Estimation on Graphs From Relative Measurements

A Paper by Prabir Barooah and João P. Hespanha

Presented by Mark Edwards

mark.edards727@myci.csuci.edu
University of California Channel Islands

May 12, 2016


Building Consensus
One thumb at a time.

Overview

Relative Measurements

Relative Measurements are sensor readings taken relative to other nodes in a sensor network

Relative Measurements

Relative Measurements are *NOT*

- Absolute Position (GPS)

Relative Measurements

Relative Measurements are *NOT*

- Absolute Position (GPS)
- Absolute Temperature (Thermometer Sensor)

Relative Measurements

Relative Measurements are *NOT*

- Absolute Position (GPS)
- Absolute Temperature (Thermometer Sensor)
- Absolute Velocity (Accelerometer)

Relative Measurements

Relative Measurements *ARE*

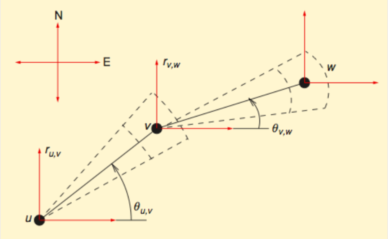
- Relative Position from Heading and Bearing

Relative Measurements

Relative Position

- Heading measured using Compass Sensor
- Distance measured using Optical Sensor
- We want relative Cartesian displacement

Example

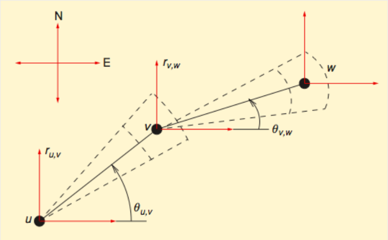


Relative Measurements

Relative Position

- Let w be our reference "master"
- Let $\theta_{u,v} = 30^\circ, d_{u,v} = 1$ $\theta_{v,w} = 60^\circ, d_{v,w} = 1$
- Then u is at $\left(-\frac{\sqrt{3}+1}{2}, -\frac{\sqrt{3}+1}{2}\right)$

Example



Relative Measurements

Relative Measurements *ARE*

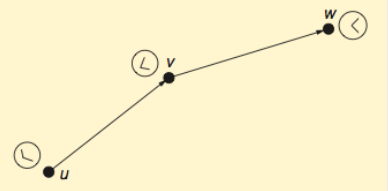
- Relative Position from Heading and Bearing
- Time-Synchronization

Relative Measurements

Time-Synchronization

- We want to synchronize time between u and "master node" w
- We want to update u to $t_u = t_u + (t_v - t_u) + (t_w - t_v)$

Example



Relative Measurements

Relative Measurements *ARE*

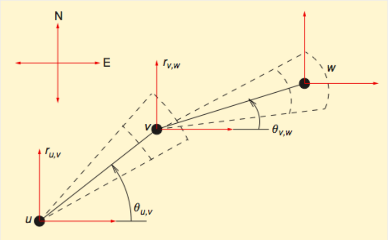
- Relative Position from Heading and Bearing
- Time-Synchronization
- Relative Velocity

Relative Measurements

Relative Velocity

- Let w be our reference "master"
- We track relative position over time to get velocity
- We want $\dot{u} = \dot{u} + (\dot{v} - \dot{u}) + (\dot{w} - \dot{v})$

Example



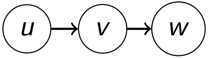
Relative Measurements

A few points:

- The preceding graphs are *Measurement Graphs* not *Connectivity Graphs*
- We implicitly assume bi-directional communication between all nodes that share a measurement edge.

Relative Measurements

Each “hop” in a measurement graph is a measurement ζ with some error ϵ



$$\zeta_{u,v} = u - v + \epsilon_{u,v}, \zeta_{v,w} = v - w + \epsilon_{v,w}$$

Relative Measurements

This can be written in "matrix form" as

$$\mathbf{z} - \mathcal{A}_r^T \mathbf{x}_r = \mathcal{A}_b^T \mathbf{x} + \epsilon$$

- $\mathbf{z}$ are the measurements
- $\mathcal{A}_r$ is the incidence matrix for the reference nodes $\mathbf{x}_r$
- $\mathcal{A}_b$ is the incidence matrix for unknown nodes $\mathbf{x}$
- ϵ is still the error.

Relative Measurements

We may also define the covariance matrix $\mathcal{P}$ as

$$P := E[\epsilon \epsilon^T]$$

Since we assume that ϵ is a random vector with zero mean.

