

In-Network Nonparametric Loopy Belief Propagation on Sensor Networks for Ad-Hoc Localization

Jeremy Schiff and Dominic Antonelli

University of California, Berkeley
May 19, 2006

Abstract

Sensor Networks provide a cheap, unobtrusive, and easy-to-deploy method for gathering large quantities of data from an environment. While this data is often noisy, we can compensate by exploiting spatial correlation. This paper proposes the use of the statistical inference method of Loopy Belief Propagation (LBP) to exploit this correlation structure in the context of a well-examined problem in Sensor Networks: Localization. We discuss how to formulate Localization as an LBP problem, discuss the systems issues that deployments will face, and finally provide a broad spectrum of simulation results to help understand how to maximize this algorithm's effectiveness. Ultimately, we show that Loopy Belief Propagation is well suited for localization of Sensor Networks; however, there are certain limits that require further exploration.

1 Introduction

Sensor networks utilize low-cost distributed motes, each with an array of sensors, to gather and process information. As sensor network deployments involve large quantities of motes, they are limited to cheap, poor-quality sensors. To overcome these limitations, integrating data from nearby motes is vital for maximizing data robustness.

Most sensor network applications require knowledge of each mote's location as sensor readings only have meaning with respect to where this data is acquired. This need motivates the problem of ad-hoc sensor localization, where we wish to determine the motes' locations using noisy estimates of the distances between motes that are near one another.

This problem can be factored into two steps: gathering distance information, and fusing distance information into meaningful location information. There are many technologies for gathering distance information, such as radio signal strength[1] and differences in time of flight between radio and ultrasonic signals[2].

For fusion, we propose applying Nonparametric Loopy Belief Propagation (NLBP).

NLBP is a statistical method for inferring unobserved data from correlated observations. NLBP is well adapted to this context as it utilizes entire distributions and is easily parallelizable. This algorithm has been proposed by [3] as a method for sensor network localization. The algorithm uses information about the distances between motes in order to infer actual physical locations, assuming that there are a few motes with a priori knowledge of where they are located. Without this a priori knowledge, the algorithm can still infer relative locations using some arbitrary reference information. We implement this algorithm and analyze its effectiveness and efficiency in localization. We conclude that this algorithm is useful for localization but has several limitations that must be understood.

1.1 Sensor Networks

Sensor Networks is a field in which small, wirelessly communicating devices sense and reason in a collaborative fashion about the world around them. These are small devices; the modern Telos mote [4] is about 1.25x2.60x0.25 inches. This class of devices presents a new paradigm of computation, dealing with extreme constraints on power, communication, and sensing quality. Sensor networks must interact with incomplete, noisy data and often infer higher-level phenomena from this information in real-time. Sensor networks have been applied to a large range of applications such as tracking zebra movement patterns [5], or monitoring temperature variations in a forest canopy [6].

1.2 Graphical Modeling

Graphical modeling is a technique of describing correlations among random variables. One common class of graphical models are Markov Random Fields. A Markov Random Field is defined as an undirected graph where nodes represent random variables, and edges represent direct correlations between pairs of variables. Each node has a potential function, ψ_i , that

is similar to an a priori probability - if there were no edges, then the node potentials would be the a priori probabilities. Each edge, (i, j) , has a potential functions, $\psi_{i,j}$, which represents the strength of the correlation between nodes i and j .

These models are useful in many real-world applications. For example, we can define a grid of random variables (nodes) to represent temperatures throughout a room with edges between nearby nodes to represent the high spatial correlation between neighboring nodes. The node potentials represent individual node temperature information, while edge potentials represent the strength of spatial correlation. Using this model, and some temperature readings, we can perform inference to calculate posterior probabilities of each of the random variables. Each posterior describes the probability distribution of the temperature at that location, given all of the readings.

These models are an elegant way of posing the localization problem, as the random variables represent each mote's position, and the edges represent relative position information obtained from nearby motes' distance readings. Rather than performing the simplistic method of Loopy Belief Propagation, where each random variable and message is discretized to form a histogram, we apply NLBP, where each random variable and message is represented as a multiset of samples drawn according to their likelihood. This alternative formulation is superior because it increases resolution at likely locations.

2 Related Work

Two application of sensor fusion are tracking and monitoring. Two example of tracking problems are leveraging multiple classes of sensors to improve estimates of an stationary object's location [7] and a moving person's location [8]. Oh et al. [9] examined the problem of tracking multiple targets, and associating paths with the targets. Two good examples of monitoring applications are gathering temperature data across a forest's canopy [6] and monitoring a bird habitat on Great Duck Island [10].

