both cycles, i.e. $j \in \mathcal{B}_k \cap \mathcal{B}_{k'}$. The cycle adjacency between two cycles increases if both cycles traverse the edge in the same direction and decreases if the cycles traverse the edge in opposite directions. The cycle adjacency between $\mathbf{c}_k$ and $\mathbf{c}'_k$ is

$$A_{k,k'} \equiv \sum_{j \in \mathcal{B}_k \cap \mathcal{B}_{k'}} P_j \mathbf{c}_{j,k} \mathbf{c}_{j,k'},$$

where $\mathbf{c}_{j,k}$ is the j^{th} element of $\mathbf{c}_k$ (the element in the j^{th} row and k^{th} column of the cycle space matrix, C). If two cycles have no common edges, then the weighted adjacency between the cycles is zero. The weighted cycle degree and

weighted cycle adjacency, so defined, are related to the cycle Laplacian by

$$\mathcal{L}_{\mathrm{c}} = D - A,$$

$$D = \mathrm{diag}(D_1, \ldots, D_{\mathcal{R}}),$$

$$A = [A_{k,k'}].$$

To solidify the idea of cycle degree, cycle adjacency, and cycle laplacian, consider the example graph shown in Figure 4.1. If all measurements in the graph have covariance, P_{m}, then,

$$D = \begin{bmatrix} 4P_{\mathrm{m}} & 0 \\ 0 & 3P_{\mathrm{m}} \end{bmatrix}, \; A = \begin{bmatrix} 0 & P_{\mathrm{m}} \\ P_{\mathrm{m}} & 0 \end{bmatrix}, \; L_{\mathrm{c}} = \begin{bmatrix} 4P_{\mathrm{m}} & -P_{\mathrm{m}} \\ -P_{\mathrm{m}} & 3P_{\mathrm{m}} \end{bmatrix}.$$

4.2.2 The Distributed Tension Estimate

Formally, the following assumptions are made for the JCSE algorithm:

1. An agent knows the measurements corresponding its edges.

2. Each agent knows how to send messages to each one of its neighbors in $\mathcal{G}$.

3. A basis C for the cycle space of the graph is known and a "leader" has been elected for each simple cycle corresponding to the columns of C.

4. All agents that belong to one (or more) of the simple cycles associated with a column of C know how to send measurements to the leaders of these cycles (perhaps through multi-hop communication).

JCSE starts with each cycle leader collecting the sum of measurements in its cycle. Each cycle leader then computes the total discrepancy in its cycle by adding the measurements of the cycle: $\boldsymbol{\Delta}_k = \sum_{j \in \mathcal{B}_k} \tilde{\mathbf{z}}_j$. Note that all of the discrepancies can be represented in one equation as $\boldsymbol{\Delta} = C^{\mathsf{T}} \tilde{\mathbf{z}}$.

Each cycle leader possesses an iteration variable $\hat{\mathbf{y}}_k$ that is initialized to zero. Let $\mathcal{A}_k$ be the set of cycles that share at least one edge with cycle k. Then, as the JCSE algorithm proceeds, at each step each cycle leader:

1. Sends the value of its iteration variable, $\hat{\mathbf{y}}_k$, to the leaders of the cycles that share edges with its own cycle, as well as to all the agents in its cycle,

2. Receives iteration variables, $\hat{\mathbf{y}}_{\mathcal{A}_k}$ from the leaders of neighboring cycles,

3. Recomputes its iteration variable,

$$\hat{\mathbf{y}}_k^{t+1} = D_k^{-1} \left(\sum_{k' \in \mathcal{A}_k} A_{k,k'} \, \hat{\mathbf{y}}_{k'}^t - \boldsymbol{\Delta}_k \right). \tag{4.9}$$

In addition to the iteration above carried out by the cycle leaders, each agent also maintains internal estimates of tensions for all its incident edges. In

particular, the estimate of the tension $\hat{\mathbf{z}}_j^t$ for edge j is computed as follows:

$$\hat{\mathbf{z}}_j^t = \tilde{\mathbf{z}}_j + P_j \sum_{k \in \mathcal{K}_j} \hat{\mathbf{y}}_k^t, \tag{4.10}$$

where $\tilde{\mathbf{z}}_j$ is the measurement corresponding to the edge j, P_j the error covariance for that edge, and $\mathcal{K}_j$ denotes the set of cycles that contain the edge j. The set $\mathcal{K}_j$ can be empty if the edge does not belong to any cycle, in which case $\hat{\mathbf{z}}_j^t = \tilde{\mathbf{z}}_j$.

4.2.3 The Distributed State Estimate

To obtain its current state estimate from the tension estimates, an agent needs only to obtain the sum of the tensions along a path between itself and a reference agent, which can be accomplished via a distributed flagging method similar to that introduced in [4]. In this way, all non-reference agents start with a flagged state value, $\hat{\mathbf{x}}^0 = \infty$. The first time that a node, say $\mathrm{v}_i \in \mathcal{V}$, communicates with a neighbor, say $\mathrm{v}_{i'} \in \mathcal{V}$, that has a non-flagged state, v_i is aware that $\mathrm{v}_{i'}$ is the last node in a path connected to a reference node. v_i will then compute its state estimate, $\hat{\mathbf{x}}_i^{t+1}$, to be $\mathrm{v}_{i'}$'s state, $\hat{\mathbf{x}}_{i'}^t$, plus or minus (depending on the orientation of the edge that connects the nodes, $\mathrm{e}_{i,i'} \in \mathcal{E}$) the tension estimate connecting the two nodes, $\hat{\mathbf{z}}_{i,i'}^t$. As the state estimates, $\hat{\mathbf{x}}^t$, and tension estimates, $\hat{\mathbf{z}}^t$, are refined, v_i will maintain its estimate as the current

state estimate for v'_i, $\hat{\mathbf{x}}^t_{i'}$, plus or minus the current tension estimate for the edge connecting the two nodes, $\hat{\mathbf{z}}^t_{i,i'}$. Mathematically, this process can be expressed as:

$$\hat{\mathbf{x}}^0_i \equiv \begin{cases} \infty & \text{if } \mathbf{x}_i \in \mathbf{x} \\ \text{reference} & \text{if } \mathbf{x}_i \in \mathbf{x}_{\mathcal{R}} \end{cases}$$

$$\hat{\mathbf{x}}^{t+1}_i \equiv \begin{cases} \infty & \text{if } \hat{\mathbf{x}}^t_{i'} = \infty \\ \hat{\mathbf{x}}^t_{i'} + \hat{\mathbf{z}}^t_{i,i'} & \text{if } \mathrm{e}_{i,i'} \text{ leaves } \mathrm{v}_i \ \& \ \hat{\mathbf{x}}^t_{i'} \neq \infty \\ \hat{\mathbf{x}}^t_{i'} - \hat{\mathbf{z}}^t_{i,i'} & \text{if } \mathrm{e}_{i,i'} \text{ enters } \mathrm{v}_i \ \& \ \hat{\mathbf{x}}^t_{i'} \neq \infty \end{cases} \tag{4.11}$$

Using this flagging method, a breadth first tree (or trees) will grow out from a reference agent (or reference agents) as shown in Figure 4.2. A breadth first tree is a tree in which each node in the graph is connected to the root node (i.e. the reference node) via the shortest path [50]. Under the assumption that the measurement graph is weakly connected and neighbors in the graph can communicate, the number of iterations for the flagged initialization process to complete and all agents to have a state estimate is less than or equal to the diameter of the graph.

For a given graph, there can be many trees rooted at the reference nodes. Choosing different trees will give different initial state estimates, $\hat{\mathbf{x}}$. However, as

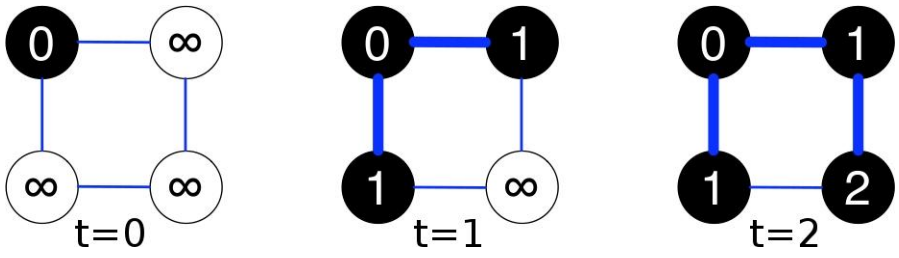


Figure 4.2: An example of the breadth first flagging method. In the first time instant only the reference agent has an estimate of its state. At $t = 1$ both of the reference agent's neighbors have estimates of their states. By $t = 2$, all agents have state estimates and a breadth first tree has been created for the graph (indicated by bold edges).

the tension estimates converge to the optimal estimate, $\hat{\mathbf{z}}^*$, the state estimates obtained using any tree will converge to the minimum variance estimate, $\hat{\mathbf{x}}^*$, because $\hat{\mathbf{z}}^*$ is consistent.