Relative Measurements

And based on this, we can now apply least squares to get the *Best Least Unbiased Estimator*

$$\hat{\mathbf{x}}^* := \mathcal{L}^{-1}\mathbf{b}$$

Where

$$\mathcal{L} := \mathcal{A}_b \mathcal{P}^{-1} \mathcal{A}_b^T$$

$$b := \mathcal{A}_b \mathcal{P}^{-1} \left(\mathbf{z} - \mathcal{A}_r^T \mathbf{x}_r \right)$$

Relative Measurements

We may also

$$\Sigma := E[(\mathbf{x} - \hat{\mathbf{x}}^*)(\mathbf{x} - \hat{\mathbf{x}}^*)^T] = \mathcal{L}^{-1}$$

Relative Measurements

For this particular graph, we see that error scales linearly with respect to the number of nodes

$$\longrightarrow u \longrightarrow v \longrightarrow w \longrightarrow$$

$$\epsilon_{u,w} = \epsilon_{u,v} + \epsilon_{v,w}$$

$$\epsilon_{i,j} = \sum_{k=i}^{j-1} \epsilon_{k,k+1}$$

Relative Measurements

This begets the question, "can we more generally put a lower bound on our error"?

Relative Measurements

This begets the question, “can we more generally put a lower bound on our error”?

Yes!

Error Bounds

In order to address error bounds, we need a way to address structure within the graph.

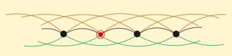
Error Bounds

In order to address error bounds, we need a way to address structure within the graph.

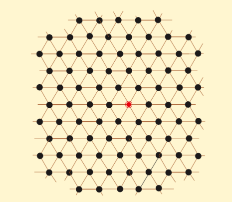
A traditional method of quantifying the structure of a graph is with the degrees of each node.

Error Bounds

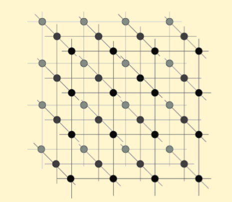
But this doesn't work. . .



(a) Linear Error Scaling



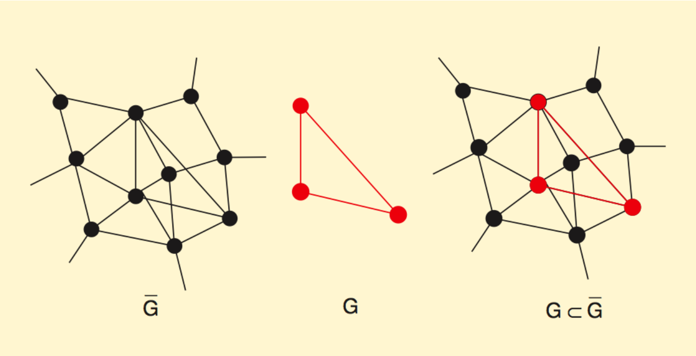
(b) Logarithmic Error Scaling



(c) Constant Error Scaling

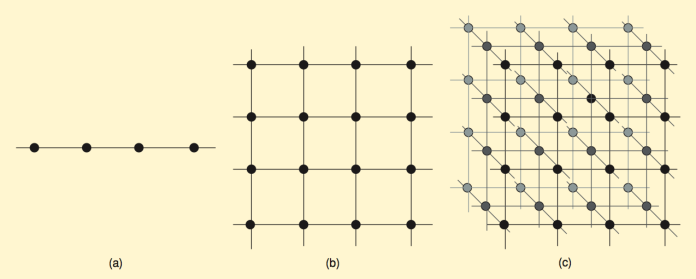
Error Bounds

In order to more directly measure structure, we define the concept of an *embedding*



Error Bounds

And the concept of a *lattice*



Error Bounds

And make the following claims with respect to the number of nodes:

- A 1D lattice has error that scales linearly

Error Bounds

And make the following claims with respect to the number of nodes:

- A 1D lattice has error that scales linearly
- A 2D lattice has error that scales logarithmically

Error Bounds

And make the following claims with respect to the number of nodes:

- A 1D lattice has error that scales linearly
- A 2D lattice has error that scales logarithmically
- A 3D lattice has error that is constant with respect to the number of nodes!

Error Bounds

Finally, we generalize this using embeddings by noting that if a graph G is embedded in another graph $\bar{G}$, that the error G is at least that of $\bar{G}$

Error Bounds

Finally, we generalize this using embeddings by noting that if a graph G is embedded in another graph $\bar{G}$, that the error G is at least that of $\bar{G}$

With this, we now turn to estimation methods that approach these error bounds.

Jacobi Iteration

Jacobi Iteration is an algorithm we inherit from Linear Algebra.

Jacobi Iteration

Jacobi Iteration is an algorithm we inherit from Linear Algebra.

Intuitively, it can be understood as "guess and check"

Jacobi Iteration

In layman's terms, the algorithm is

1. Arbitrarily guess values for 1 hop neighbors.

Jacobi Iteration

In layman's terms, the algorithm is

1. Arbitrarily guess values for 1 hop neighbors.
2. At the *i*th iteration, use the current estimates of 1 hop neighbors to estimate each node's current value. Each node then broadcasts it's estimate of its own value to all its 1 hop neighbors.

Jacobi Iteration

In layman's terms, the algorithm is

1. Arbitrarily guess values for 1 hop neighbors.
2. At the ith iteration, use the current estimates of 1 hop neighbors to estimate each node's current value. Each node then broadcasts it's estimate of its own value to all its 1 hop neighbors.
3. At the end of the ith iteration, each node then uses the broadcasted estimates for the $i + 1$st iteration.