There has been extensive research on extending the Loopy Belief Propagation algorithm. Other than the undirected discrete graphical models we use, it is applicable to Gaussian graphical models[11], directed graphical models (Bayes' Nets)[12], and many others. In addition, many have worked on theoretical guarantees about the convergence of the algorithm. [13] discusses convergence conditions for Belief Propagation on factor graphs (our undirected models are a subset of factor graph models). [14] and [15] talk about the uniqueness of fixed points found by Belief Propagation under certain conditions. Finally, [16] presents Tree-Reweighted Belief Propagation which is an extension that provides some theoretical guarantees.

There are many other formulations of ad-hoc local-

ization. K. Whitehouse presents an in-depth discussion of localization and the many approaches in his Master's Thesis [17].

Ihler et. al. [3] propose Nonparametric Belief Propagation as a method for solving the localization problem. This method involves each mote computing samples of its location distributed according to their likelihood. Each mote calculates this likelihood by integrating the information it receives about its location from all motes from which it has received distance samples. In turn, each mote uses this to provide its neighbors with an estimate of their locations. This process runs for a series of iterations where the motes share information about their estimates. This method provides no guarantees on the quality of the solution found, but in practice, it works well.

3 Inputs and Assumptions

Each mote, i , has a set, $D_{i,j}$, of readings associated with noisy distance measurements for each nearby mote, j . To compute absolute locations, we must know the locations of at least three motes, as with less, even with perfect distance readings, it is impossible to accurately localize. We also need the boundaries of potential mote locations, as we need this information to initialize the system.

There are also parameters that control our model and the tradeoff between efficiency and quality of results. Because we perform NLBP, we reason about a set of samples of size M at each of n iterations. There is an additional parameter, k , which trades accuracy for time and space.

We model how distance affects a mote's ability to obtain distance estimates from neighbors with parameter R . This is described in more detail in Section 5.

Our current formulation is restricted to two dimensions and assumes each mote has a unique identifier.

4 Outputs

Our objective is for each sensor to accurately determine its location. More formally, we want to minimize the error of Euclidean distance between the true locations and the estimated ones. Rather than a single estimate, our algorithm provides a distribution of locations, which can be used to provide additional information, such as a mote's confidence in its estimated location. If a single location value is needed from this localization service, mean, median, or mode can be used to summarize the distribution into a single estimate.

5 Description of LBP

Loopy Belief Propagation (LBP) is an iterative algorithm for computing marginal probabilities of random variables given a priori probabilities and correlation information. As we represent the graphical model for

loopy belief propagation as a graph, the a priori information for each node is given as a $\psi_i(x_i)$ function for each random variable, X_i . The correlation information is given as pairwise potential functions, $\psi_{i,j}(x_i, x_j)$, for each edge, (i, j) . The algorithm consists of computing messages from each node, X_i to each of its neighbor nodes X_j . Message, $m_{i,j}(x_j)$, from node X_i to X_j , is computed according to this equation

$$m_{i,j}(x_j) = \sum_{x_i} \left(\psi_i(x_i) \psi_{i,j}(x_i, x_j) \prod_{k \in N(i) \setminus \{j\}} m_{k,i}(x_i) \right)$$

Once all the message have been computed, marginal probabilities can be computed according to the equation

$$\Pr[x_i] = \psi_i(x_i) \prod_{k \in N(i)} m_{k,i}(x_i)$$

Note that the definition of each message is recursive. Thus, the algorithm proceeds by computing new outgoing messages when new incoming messages arrive, and computing the marginal probability once the messages have converged. When this algorithm is performed in an asynchronous setting such as a sensor network, these messages can be computed and recomputed in a somewhat arbitrary order. Though there are no guarantees about how well this asynchronous version will work, in practice, it works well, and has been proven to be schedule invariant for Gaussian graphical models[18].

6 Nonparametric Loopy Belief Propagation

LBP works well for problems with Gaussian or discrete random variables. It can be used for more general continuous distributions, but they must be parameterized or discretized. For localization, a reasonably accurate discretization yields messages that are extremely large and inefficient to compute and send. Instead, we utilize a method proposed in [3], where the algorithm uses weighted samples from the distribution as a representation of the distribution itself. $m_{i,j}$ is the message from mote i to mote j , which contains a set of weighted location estimates for mote j produced by mote i . Let $\mu_{i,j}^{(z)} \in \mathbb{R}^2$ be the z 'th sample in $m_{i,j}$, and let $w_{i,j}^{(z)} \in [0, 1]$ be the corresponding normalized weight. Likewise, let $\mu_{i,j}^{(Z)}$ be the set of all M samples in message $m_{i,j}$, and let $w_{i,j}^{(Z)}$ be the corresponding set of weights. In addition, let $\text{wcovar}_Z[w_{i,j}^{(Z)}, \mu_{i,j}^{(Z)}]$ be the weighted covariance of the samples:

$$\text{wcovar}_Z[w_{i,j}^{(Z)}, \mu_{i,j}^{(Z)}] = \sum_{z=1}^M w_{i,j}^{(z)} (\mu_{i,j}^{(z)} - \bar{\mu}_{i,j})(\mu_{i,j}^{(z)} - \bar{\mu}_{i,j})^T$$

and $\bar{\mu}_{i,j}$ is the mean of the samples in $m_{i,j}$.