4.3 Convergence of the JCSE Algorithm

The convergence properties for the JCSE can be examined by considering the algorithm's effects on the graph as a whole. Start by stacking the iteration variables, $\hat{\mathbf{y}}_k$, into a single dr-length vector, $\hat{\mathbf{y}}$. The update law (4.9) can be rewritten:

$$\hat{\mathbf{y}}^{t+1} = D^{-1}\left(A\hat{\mathbf{y}}^t - \boldsymbol{\Delta}\right), \tag{4.12}$$

where D is the graph's weighted cycle degree matrix, A is the graph's weighted cycle adjacency matrix, and $\boldsymbol{\Delta}$ is the vector of cycle discrepancy. Similarly,

when the tension estimates are stacked into a single vector, $\hat{\mathbf{z}}$, the update law corresponding to (4.10) can be expressed as:

$$\hat{\mathbf{z}}^t = \tilde{\mathbf{z}} + PC\hat{\mathbf{y}}^t. \tag{4.13}$$

To represent the flagging process in a similar manner, define the *directed adjacency matrix* of graph $\mathcal{G}$ as $A_{\mathcal{G}} = [a_{ii'}]$, $i \in \{1, \ldots, n\}$, $i' \in \{1, \ldots, n\}$,

$$a_{ii'} \equiv \begin{cases} I_d & \text{if an edge leaves node } i' \text{ and enters node } i \\ 0 & \text{otherwise.} \end{cases}$$

Define the *in-incidence matrix* of graph $\mathcal{G}$ as $B^{\text{in}}_{\mathcal{G}} = \left[b^{\text{in}}_{ij}\right]$, $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$,

$$b^{\text{in}}_{ij} \equiv \begin{cases} -I_d & \text{if edge } j \text{ enters node } i \\ 0 & \text{otherwise.} \end{cases}$$

As previously discussed, the flagging procedure in (4.11) constructs a breadth first tree rooted at the reference node. Denote this tree as the subgraph $\mathcal{T} \in \mathcal{G}$, then the flagging process can be represented using the equation:

$$\boldsymbol{\chi}^{t+1} = A_{\mathcal{T}}\boldsymbol{\chi}^t + B_{\mathcal{T}}\hat{\mathbf{z}}^t, \tag{4.14}$$

where

$$\boldsymbol{\chi}^t = \begin{bmatrix} \mathbf{x}_{\mathcal{R}} \\ \hat{\mathbf{x}}^t \end{bmatrix},$$

and $A_{\mathcal{T}}$ and $B_{\mathcal{T}}$ are the directed adjacency and the in-incidence matrices of $\mathcal{T}$, respectively. Equations (4.12), (4.13), and (4.14) can be combined into the following linear equation representing the JCSE algorithm:

$$\begin{bmatrix} \hat{\mathbf{y}}^+ \\ \boldsymbol{\chi}^+ \end{bmatrix} = \begin{bmatrix} D^{-1}A & 0 \\ B_{\mathcal{T}}PC & A_{\mathcal{T}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{y}} \\ \boldsymbol{\chi} \end{bmatrix} - \begin{bmatrix} D^{-1}C^{\mathsf{T}} \\ -B_{\mathcal{T}} \end{bmatrix} \tilde{\mathbf{z}}. \tag{4.15}$$

With the JCSE algorithm in vectorized form, (4.15), the algorithm's convergence properties can be explored. The remainder of this section is divided into four parts. The first subsection provides a sufficient condition for convergence of the JCSE algorithm. In the second subsection, it is proved that any planar graph has a cycle space matrix such that the JCSE algorithm converges. Next, the convergence rate for the JCSE algorithm is given for square lattice graphs. The last subsection shows that certain cycle space matrices lead to convergence in finite time.

4.3.1 Condition for Convergence of the JCSE Algorithm

The following theorem provides a sufficient condition for convergence of the JCSE algorithm.

Theorem 6 *[Sufficient Condition for Convergence]*

Given D and A, the weighted cycle degree matrix and the weighted cycle adjacency matrix, respectively, if the spectral radius (i.e. the largest eigenvalue) of $D^{-1}A$ is strictly less than 1, then the JCSE algorithm converges, the tension estimate, $\hat{\boldsymbol{z}}^t$, converges to the optimal (weighted minimum mean squared error) tension estimate, $\hat{\boldsymbol{z}}^*$, and the state estimate, $\hat{\boldsymbol{x}}$, converges to the BLUE, $\hat{\boldsymbol{x}}^*$.

Proof: In view of (4.15), the JCSE algorithm is a cascade linear system; for stability, both sections of the cascade must be stable. All eigenvalues for the adjacency matrix for the directed tree, $A_{\mathcal{T}}$, are zero. This can be seen by enumerating nodes in the tree according to their proximity to reference nodes. The reference nodes in the graph have the lowest number, followed by neighbors of reference nodes, etc; the leafs on the tree are labeled with the the largest numbers. When this is done, $A_{\mathcal{T}}$ is strictly lower triangular and therefore has only zero eigenvalues. From this, the rate of convergence for (4.15) is bounded by the spectral radius, $\rho(J)$, of the iteration matrix, $J = D^{-1}A$.

From (4.12), when $\hat{\mathbf{y}}^t$ converges, $\hat{\mathbf{y}}^* = D^{-1}(A\hat{\mathbf{y}}^* - \boldsymbol{\Delta})$. The error between $\hat{\mathbf{y}}^t$ and $\hat{\mathbf{y}}^*$ has the following dynamics:

$$\begin{aligned}\hat{\mathbf{y}}^{t+1} - \hat{\mathbf{y}}^* &= D^{-1}(A\hat{\mathbf{y}}^t - \boldsymbol{\Delta}) - D^{-1}(A\hat{\mathbf{y}}^* - \boldsymbol{\Delta}) \\ &= D^{-1}A(\hat{\mathbf{y}}^t - \hat{\mathbf{y}}^*).\end{aligned}$$

When the spectral radius of $D^{-1}A$ is strictly less than 1, the error between $\hat{\mathbf{y}}^t$ and $\hat{\mathbf{y}}^*$ goes to zero as t goes to infinity. Similarly, from (4.13), when $\hat{\mathbf{z}}^t$ converges, $\hat{\mathbf{z}}^* = \tilde{\mathbf{z}} + PC\hat{\mathbf{y}}^*$. The error between $\hat{\mathbf{z}}^t$ and $\hat{\mathbf{z}}^*$ has the following dynamics:

$$\begin{aligned}\hat{\mathbf{z}}^{t+1} - \hat{\mathbf{z}}^* &= \tilde{\mathbf{z}} + PC\hat{\mathbf{y}}^t - \tilde{\mathbf{z}} + PC\hat{\mathbf{y}}^* \\ &= PC(\hat{\mathbf{y}}^t - \hat{\mathbf{y}}^*).\end{aligned}$$

If the spectral radius of $D^{-1}A$ is less than 1 and $\hat{\mathbf{y}}^t - \hat{\mathbf{y}}^*$ goes to zero as t goes to infinity, then $\hat{\mathbf{z}}^t - \hat{\mathbf{z}}^*$ will also go to zero as t goes to infinity. Furthermore,

when $\hat{\mathbf{y}}^t$ converges:

$$\begin{aligned}
\hat{\mathbf{y}}^* &= D^{-1}(A\hat{\mathbf{y}}^* - \boldsymbol{\Delta}) \\
D\hat{\mathbf{y}}^* &= A\hat{\mathbf{y}}^* - \boldsymbol{\Delta} \\
(D - A)\hat{\mathbf{y}}^* &= -\boldsymbol{\Delta} \\
(C^\mathsf{T} P C)\hat{\mathbf{y}}^* &= -C^\mathsf{T}\tilde{\mathbf{z}} \\
\hat{\mathbf{y}}^* &= -(C^\mathsf{T} P C)^{-1} C^\mathsf{T}\tilde{\mathbf{z}},
\end{aligned}$$

from which, $\hat{\mathbf{z}}^* = \tilde{\mathbf{z}} - PC(C^\mathsf{T} P C)^{-1} C^\mathsf{T}\tilde{\mathbf{z}}$. Hence the tension estimate, $\hat{\mathbf{z}}$ converges to the optimal tension estimate, (4.3).