Jacobi Iteration

Mathematically, the second step involves solving

$$\left(\sum_{e \in E_u} P_e^{-1}\right) \hat{x}_u^{(i+1)} = \sum_{e \in E_u} P_e^{-1} \left(\hat{x}_{v_e}^{(i)} + a_{ue}\zeta_e\right)$$

for

$$\hat{x}_u^{(i+1)}$$

Intuitively this systematically uses the measurements to update our estimate, which is then filtered by the covariance matrix.

Jacobi Iteration

The Jacobi Iteration method approaches the optimal estimate, scales, and is robust to temporary link failures.

Jacobi Iteration

The Jacobi Iteration method approaches the optimal estimate, scales, and is robust to temporary link failures.

However, its convergence rate is relatively slow.

Overlapping Subgraph Estimator

The Overlapping Subgraph Estimator (OSE) algorithm extends the Jacobi method.

- Instead of each node sending only their estimates of their own values, each node sends both their own estimates and the estimates received.

Overlapping Subgraph Estimator

The Overlapping Subgraph Estimator (OSE) algorithm extends the Jacobi method.

- Instead of each node sending only their estimates of their own values, each node sends both their own estimates and the estimates received.
- This essentially enables 2-hop communication.

Overlapping Subgraph Estimator

The Overlapping Subgraph Estimator (OSE) algorithm extends the Jacobi method.

- Instead of each node sending only their estimates of their own values, each node sends both their own estimates and the estimates received.
- This essentially enables 2-hop communication.
- While each node in the Jacobi method only considers its immediate neighbors, the nodes in the OSE algorithm see themselves as the center of their own 2 hop graph.

Overlapping Subgraph Estimator

The OSE algorithm follows the following steps. For a given node u,

1. Generate an arbitrary guess for the values of the 2-hop neighbors of u

Overlapping Subgraph Estimator

The OSE algorithm follows the following steps. For a given node u,

1. Generate an arbitrary guess for the values of the 2-hop neighbors of u
2. Get estimates for each node in the 1 hop neighborhood of u based on the least squares of the estimates of the 2-hop neighbors.

Overlapping Subgraph Estimator

The OSE algorithm follows the following steps. For a given node u,

1. Generate an arbitrary guess for the values of the 2-hop neighbors of u
2. Get estimates for each node in the 1 hop neighborhood of u based on the least squares of the estimates of the 2-hop neighbors.
3. Perform a weighted average between the previous value for u and the value for u generated in step 2. Broadcast this new value as well as the previously received values for the 1-hop neighbors.

Overlapping Subgraph Estimator

The OSE algorithm follows the following steps. For a given node u,

1. Generate an arbitrary guess for the values of the 2-hop neighbors of u
2. Get estimates for each node in the 1 hop neighborhood of u based on the least squares of the estimates of the 2-hop neighbors.
3. Perform a weighted average between the previous value for u and the value for u generated in step 2. Broadcast this new value as well as the previously received values for the 1-hop neighbors.
4. Listen for updates from the 1-hop neighbors and update estimates for 2-hop neighbors. Repeat steps 2–4 as needed.

Overlapping Subgraph Estimator

While the OSE algorithm as previously described examines a 2-hop radius, the radius can be made arbitrarily large. However, this requires much more communication bandwidth, with diminishing marginal benefit.

Conclusions

To compare the convergence of both the Jacobi and the OSE methods, consider the following test.

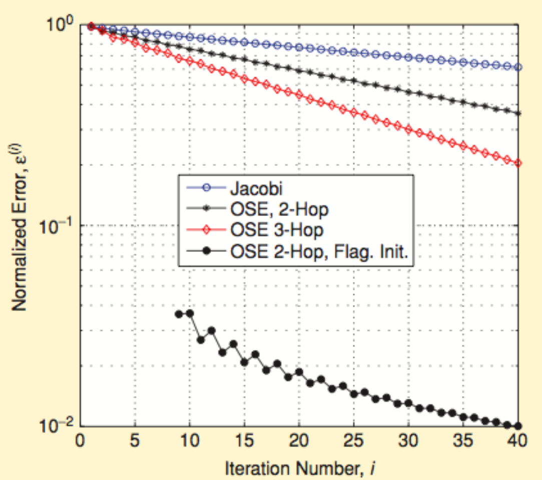
- 200 nodes separated by a distance less than 0.11

Conclusions

To compare the convergence of both the Jacobi and the OSE methods, consider the following test.

- 200 nodes separated by a distance less than 0.11
- Perform the bearing, range test mentioned previously.
- Add Guassian noise with $\sigma = 0.0165$ for radial distance and $\sigma = 10^\circ$ for bearing.

Conclusions



Questions?

References

P. Barooah and J.P. Hespanha.
Estimation on graphs from relative measurements.
IEEE Control Systems IEEE Control Syst., 27(4):5774, 2007.