In order for this message to represent a continuous distribution, the individual samples need to be blurred. [3] proposes treating the messages as a mixture of Gaussians centered at each sample point and with a single covariance, $\Sigma_{i,j}$ for all of them. We chose the following definition for $\Sigma_{i,j}$:

$$\Sigma_{i,j} = M^{-1/3} \text{wcovar}_Z[w_{i,j}^{(Z)}, \mu_{i,j}^{(Z)}]$$

The $M^{-1/3}$ term reduces the blurring when there are many samples as less blurring is necessary when the samples approximate the distribution well.

6.1 Subproblems

There are two subproblems that must be solved in order to run NLBP. We must be able to evaluate a message at a specific location, and we must be able to draw samples from a message.

6.1.1 Evaluating a Message

One important subproblem of this algorithm is given a message and a position, computing the probability of that position for this message. The probability of position x_j in message $m_{i,j}$ is

$$m_{i,j}(x_j) = \sum_{z=1}^M w_{i,j}^{(z)} \cdot f_{i,j}^{(z)}(x_j)$$

where $f_{i,j}^{(z)}$ is the two-dimensional Gaussian density function,

$$f_{i,j}^{(z)}(x_j) = \frac{\exp\left(-(x_j - \mu_{i,j}^{(z)})^T \Sigma_{i,j}^{-1} (x_j - \mu_{i,j}^{(z)})\right)}{2\pi |\Sigma_{i,j}|}$$

6.1.2 Drawing a Sample from a Message

Another important subproblem is drawing samples from the distribution defined by the message. As the message is represented as a mixture of Gaussians, we first draw a random mean $\mu_{i,j}^{(z)}$ from the message according to the weights, $w_{i,j}^{(z)}$. Then, we the sample from $\mathcal{N}(\mu_{i,j}^{(z)}, \Sigma_{i,j})$ and return this sample.

6.2 The Two Phases of NLBP

The algorithm itself consists of two phases: computing outgoing messages, and updating local marginals.

6.2.1 Computing Outgoing Messages

This phases takes as input the local marginals. These marginals are in the form of a multiset of possible positions, $X_i = \{x_i^{(1)}, \dots, x_i^{(M)}\}$. From this, we compute the messages, $m_{i,j}$ for each neighbor $j \in N(i)$ as follows. For each possible position $x_i^{(z)} \in X_i$, choose a random angle, θ , and draw a distance sample, d from

the distribution, $D_{i,j}$, of distances between i and j . Set

$$\mu_{i,j}^{(z)} = x_i^{(z)} + d \cdot [\sin \theta, \cos \theta]$$

and

$$w_{i,j}^{(z)} = \frac{\exp\{-d^4/(2R^4)\}}{m'_{j,i}(x_i^{(z)})}$$

where $m'_{j,i}$ is the message from j to i from the previous iteration.

6.2.2 Updating Local Marginals

Upon receiving messages, $m_{i,j}$ from each neighbor (or using old messages if a new one hasn't arrived yet), mote j can compute $\Sigma_{i,j}$ for each message as described above. Then, we draw $\frac{kM}{|N(j)|}$ samples from each incoming message. This yields a total of kM samples. Each sample, s , is weighted with $\prod_{v \in N(j)} m_{v,j}(s)$, and we redraw M of these samples (with replacement) according to their weights. These M samples make up X_j .

6.3 Utilizing the Analytic Function

Other than the distance measurement information, additional information can be inferred from the lack of distance measurements. If nodes i and j , located at x_i and x_j respectively, do not have distance measures from each other, then it is likely they are far apart. We model the probability that two nodes have distance measures as

$$\exp\left\{-\frac{\|x_i - x_j\|^4}{2R^4}\right\}$$

. In order to take advantage of this, pairs of nodes that do not have distance measures need to share their marginals with each other. There is additional cost in this, as multi-hop messages require multiple sends. The additional information from this is taken into account in the weighting of samples in the local marginal. Let $A(j)$ represent the set of nodes for which j has this additional information. When weighting a sample, s , drawn from the standard messages, the weight is now

$$\prod_{v \in N(j)} m_{v,j}(s) \prod_{u \in A(j)} f_a(s, X_u)$$

where f_a is the analytic function

$$f_a(s, X_u) = 1 - \sum_{z=1}^M \exp\left\{-\frac{\|x_u^{(z)} - s\|^4}{2R^4}\right\}$$

This function models the probability that these two nodes cannot obtain distance measures from each other given the estimated locations of one of the nodes, and a proposed location for the other. By multiplying this into the weights, we make it less likely that these two nodes are close to each other. Because they do

not have a distance measure with each other, they are less likely to be within R units of each other, and this function captures that.