Theorem 5, tells us that the optimal tension estimate vector, $\hat{\mathbf{z}}^*$, consists of the pairwise differences of the BLUE, $\hat{\mathbf{x}_i}^*$, corresponding to edges in the graph. As such, summing $\hat{\mathbf{z}}^*$ along the directed tree provided by $A_\mathcal{T}$ and $B_\mathcal{T}$ renders the BLUE. ■

Because C is only required to be a basis for the cycle space, C is not unique. Consequentially, the choice of C affects the behavior of the JCSE because A and D depend on C. While Theorem 6 provides conditions for convergence based on the spectral radius of $D^{-1}A$, it does little to explain which cycle space matrices lead to convergent cycle space estimates. The optimal choice of C remains an

open problem; however, the following convergence results can be used to evaluate potential C matrices.

4.3.2 Convergence of JCSE on a Planar Graph

When a planar graph is embedded in the Euclidean Plane such that no edges cross, the plane is divided into contiguous regions called *faces* (see Figure 4.3 for an example of graph faces); each graph face characterizes a cycle that forms its boundary [21]. The *faces of the graph* are taken to be the set of interior faces in the graph, which is a basis for the cycle space of the graph [24]. The following theorem provides a convergence result for planar graphs. In this theorem, the faces of the graph are chosen to derive the basis for the cycle space.

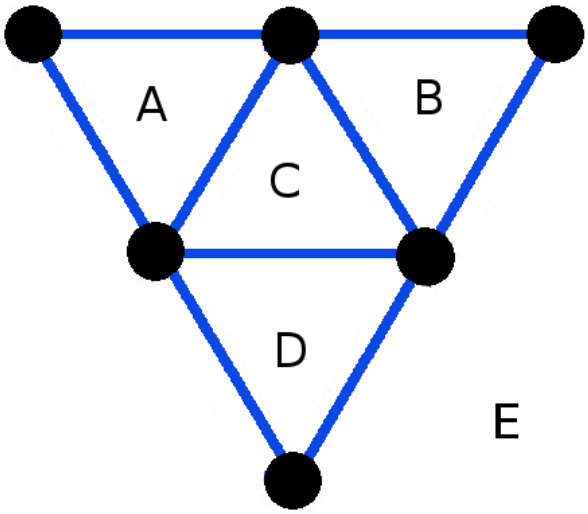


Figure 4.3: Simple graph with interior faces A,B,C, and D and exterior face E.

Theorem 7 *[Convergence for Planar Graphs]*

When $\mathcal{G}$ is a planar graph and C is the basis of the cycle space associated with the internal faces of the graph, then the JCSE, (4.15)*, converges and* $\lim_{t\to\infty} \hat{\boldsymbol{x}}^t = \hat{\boldsymbol{x}}^*$*, the BLUE of* $\boldsymbol{x}$*.*

Proof: The cycle degree matrix, D, is a block diagonal matrix whose blocks are sums of covariance matrices. As such, D and D^{-1} are symmetric positive definite matrices. The cycle Laplacian matrix is defined as $\mathcal{L}_{\mathrm{c}} \equiv C^{\mathsf{T}} P C$, where P is symmetric positive definite and C is a basis, hence $\mathcal{L}_{\mathrm{c}}$ is a symmetric positive definite matrix. From [19] (Theorem 7.6.3) all eigenvalues of $D^{-1}\mathcal{L}_{\mathrm{c}}$ are positive, from which:

$$\begin{aligned} \rho\left(D^{-1}\mathcal{L}_{\mathrm{c}}\right) &> 0 \\ \rho\left(I - D^{-1}\mathcal{L}_{\mathrm{c}}\right) &< 1 \\ \rho\left(I - D^{-1}(D - A)\right) &< 1 \\ \rho\left(D^{-1}A\right) &< 1 \end{aligned}$$

Hence, the JCSE algorithm converges. ■

4.3.3 Convergence Rate of JCSE on a Square Lattice

Currently, no closed form solution exists to find $\rho(J)$ for a general measurement graph. However, a closed form expression for $\rho(J)$ can be found for the JCSE algorithm applied to a square lattice of measurements with equal error covariances.

Theorem 8 *[Conv. Rate of JCSE on a Square Lattice]*

Consider a K-by-L square lattice of nodes connected by relative difference measurements with equal covariances. The convergence rate of the JCSE algorithm is:

$$\rho(D^{-1}A) = \frac{1}{2}\left[\cos\left(\frac{\pi}{K}\right) + \cos\left(\frac{\pi}{L}\right)\right].$$

Proof: In the JCSE algorithm, the iteration variables, $\hat{\mathbf{y}}_k$, can be stacked into a single vector, $\hat{\mathbf{y}}$. The update equation for $\hat{\mathbf{y}}$ is given by (4.12), whose convergence rate depends on the spectral radius of $D^{-1}A$. Because of symmetry in the lattice, the convergence rate can be found by examining the eigenvalues of the non-stacked update equation (4.9).

Consider a graph called the *cycles graph*, in which each cycle is represented by a node and there is an edge connecting two nodes if their respective cycles are adjacent. Let the columns of the cycle space matrix, C, be the set of simple cycle vectors corresponding to faces of the measurement graph, which is a K x

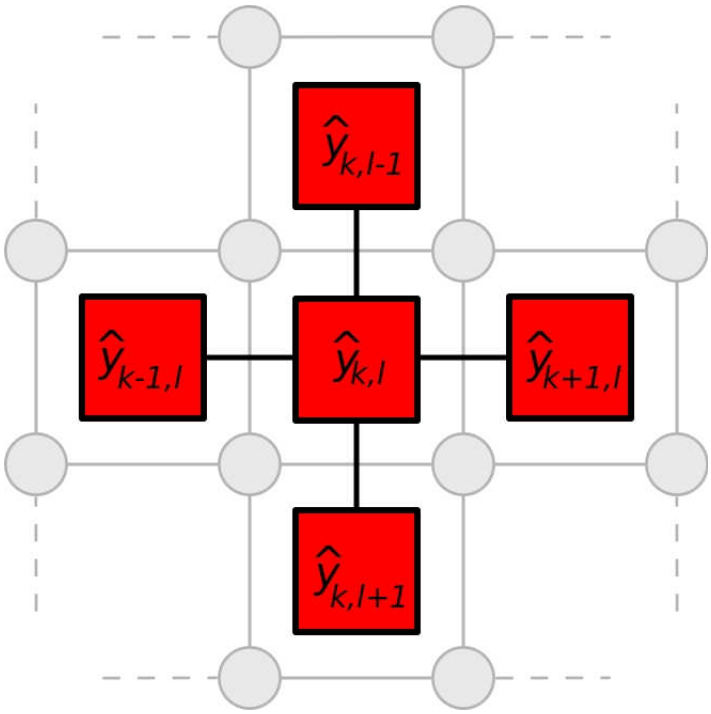


Figure 4.4: Example of the interconnection of cycles and iteration variables in a lattice graph.

L square lattice. The cycles graph is a $(K-1)$ x $(L-1)$ grid (see Figure 4.4). Enumerate the faces of the graph based on their grid location in the faces graph. Denote by $\hat{\mathbf{y}}_{k,l}$ the cycle iteration variable for the face in the k^{th} row and l^{th} column of the cycles graph, where $k \in \{1, \ldots, K-1\}$ and $l \in \{1, \ldots, L-1\}$. Because all of the faces in a square lattice have four edges, the cycle degree for each face is $4P_{\text{m}}$, where P_{m} is the covariance matrix of a single measurement. Furthermore, if two faces are adjacent, they share exactly one edge and have a cycle adjacency of P_{m}. With a cycle degree of $4P_{\text{m}}$ and a cycle adjacency of P_{m},

the iteration variable update equation, (4.9), can be expressed as:

$$\hat{\mathbf{y}}^{+}_{k,l} = \frac{1}{4}(\hat{\mathbf{y}}_{k+1,l} + \hat{\mathbf{y}}_{k-1,l} + \hat{\mathbf{y}}_{k,l+1} + \hat{\mathbf{y}}_{k,l-1}) - \frac{1}{4}P_{\mathrm{m}}^{-1}\boldsymbol{\Delta}_{k,l},$$

where $y_{0,l} = y_{K,l} = y_{k,0} = y_{k,L} = 0$. In steady state:

$$\hat{\mathbf{y}}^{*}_{k,l} = \frac{1}{4}(\hat{\mathbf{y}}^{*}_{k+1,l} + \hat{\mathbf{y}}^{*}_{k-1,l} + \hat{\mathbf{y}}^{*}_{k,l+1} + \hat{\mathbf{y}}^{*}_{k,l-1}) - \frac{1}{4}P_{\mathrm{m}}^{-1}\boldsymbol{\Delta}_{k,l}.$$

The error between $\hat{\mathbf{y}}_{k,l}$ and $\hat{\mathbf{y}}^{*}_{k,l}$, for $k \in \{1, \ldots, K-1\}$ and $l \in \{1, \ldots, L-1\}$, evolves as:

$$\hat{\mathbf{e}}^{+}_{k,l} = \frac{1}{4}(\hat{\mathbf{e}}_{k+1,l} + \hat{\mathbf{e}}_{k-1,l} + \hat{\mathbf{e}}_{k,l+1} + \hat{\mathbf{e}}_{k,l-1}), \tag{4.16}$$

where $\mathbf{e}_{0,l} = \mathbf{e}_{K,l} = \mathbf{e}_{k,0} = \mathbf{e}_{k,L} = 0$. Equation (4.16) is a Discrete Poisson Equation with homogeneous Dirichlet boundary conditions. From [14] (Chapter 4, Section 3.3), a Discrete Poisson Equation with homogeneous Dirichlet boundary conditions has eigenfunctions of the form:

$$\hat{\mathbf{e}}_{k,l} = \sin\left(\frac{\pi\mu k}{K}\right)\sin\left(\frac{\pi\nu l}{L}\right), \tag{4.17}$$

where $\mu \in \{1, \ldots, K-1\}$ and $\nu \in \{1, \ldots, L-1\}$. Eigenvalues for (4.16), λ, can be found by substituting in the eigenfunction, (4.17):

$$\lambda \hat{\mathbf{e}}_{k,l} = \frac{1}{4}(\hat{\mathbf{e}}_{k+1,l} + \hat{\mathbf{e}}_{k+1,l} + \hat{\mathbf{e}}_{k,l+1} + \hat{\mathbf{e}}_{k,l-1}),$$

as such:

$$\begin{aligned}\lambda \left[\sin\left(\frac{\pi\mu k}{K}\right) \sin\left(\frac{\pi\nu l}{L}\right)\right] = \frac{1}{4}\Bigg[&\sin\left(\frac{\pi\mu(k+1)}{K}\right)\sin\left(\frac{\pi\nu l}{L}\right) + \\ &\sin\left(\frac{\pi\mu(k-1)}{K}\right)\sin\left(\frac{\pi\nu l}{L}\right) + \\ &\sin\left(\frac{\pi\mu k}{K}\right)\sin\left(\frac{\pi\nu(l+1)}{L}\right) + \\ &\sin\left(\frac{\pi\mu k}{K}\right)\sin\left(\frac{\pi\nu(l-1)}{L}\right)\Bigg]. \qquad (4.18)\end{aligned}$$

Making use of the trigonometric identity, $\sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b)$, (4.18) reduces to:

$$\begin{aligned}\lambda \left[\sin\left(\frac{\pi\mu k}{K}\right) \sin\left(\frac{\pi\nu l}{L}\right)\right] = \frac{1}{2}\Bigg[&\sin\left(\frac{\pi\mu k}{K}\right)\sin\left(\frac{\pi\nu l}{L}\right)\cos\left(\frac{\pi\mu}{K}\right) + \\ &\sin\left(\frac{\pi\mu k}{K}\right)\sin\left(\frac{\pi\nu l}{L}\right)\cos\left(\frac{\pi\nu}{L}\right)\Bigg],\end{aligned}$$

which simplifies further to:

$$\lambda = \frac{1}{2}\left[\cos\left(\frac{\pi\mu}{K}\right) + \cos\left(\frac{\pi\nu}{L}\right)\right],$$

which is largest in magnitude when $\mu = 1$ and $\nu = 1$. Hence, the spectral radius of linear system (4.16) is

$$\lambda_{\max} = \frac{1}{2}\left[\cos\left(\frac{\pi}{K}\right) + \cos\left(\frac{\pi}{L}\right)\right].$$

■

4.3.4 Single Step Convergence

As defined, the entries of C are from the set $\{1, 0, -1\}$. The graphs cycles have a physical representation; they are a collection of edges. If this constraint on C is relaxed and the elements are allowed to be real numbers, the following result is obtained.

Theorem 9 *[Convergence in One Iteration]*

If C is chosen to be a orthogonal basis, then the JCSE algorithm converges in a single iteration.

Proof: When C is chosen to be a orthogonal basis of the cycle space, the cycle Laplacian matrix, $\mathcal{L}_{\mathrm{c}}$, is a block diagonal matrix, with blocks of size d, the dimension of the measurements. Consequentially, all elements of cycle adjacency matrix, A, and iteration matrix, J, are equal to zero. Hence, the cycle iteration variables, y_k, are set to the cycle discrepancy, $\boldsymbol{\Delta}_k$, on the first iteration. ■

In general, when C is chosen to be a orthogonal basis for the cycle space, the cycle Laplacian matrix, $\mathcal{L}_{\mathrm{c}}$, will not be sparse, indicating that cycle leaders performing JCSE will need to collect and share information with many other cycle leaders. While this is not a desirable trait in a distributed algorithm, iteration on this basis may render solutions with less communication than the classic centralized solution (one in which all measurements are passed to a central processor). This is because the cycle leaders only need to know the cycle discrepancy (a weighted sum of measurements as opposed to the measurements themselves).

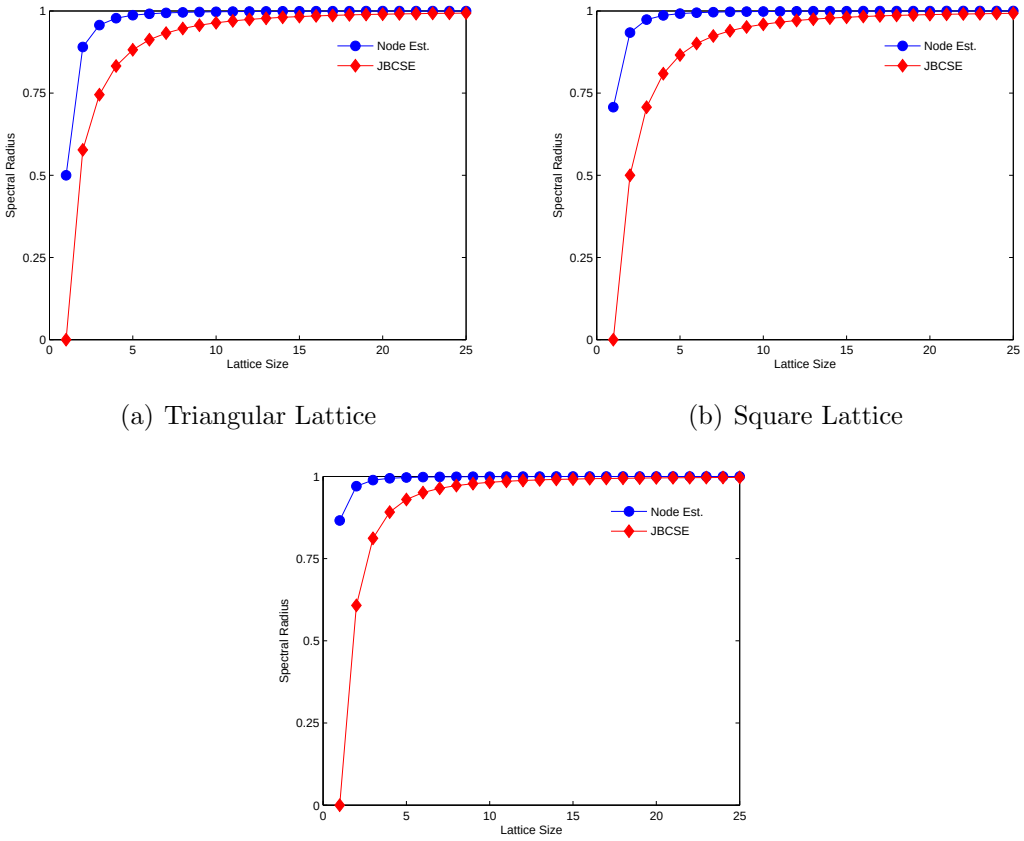
For example, consider a ring graph with n nodes. Computing the sum around the ring takes n messages. However, to pass all of the messages to one node for processing would take $\frac{n^2-1}{4}$ messages (for an odd number of nodes). This can be seen by considering a graph with an odd number of nodes. The furthest distance a measurement must travel is $\frac{n-1}{2}$ (there are two such measurements).