In practice, using this information from nodes that are far in the network topology is prohibitively expensive in terms of communication cost and provide increasingly negligible benefit. It may be feasible to obtain this information from nodes that are two hops away in the network graph, which motivates us to analyze the benefits of using the analytic function as can be seen in Section 8.6.

6.4 Runtime Analysis

The runtime for each individual mote is bounded by the speed that the marginal can be computed. This takes $O(cnkM^2)$, where c is the number of neighbors that mote has, and n , k , and M are as described in Section 3. Each mote computes kM samples, and weighting each sample requires evaluating the message at that sample in each of the c incoming messages. This takes $O(M)$ time since it requires taking a sum over the M samples in the message, computing the 2D Gaussian density which takes $O(1)$ time, and multiplying by the weight.

7 Description of Implementation

To implement such a service, we propose a four stage algorithm.

1. Gather distance estimates from nearby motes
2. Estimate link quality
3. Broadcast neighbor lists
4. Perform inference to determine location.

7.1 Gather Distance Estimates

In this phase of the algorithm, all motes use their radios, ultrasonic transceivers, or other sensors to estimate their distances to nearby motes.

7.2 Estimate Link Quality

In order to determine over which links we should perform inference, we need to know which links we can reliably communicate over. We can use any link quality protocol, however if our localization protocol involves a metric for computing how likely a mote can hear from each neighbor, we can piggy-back the link quality estimate on the localization estimates and merge phases one and two. For example, if we estimate distances from the signal strength of a number of messages sent, then we can determine if the link is of sufficient quality for performing inference if the mote receives at least a certain percentage of these messages.

7.3 Broadcast Neighbor Lists

Next, all motes broadcast their neighbor table. While Ihler et. al [3] suggests using the neighbors of neighbors as a mechanism for pruning out potential impossible location of nodes during inference, knowing this information is useful for pruning asymmetric links where mote A can communicate to mote B, but not vice versa. As the neighbor table functions implicitly as an acknowledgement, we can use this information to restrict ourselves to good neighbors for inference. Also, these neighbor lists can be used to route the analytic function information discussed in Section 6.3.

7.4 Perform Inference

Once all nodes know their one-hop and two-hop neighbors and have determined which links are of sufficient quality, we run Nonparametric Belief Propagation as is described in Section 5.

7.5 Implementation Details

There are number of issues that arise in the algorithm that aren't discussed in any of the previous work. A significant problem we encountered is that a mote would get messages that completely disagree. In one message, location z has a significant probability while location z' has near-zero probability; however, in another message, z has near-zero probability and z' has significant probability. If two messages do not agree on a single location, then there is no way to ever get a sample that does not have near-zero probability. This becomes a problem as we only have a finite number of bits to represent the probability. When we detect this, by noticing that a large number of positions with zero probability have been generated, we double the values in every messages sigma matrix and regenerate new samples. In all cases we ran into, performing this doubling one to three times was enough to get sufficient consistency.

This is an inherent flaw in particle based methods. Since we only have a finite number of particles from each distribution, there is a chance, which decreases as the number of particles increases, that a portion of the distribution with significant probability will end up with no particles. On the graphs we tried, using $M = 50$ was sufficient to prevent rampant inconsistency, though it would still appear occasionally.

8 Experimental Results

We wrote a Java simulator to see how the many different parameters affect the system's performance. We experimented with tradeoffs in the algorithm which either involved time vs. accuracy or power vs. accuracy. Our simulations are all over a 40X40 grid, with 50 sensors. In order to compare different parameters,

we limit both the number of particles M , and the number of iterations n , to be 50 each for most tests and use $k = 5$.

8.1 Time Evolution

In order to get an intuition for how the algorithm works, we present a sequence of "snapshots" of the belief of each mote as the algorithm progresses. Figure 13 at the end of this document provides a view of the actual and believed positions of each mote at different iterations of the algorithm.

Iteration 1 shows the random initialization. Because we initialize each mote with multiple uniformly chosen random points as the M samples in the local marginal (X_i), we see that each mote believes it is somewhere near the middle of the space; though this belief has a very high variance. As the algorithm progresses, we see many of the motes' beliefs converge to very near the correct beliefs by Iteration 30. Also, by then, the motes that are significantly incorrect are all consistent with one another but slightly rotated from their absolute positions. As we increase the number of iterations, they slowly rotate towards the correct answer.