The next furthest distance is $\frac{n-1}{2} - 1$, etc. Two times the sum from 1 to $\frac{n-1}{2}$ is $\frac{n^2-1}{4}$.

4.4 Simulations

The chief aim of this work is to develop a distributed estimation algorithm that swiftly converges to the BLUE. As such, it is appropriate to offer a comparison with previously developed algorithms. The Jacobi Algorithm introduced in [4] is a node estimation method that uses relative state difference measurements to compute the optimal estimate of a graph's state. In the Jacobi Algorithm, each agent considers a local subgraph consisting of itself, its one-hop neighbors, and all edges incident on itself. At each iteration, each agent shares (with its neighbors) its estimate of its own position. Each agent then computes the BLUE (3.14) of its local subgraph assuming that its neighbors' own estimates of state are reference. The Jacobi Algorithm With Flagged Initialization (JAWFI) is an improved version of the Jacobi algorithm where agents with unknown state initialize their estimates to a flagged condition. When computing the BLUE of its local subgraph, a agent only includes estimates of neighbors with non-flagged state. Figure 4.5 compares the spectral radii of this node estimation method and the JCSE algorithm (where the cycle space is chosen to be the faces of the

graph). For triangular, square, and hexagonal lattices, the JCSE always has a smaller spectral radius and therefore has a faster convergence rate.



(a) Triangular Lattice

(b) Square Lattice

(c) Hexagonal Lattice

Figure 4.5: The spectral radii of the Jacobi iteration's state transition matrices for various sized (a) triangular, (b) square, and (c) hexagonal lattices.

Comparing the spectral radii for these two algorithms only compares the slowest modes of the algorithms. To provide a valid comparison of the two algorithms, one must consider the additional overhead of initializing the algorithms

and the fact that the iterations will not necessarily occur on the slowest possible modes. While it is not practical to compare performance on every type of graph, this section first considers a range of regular graphs that include triangular, square, and hexagonal lattices. This section then considers random graphs including Delaunay, Gabriel, and relative neighborhood graphs [49]. The simulation results demonstrate the performance gains possible when using JCSE with respect to the JAWFI introduced in [4].

First, simulations were run on lattices of various size. The nodes in each lattice are one unit away from their edge neighbors. The goal of each agent is to estimate its (two dimensional) position from measurements that are corrupted by zero mean Gaussian noise with standard deviation 0.25. For each set of measurements, both algorithms were run until the 2-norm of the estimation error normalized by the number of agents, $\frac{\|\hat{\mathbf{x}}^* - \hat{\mathbf{x}}\|}{n}$, was below 0.025 (one tenth of the standard deviation of the measurement noise). This process was repeated on two thousand sets of measurements for each size of each lattice graph. The size of a lattice was taken to be the number of tessellations on one side of a given lattice. Figure 4.6 shows the first three sizes for each type of lattice.

Figure 4.7(a) depicts the ratio of the average convergence times, computed as the average number of iterations needed by the JCSE over the average number of iterations needed by for the JAWFI method, as a function of the lattice size.

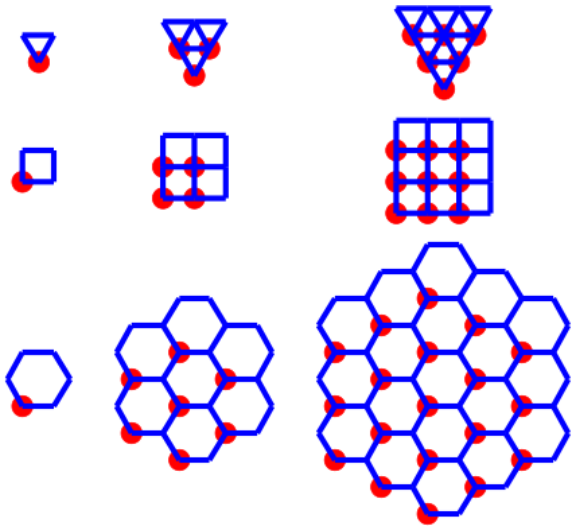
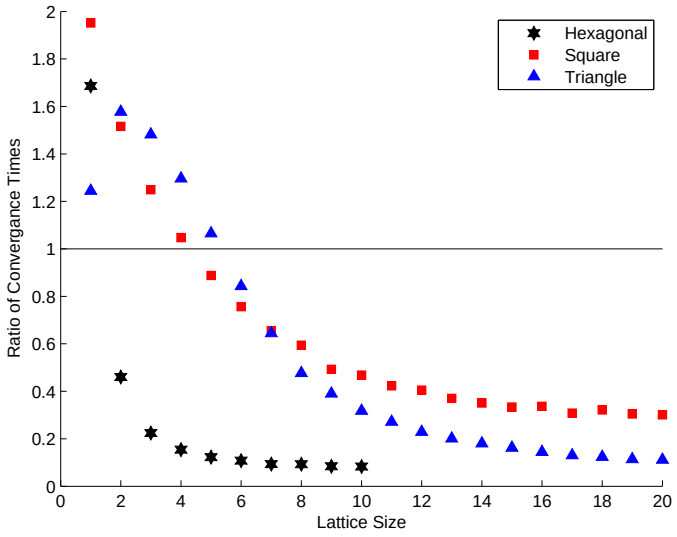
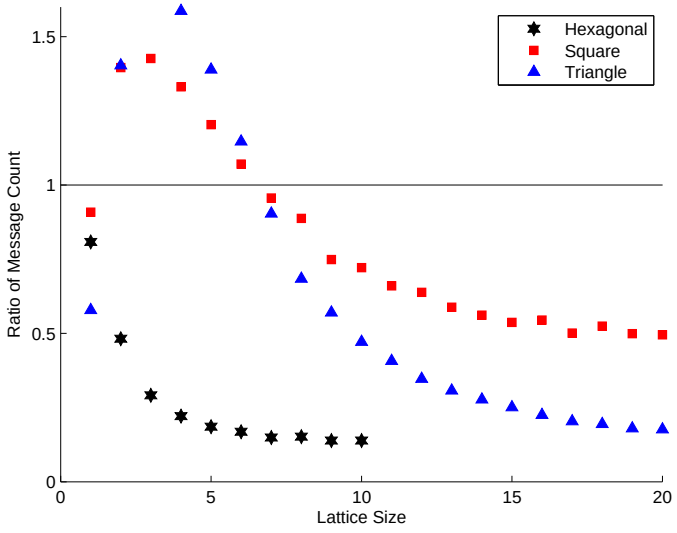


Figure 4.6: Size one, two, and three lattices with cycle leaders highlighted.