8.2 Varying Connectivity

In this experiment, we modeled distance readings with a Gaussian of mean as the actual distance between particles, and standard deviation as the mean divided by twenty. We do this to model the error lessening for closer distances. We vary R , the distance that motes can get distance estimates, over values 10.0, 12.5, 15.0, and 17.5. We ran the inference for $M = 50$, used median as the function for computing locations from our distributions and additionally used the two-hop analytic edge information.

An example of the setup for 12.5 is shown in Figure 1. By running four different initializations, we observe the average cumulative error illustrated in Figure 2. As expected, generally adding more connectivity helps, but eventually yields only marginal benefits, and is not worth the extensive extra computation needed to compute messages to and from many extra neighbors. It is interesting that $R = 10$ performed better on average than $R = 12.5$, which shows that while good communication is necessary, a bad initialization can drastically harm the quality of localization.

8.3 Constant Computation

Our next test was holding computation constant. While this is generally not a primary concern of sensor networks, as inference is a particularly computationally intensive procedure, it is important to understand the tradeoff between the number of particles and the number of iterations, keeping total computation constant. Keeping all other parameters constant, as each

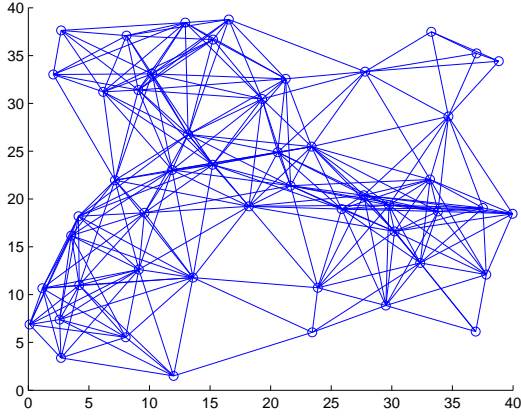


Figure 1: An example of graph connectivity for $R = 12.5$

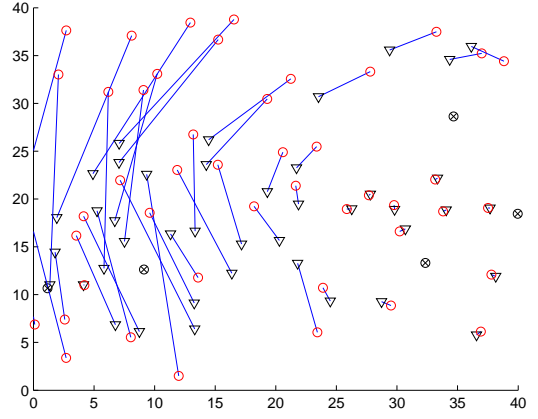


Figure 3: Constant Computation: $M = 100$ and $n = 13$, localizes some nodes very well, but does not converge to the correct answer

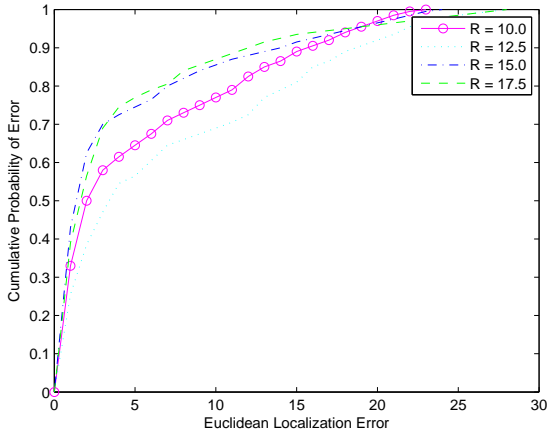


Figure 2: The change in localization as we increase the range of distance measures between nodes by varying R .

iteration takes M^2 steps, doubling the number of samples means we divide the number of iterations by four. Conversely, dividing the number of samples by two means we can quadruple the number of iterations. In Figure 3, we use a high number of samples, and a low number of iterations ($M = 100$ and $n = 13$). In Figure 4 we use an equal number of iterations and samples ($M = 50$ and $n = 50$). Finally, in Figure 5, we use a low number of samples, but a high number of iterations ($M = 25$ and $n = 200$). From these graphs, it can be seen that more iterations clearly are better. We also observed over other simulations that it is worth sacrificing samples in order to increase the number of iterations, as it can take a long time for information to propagate across the network.