Significant gains in terms of the number of iterations can be seen for large graphs as predicted from the analysis of the convergence rates (see Figure 4.5). Figure 4.7(b) charts the ratio of the average number of messages passed by the JCSE algorithm over the average number of messages passed by the JAWFI method, as a function of lattice size. For the JCSE algorithm, all messages passed during computation of cycle discrepancies were counted, during the flagged initialization process, between cycle leaders during the iteration process, and from cycle leaders to other agents for state updates. In both of these figures, the horizontal axis corresponds to the size of the lattice. For smaller lattices, the overhead of obtaining the cycle discrepancies makes JCSE less advantageous, but for larger graphs, JCSE shows faster convergence with less message passing. For both



(a) Iteration Count



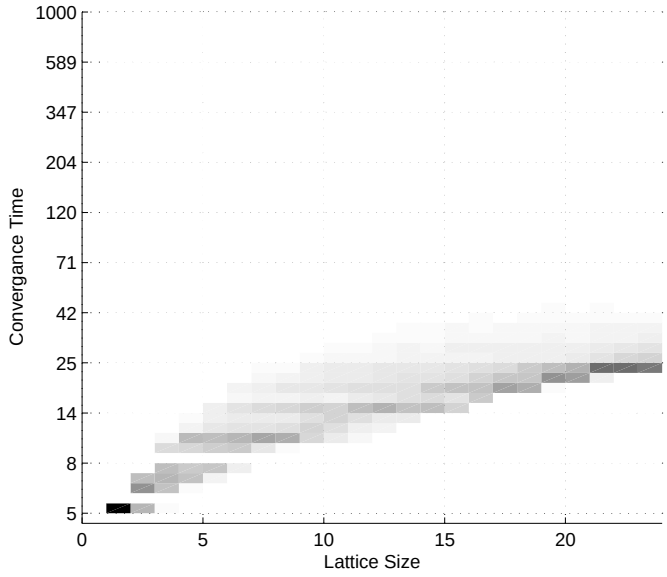
(b) Message Count

Figure 4.7: (a) The average number of iterations for JCSE to finish over the average number of iterations for JAWFI to finish as a function of graph size. (b) The average number of messages passed during JCSE over the average number of messages passed during JAWFI as a function of graph size.

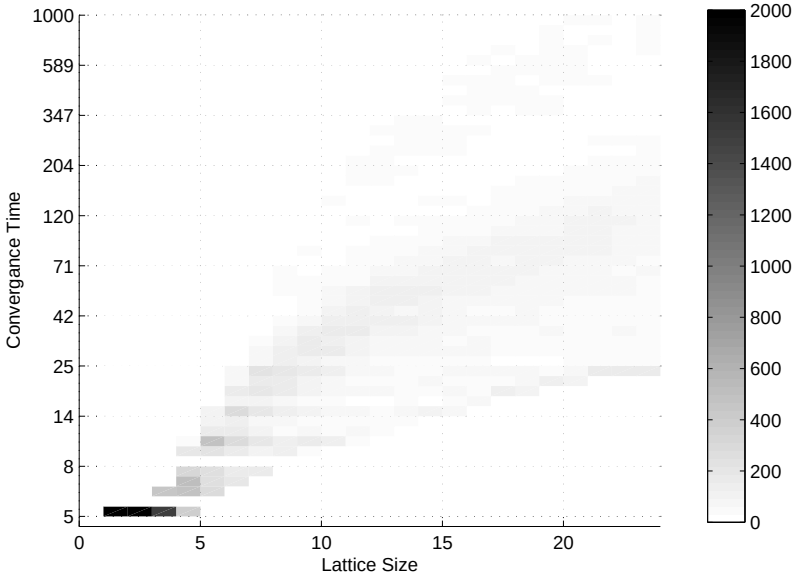
plots in Figure 4.7, the results for hexagonal lattices stop at size ten due to the lengthy JAWFI convergence times.

Figures 4.8, 4.9, and 4.10 are density plots of the number of iterations for each algorithm to reach the completion criteria on triangular, square, and hexagonal lattices, respectively. Due to skewness in the histograms, these figures are plotted using a log-linear scaling. These histograms reveal another benefit of the JCSE algorithm; the settling times for the JCSE are fairly consistent.

Similar simulations were run on Delaunay, Gabriel, and relative neighborhood graphs. For each simulation, a number of agents are randomly placed inside the unit box and edges are connected according to the graphs' definitions. Again, the goal of each agent is to estimate its position from measurements that are corrupted by zero mean Gaussian noise with standard deviation 0.25. For each random graph, both algorithms were run until $\frac{\|\hat{\mathbf{x}}^* - \hat{\mathbf{x}}\|}{n}$ was below 0.025. For each type of graph, this process was repeated two thousand times. Figure 4.11 depicts the ratio of the average convergence times for the random graphs. The results show that for these three classes of random graphs, the JCSE leads to faster convergence in large graphs.

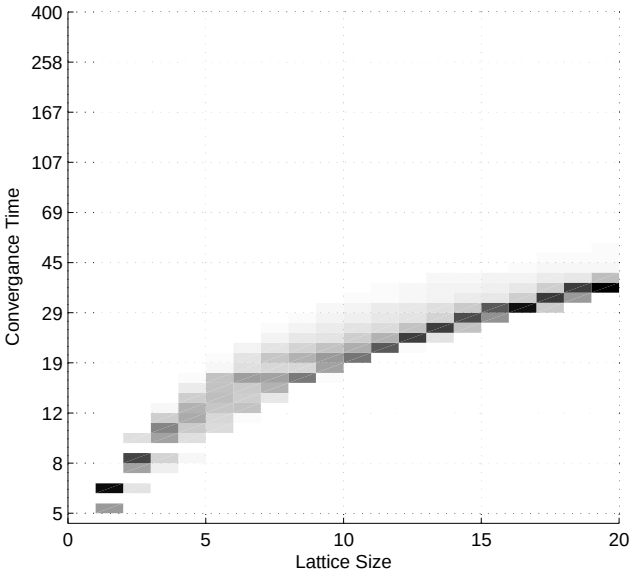


(a) JCSE Algorithm

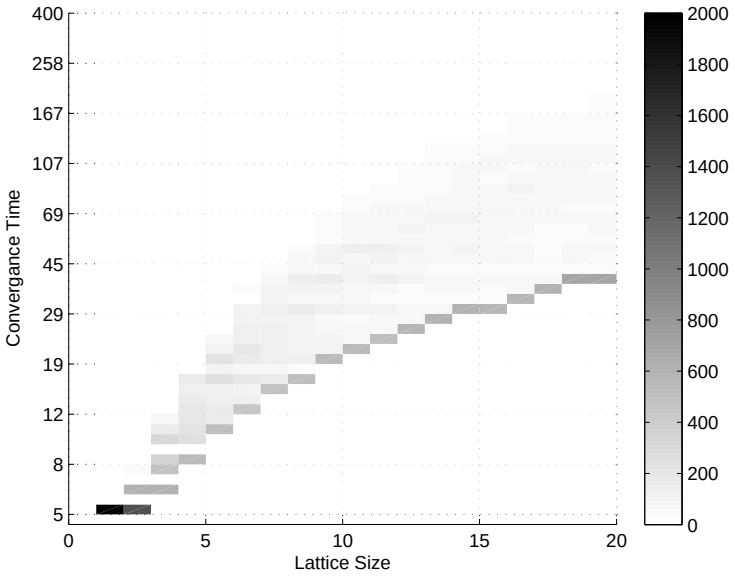


(b) JAWFI Algorithm

Figure 4.8: Number of iterations to meet the completion criteria on a triangular lattice for the (a) JCSE algorithm and the (b) JAWFI algorithm from [4].