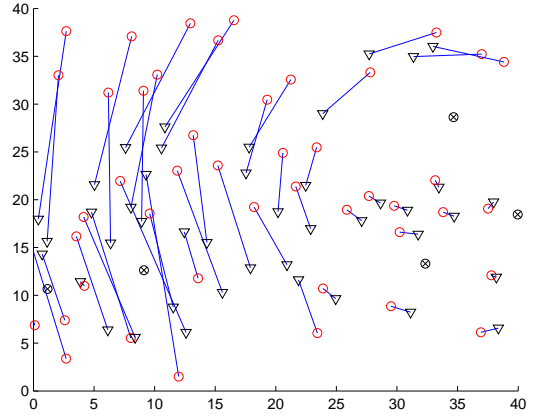


Figure 4: Constant Computation: $M = 50$ and $n = 50$, localizes some nodes very well, but does not converge to the correct answer

8.4 Constant Communication

In this test, we explore varying the number of particles M , and the number of iterations n , while keeping the total communication cost the same. In each of the n iterations, each node sends a message of length M to each neighbor, so the overall communication cost, in terms of these two parameters, is $O(Mn)$. Because of this, we tried three configurations of these parameters: $M = 100$ and $n = 25$, $M = 50$ and $n = 50$, and $M = 25$ and $n = 100$. Figures 6, 7, and 8 show results for these configurations for 10 runs on random graphs.

Comparing the different results we see that the $M = 50, n = 50$ results show a very high variance, but the mean is about the same as for the lower variance $M = 100, n = 25$ results. However, the $M = 25, n = 100$ show clearly superior results. Of

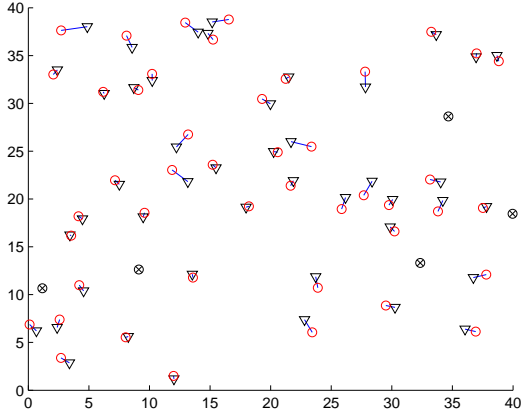


Figure 5: Constant Computation: $M = 25$ and $n = 200$, localizes all nodes in the graph effectively

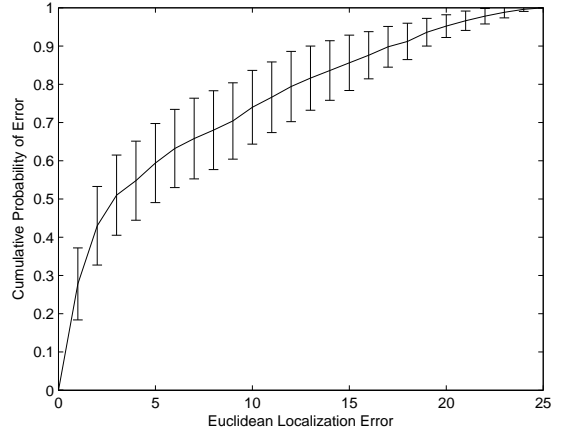


Figure 7: This plot shows the cumulative distribution of the error for $M = 50$ and $n = 50$. Results show the mean as well as 95% confidence intervals.

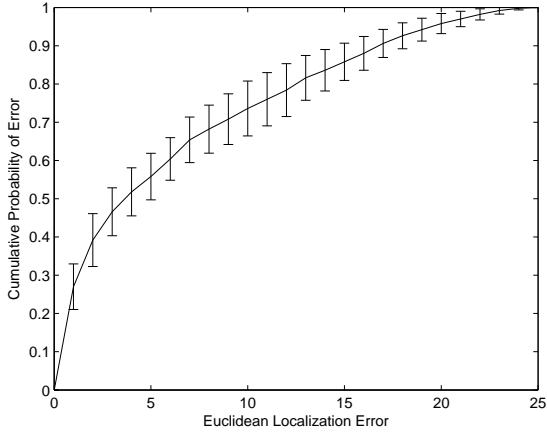


Figure 6: This plot shows the cumulative distribution of the error for $M = 100$ and $n = 25$. Results show the mean as well as 95% confidence intervals.

course, with the large variance of these results, this is by no means conclusive, but this is significant evidence. In addition, the $M = 25, n = 100$ case also requires less computation, providing more motivation for using higher numbers of iterations and fewer particles.