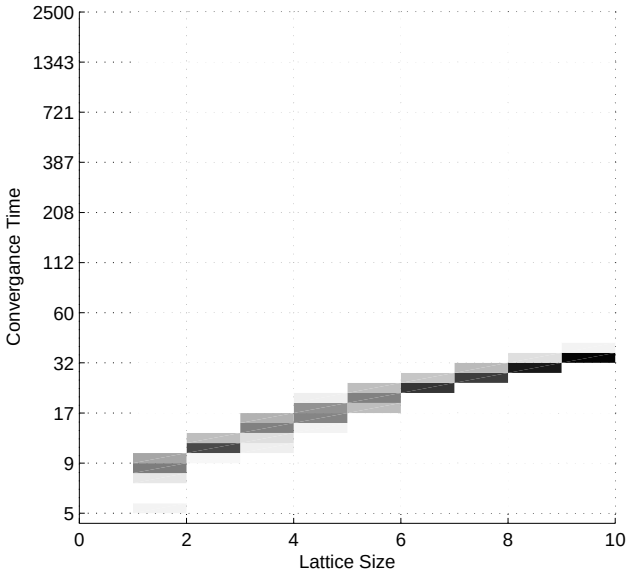


(a) JCSE Algorithm

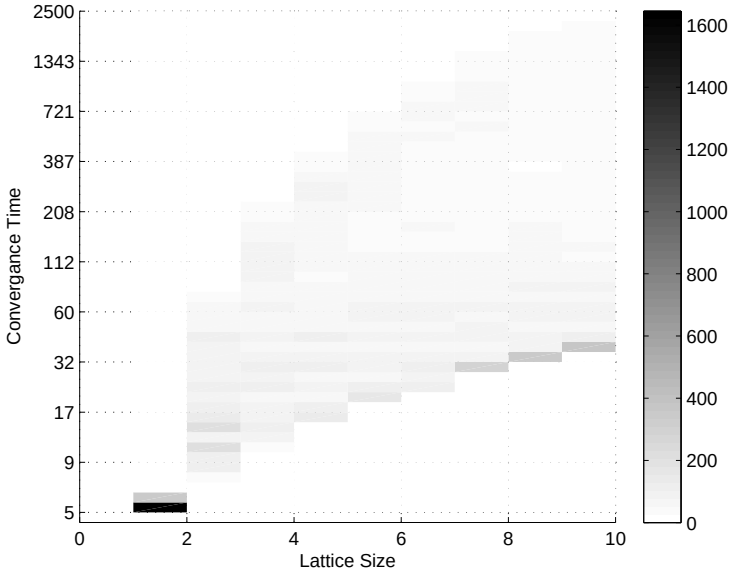


(b) JAWFI Algorithm

Figure 4.9: Number of iterations to meet the completion criteria on a square lattice for the (a) JCSE algorithm and the (b) JAWFI algorithm.



(a) JCSE Algorithm



(b) JAWFI Algorithm

Figure 4.10: Number of iterations to meet the completion criteria on a hexagonal lattice for the (a) JCSE algorithm and the (b) JAWFI algorithm.

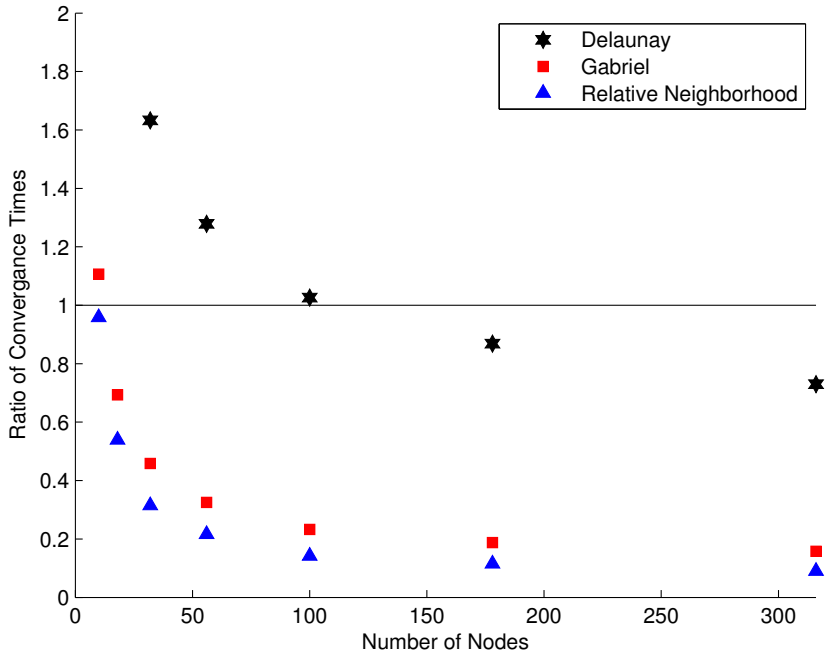


Figure 4.11: The average number of iterations for JCSE to finish over the average number of iterations for JAWFI to finish as a function of the number of nodes in for random graphs.

4.5 Conclusion

This chapter introduced cycle space estimation from relative measurements and showed its equivalence to BLUE. A Jacobi-based distributed algorithm that was shown to converge to the BLUE (under appropriate conditions) was explored. Conditions for convergence of the distributed algorithm were given, as well as closed form convergence rates for the algorithm on a grid network. It

was shown that cycle space estimation reduces communication and convergence time for a large class of graphs.

To make the methods in this chapter more attractive, a few items need to be explored further. The inverse relationship between distributability and convergence rate leads to an interesting tuning problem; the JCSE algorithm can be tuned for faster convergence by choosing a set of cycles that is less local. It is also interesting to look at how the selection of the cycle leader agents in the graph affects performance of distributed cycle space algorithms; cycle leaders that are closer to each other (in the communication graph) can share information faster.

4.6 Appendix

Lemma 2 *Given matrices α and β where α and the transpose of β, β^T, are full column rank, the nullspace of β is the image of α if and only if*

$$\alpha(\alpha^T\alpha)^{-1}\alpha^T = I - \beta^T(\beta\beta^T)^{-1}\beta.$$

Proof: Denote the orthogonal complement of a subspace by $\perp$,

$$\text{image}(\alpha) = \text{null}(\beta) = \text{image}(\beta^T)^{\perp}.$$

Replacing α and β^{T} with their respective projection matrices,

$$\text{image}(\alpha(\alpha^{\mathsf{T}}\alpha)^{-1}\alpha^{\mathsf{T}}) = \text{image}(\beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta)^{\perp}.$$

As demonstrated in [47], a matrix projection, $\mathcal{P}$, is an orthogonal complement to $I - \mathcal{P}$, implying

$$\text{image}(\alpha(\alpha^{\mathsf{T}}\alpha)^{-1}\alpha^{\mathsf{T}}) = \text{image}(I - \beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta).$$

Because the projection matrices associated with two complementary subspaces are unique [28],

$$\alpha(\alpha^{\mathsf{T}}\alpha)^{-1}\alpha^{\mathsf{T}} = I - \beta^{\mathsf{T}}(\beta\beta^{\mathsf{T}})^{-1}\beta.$$

■

Chapter 5

Summary

This dissertation presented extensions on a previous body of work known as estimation from relative measurements (ERM). ERM encompasses a range of problems in which a sensor network has some environmental state that needs to be estimated. The state varies across the environment and each sensor is concerned with estimating the state at its local. The sensors cannot measure this state absolutely, rather a pair of sensors can make a noisy relative difference measurement of their respective states. Existing literature for this problem provides both the centralized optimal state estimate for the ERM problem and a distributed iterative algorithm that converges to the optimal estimate. Both of these methods relate the sensors and their measurements to a graph and then use a combination of graph theory and regression analysis to achieve their results. The preexisting distributed algorithm is elegant in that it requires little more than reliable communications for convergence.

The existing distributed estimation algorithm was extended to include networks with dynamic agents. It was shown that if the agents in the network can make inertial measurements, as well as relative measurements to other agents then the ERM framework can be used in estimating the agents' states. A distributed estimation algorithm that is based on the preexisting distributed algorithm for static networks was presented. If one freezes the agents at a particular time and implements the preexisting iterative distributed method, the resulting estimates converge to those of an RTS smoother. In a real system one cannot freeze the state and as a result the estimates never converge to optimal. Furthermore, as time goes on and more measurements are added to the network, the estimation problem grows large. At some point (dependent on computation and communication limitations) agents must cease iterating the estimates their older states and concentrate on estimating the most recent states. This truncation leads to a lasting discrepancy between the iterative estimates and the optimal estimates. As such this method approximates the optimal estimates. It was shown that with very few iterations and very little memory, the algorithm achieves estimates that are quite close to optimal.

It was shown that under certain conditions the relative difference measurements can be relaxed to include general linear equations. This is useful in systems with dynamics where inertial measurements are not available. In such systems,

the state dynamics can be modeled with linear equations. These modeled dynamics can be incorporated (in conjunction with relative measurements) into the aforementioned distributed estimation algorithm. It was shown that this is possible when the modeled dynamics and the relative measurements constitute an observable system.

The last part of this dissertation introduced estimation on the cycle space of a graph. In the ERM problem, the network's measurements can be modeled as a graph. In such a graph, there is an edge corresponding to each of the relative measurements. In the absence of noise, the sum of measurements around a cycle in the graph is zero. With noisy measurements, the sum of measurements around a cycle is non-zero. It was shown that this cycle discrepancy can be appropriately divvied up and added to the measurements of the cycle, and the result can be used to recover the optimal state estimates. An algorithm that achieves this result and showed that in large graphs it converges faster and with fewer communications then previously available distributed methods was provided. This method's convergence is dependent on what graph cycles are chosen for its implementation. It was shown that for planar graphs, convergence is guaranteed when one chooses cycles that are the faces of the graph.

Bibliography

[1] A.C. Aitken. On least squares and linear combinations of observations. *Proceedings of the Royal Society of Edinburgh*, 55:42 – 48, 1935.

[2] P. Alriksson and A. Rantzer. Distributed Kalman filtering using weighted averaging. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, 2006.

[3] P. Alriksson and A. Rantzer. Experimental evaluation of a distributed Kalman filter algorithm. In *46th IEEE Conference on Decision and Control*, December 2007.

[4] P. Barooah and J.P. Hespanha. Distributed estimation from relative measurements in sensor networks. In *Proceedings of the 3rd International Conference on Intelligent Sensing and Information Processing (ICISIP)*, 2005.

[5] P. Barooah and J.P. Hespanha. Estimation from relative measurements: Error bounds from electrical analogy. In *Proceedings of the 2nd International Conference on Intelligent Sensing and Information Processing (ICISIP)*, 2005.

[6] P. Barooah, W.J. Russell, and J.P. Hespanha. Approximate distributed Kalman filtering for cooperative multi-agent localization. In *Proc. of the Int'l Conference on Distributed Computing in Sensor Systems*, 2010.

[7] B. Bell and G. Pillonetto. A distributed Kalman filter. In *Proc. of 1st IFAC Workshop on Estimation and Control of Networked Systems*, 2009.

[8] D.S. Bernstein. *Matrix Mathematics*. Princeton University Press, Princeton, NJ, 2009.

[9] J. Borenstein, H.R. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems, Special Issue on Mobile Robots*, 14(4):231–249, April 1997.

[10] R. Carli, A Chiuso, L. Schenato, and S. Zampieri. Distributed Kalman filtering using consensus strategies. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[11] F.S. Cattivelli, C.G. Lopes, and A.H. Sayed. Diffusion strategies for distributed Kalman filtering: Formulation and performance analysis. In *Proceedings of the IARP Workshop on Cognitive Information Processing*, 2008.

[12] F.S. Cattivelli and A.H. Sayed. Diffusion mechanisms for fixed-point distributed Kalman smoothing. In *Proceedings of the 16th European Signal Processing Conference*, 2008.

[13] H. Chen, T. Kirubarajan, and Y. Bar-Shalom. Performance limits of track-to-track fusion versus centralized estimation: Theory and application. *IEEE Transactions on Aerospace and Electronic Systems*, 39(2):386–400, 2003.

[14] D. Dubin. *Numerical and Analytical Methods for Scientists and Engineers Using Mathematica*. Wiley-Interscience, Hoboken, NJ, 2003.

[15] J.A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Trans. Automatic Control*, 49(9):1465–1476, 2004.

[16] D.G. Feingold and R.S. Varga. Block diagonally dominant matrices and generalizations of the gershgorin circle theorem. *Pacific Journal of Mathematics*, 12:1241–1250, December 1962.

[17] G.H. Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, 3rd edition, 1996.

[18] F. Gustafsson and F. Gunnarsson. Positioning using time-difference of arrival measurements. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2003.

[19] R.A. Horn and Johnson C.R. *Matrix Analysis*. Cambridge University Press, New York, NY, 1985.

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

[21] R. Johnsonbaugh. *Discrete Mathematics*. Pearson Prentice Hall, Upper Saddle River, NJ, 2009.

[22] G. Kirchhoff. Über den durchgang eines elektrischen stromees durch eine ebene, insbesondere durch eine kreisförmige. *Annalen der Physik und Chemie*, 64(4):497–514, 1845.

[23] R. Kurazume, S. Nagata, and S. Hirose. Cooperative positioning with multiple robots. In *the IEEE International Conference in Robotics and Automation*, 1994.

[24] S. Mac Lane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, 3:460–472, 1937.

[25] A. Makadia and K. Daniilidis. Correspondenceless ego-motion estimation using an imu. In *IEEE International Conference on Robotics and Automation*, 2005.

[26] G. Mao, B. Fidan, and B. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(10):2529–2553, 2007.

[27] J.M. Mendel. *Lessons in Estimation Theory for Signal Processing, Communications and Control*. Prentice Hall, 1995.

[28] C.D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, 2001.

[29] A.I. Mourikis and S.I. Roumeliotis. Performance analysis of multirobot cooperative localization. *IEEE Transactions on Robotics*, 22(4):666–681, August 2006.

[30] S. Mueller, R.P. Tsang, and D. Ghosal. Multipath routing in mobile ad hoc networks: Issues and challenges. In *Performance Tools and Applications to Networked Systems, LNCS*, volume 2965, pages 209–234. Springer Berlin/Heidelberg, 2004.

[31] D. Nistér, O. Naroditsky, and J.R. Bergen. Visual odometry. In *Conference on Computer Vision and Pattern Recognition (CVPR '04)*, 2004.

[32] N. Okello and S. Challa. Joint sensor registration and track-to-track fusion for distributed trackers. *IEEE Transactions on Aerospace and Electronic Systems*, 40(3):808–823, 2004.

[33] R. Olfati-Saber. Distributed Kalman filtering in sensor networks. In *Proc. of the 46th IEEE Conference on Decision and Control*, 2007.

[34] C.F. Olson, L.H. Matthies, M. Schoppers, and M.W. Maimone. Rover navigation using stereo ego-motion. *Robotics and Autonomous Systems*, 43(4):215–229, June 2003.

[35] J.M. Ortega. *Numerical Analysis A Second Course*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[36] T. Oskiper, Z. Zhu, S. Samarasekera, and R. Kumar. Visual odometry system using multiple stereo cameras and inertial measurement unit. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, 17-22 June 2007.

[37] N. Patwari, A.O. Hero III, J. Ash, R.L. Moses, S. Kyperountas, and N.S. Correal. Locating the nodes: Cooperative localization in wireless sensor networks. *IEEE Signal Processing Magazine*, 22(4):54–69, 2005.

[38] G. Piovan, I. Shames, B. Fidan, F. Bullo, and Anderson R. On frame and orientation localization for relative sensing networks. In *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.

[39] H.E. Rauch, F. Tung, and C.T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3:1445–1450, August 1965.

[40] I.M. Rekleitis, G. Dudek, and E.E. Milios. Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In *the IEEE/RSJ International Conference on Intelligent Robots and System*, 2002.

[41] R.T. Rockafellar. *Network Flows and Monotropic Optimization*. Athena Scientific, Nashua, NH, 1998.

[42] S. Roumeliotis and G.A. Bekey. Distributed multirobot localization. *IEEE Transactions on Robotics and Automation*, (5):781–795, October 2002.

[43] S.I. Roumeliotis and G.A. Bekey. Collective localization: A distributed Kalman filter approach to localization of mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, 2000.

[44] W.J. Russell, D.J. Klein, and J.P. Hespanha. Optimal estimation on the graph cycle space. In *Proceedings of the American Control Conference*, 2010.

[45] D.P. Spanos and R.M. Murray. Distributed sensor fusion using dynamic consensus. In *IFAC World Congress*, 2005.

[46] D.P. Spanos, R. Olfati-Saber, and R.M. Murray. Approximate distributed Kalman filtering in sensor networks with quantifiable performance. In *4th international symposium on Information processing in sensor networks (IPSN '05)*, 2005.

[47] E.S. Suhubi. *Functional Analysis*. Kluwer Academic Publishers, Norwell, MA, 1990.

[48] W.F. Trench. Solution 21-3.1 eigenvalues and eigenvectors of two symmetric matrices. *Image: The Bulletin of the International Linear Algebra Society*, 22:22–23, April 1999.

[49] D. Wagner and R. Wattenhofer. *Algorithms for sensor and ad hoc networks: advanced lectures*. Springer, New York, NY, 2007.

[50] C. Wai-Kai. *Graph Theory and Its Engineering Applications*. World Scientific Pub Co Inc, River Edge, NJ, 1997.

[51] L. Xiao, S. Boyd, and S. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.

[52] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the 4th Int'l Conference on Information Processing in Sensor Networks*, 2005.

[53] P. Zhang and M. Martonosi. LOCALE: collaborative localization estimation for sparse mobile sensor networks. In *International Conference on Information Processing in Sensor Networks (IPSN)*, pages 195–206, 2008.