8.5 Modifying Number of Samples

This test illustrates how the number of distance samples can affect localization quality. More samples will assist in better approximating a distribution, but should only help up to a point. We present these results in Figure 9. Each sensor gets a set of readings from each other mote it can communicate with, ranging over 1, 8, 64, 512, and 4096 samples. At each iteration, the sensor randomly chooses the samples it will use uniformly from this set fixed set. We also

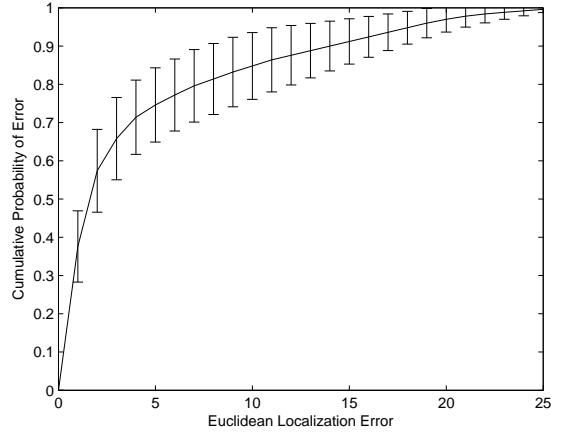


Figure 8: This plot shows the cumulative distribution of the error for $M = 25$ and $n = 100$. Results show the mean as well as 95% confidence intervals.

use a Gaussian model, where it samples from the error distribution directly, thus it has an infinite number of samples to draw from. We see that the Gaussian is optimal as we expect, and generally there is an increase in the ability to localize as a function of the number of samples. However, 8 samples localizes very well. This could be an artifact of the initial distribution that the NLBP converges to, but it is likely that this means that very few samples are necessary to sufficiently describe the distribution. It is also interesting that increasing the number of samples lowers the standard deviation as can be seen in Figure 10. However, the Gaussian seems to ignore this trend, which is contrary to intuition.

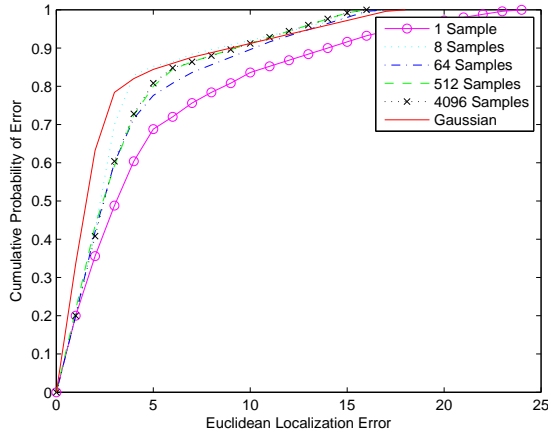


Figure 9: The effect of cumulative probability of error of our localization as a function of number of samples.

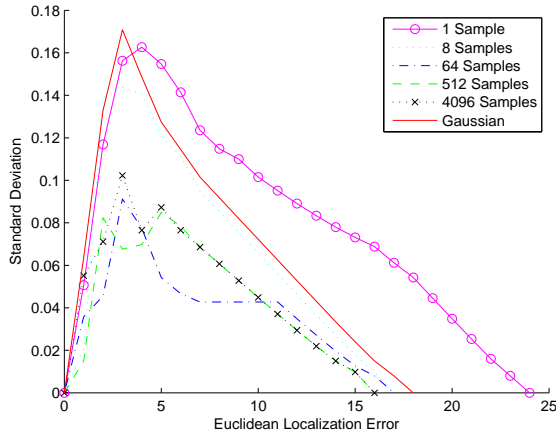


Figure 10: The effect of the standard deviation of our localization as a function of number of samples.

8.6 Effect of the Analytic Function

In this test we analyze whether using the two-hop analytic function information provides a significant improvement in the localization results. We generate five random graphs and run our algorithm both with and without analytic function information for each graph. Figure 11 shows the means of the cumulative distribution of the error for these tests. We see that the analytic information seems to provide a small benefit. However, the variance of these results (not pictured) is fairly high, and the runs without the analytic information often performed better than those with it. It is highly unlikely that using the analytic information is worth the enormous cost in additional communication (because each mote has a large number of two-hop neighbors). However, if a run of our localization algorithm yields a few motes with high localization error, one could try rerunning with this information,

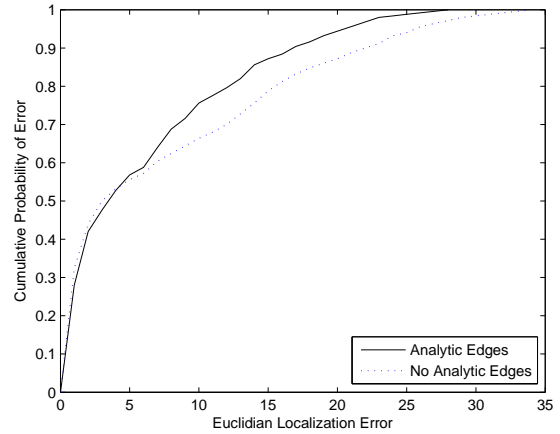


Figure 11: Mean of the cumulative distribution of error for 5 runs with and without analytic function information

or just using the analytic information once as a post-processing step to try to narrow down where that mote is.

8.7 Modifying Standard Deviations of Readings

This test analyzes the algorithm’s robustness to noisy distance readings. We ran our algorithm for a random graph where we modeled error as a Gaussian where the mean was the actual distance, and the standard deviation was a percentage of the distance, which ranged from 2.5% to 50%. We modeled standard deviation in this way so that motes that are further apart have higher error. Figure 12 presents our results when we provide 1024 noisy samples of each neighbor’s distance to each mote. This graph illustrates that the algorithm is resilient to significant distance reading noise. Even with a standard deviation of 50% of the distance, the motes localize well.

9 Algorithm Behavior Discussion

The above plots are just a small subsample of the most interesting results from the experiments we ran. Our extensive evaluation elucidated many nuances of the algorithm. First, it is important to note that even standard LBP has no convergence guarantees. It is an approximation to the exact junction tree algorithm. Unfortunately, because junction tree is exponential in the clique sizes of the graph, and we need many neighbors to perform localization effectively, it is computationally infeasible to perform this algorithm. It has been investigated as a good framework for problems with smaller clique sizes [19]. While LBP is still not well understood, there has been extensive effort to understand why it works [14, 15], as well as ways to provide better guarantees. One such algorithm

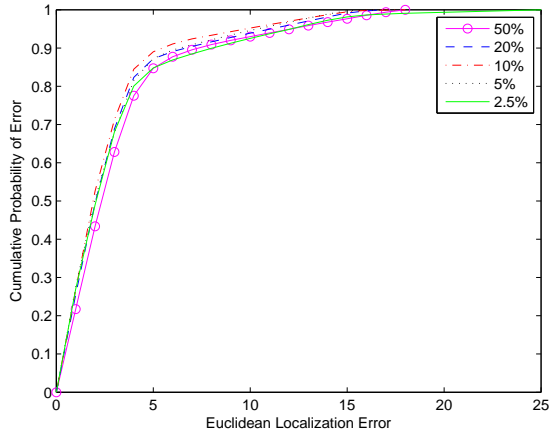


Figure 12: Mean of the cumulative distribution of error for 30 runs using 1024 samples of the distance with standard deviation varying from 2.5% to 50% of the mean

for improved guarantees is Wainwright et al.’s Tree-Reweighted Loopy Belief Propagation [16]. While normal LBP has no formal guarantees, the applications of these algorithms have been very successful. For instance, they are used in most cell phones for decoding purposes. We are also confident that better approximations that do provide formal guarantees will be available in the future. NLBP is a further approximation of LBP because it uses samples from distributions, rather than the distributions themselves.

In terms of accurately localizing a sensor network, as we show in Section 8, there are many different parameters that might affect accuracy. The most influential factor seems to be the topology of the sensor network. If there are bottlenecks, where information from known nodes is sufficiently restricted to no longer provide three known nodes indirectly, it is impossible for any algorithm to correctly localize. This includes bottlenecks that hide indirect information, such as our analytic functions.

The next most influential property appears to be initialization. Even on the same graph, with the same distribution, an unlucky initialization can dramatically alter results. This can be the difference between obtaining good estimates within 10 iterations and needing more than 200 iterations for good estimates. This is because the inference can first converge to the wrong result, a local minima of some sort, and hopefully move toward the correct solution. The time evolution plot illustrates this problem very effectively in Section 8.1.

10 Future Work

10.1 Reducing Communication

Another interesting problem would be to further reduce the amount of communication necessary in our

implementation. One potential solution is to implement the method described in [20]. Here, the authors describe a modification to the basic Loopy Belief Propagation algorithm that allows each node to broadcast a single message to all of its neighbors instead of one unique message to each of its neighbors. This broadcast message is the same size as the individual message. In wireless sensor networks, at least, this broadcast costs the same as a neighbor-to-neighbor message, so this provides a significant savings in communication.

This method directly applies to Nonparametric Belief Propagation. Each mote can broadcast its local marginals, and each neighbor can compute the messages themselves from this. Everything that the broadcasting mote would have used to compute individual messages is available to the receiver: the distance estimate, the previous message from the receiver to the sender, and the local marginals that were broadcast. In practice, however, there are complications that arrive when motes fail to receive some messages. The receiver may have sent more recent messages (broadcasts) to the sender that the sender did not receive, and thus will use the incorrect “previous message”. This can be handled using version vectors and storing the last several outgoing messages, but this extra state and complexity add up quickly. If no motes actually go down for some time and come back up, then this will work fine as is. If motes do fail for a significant period, then one might have to send a standard message to catch that node back up. It would be interesting to analyze the tradeoffs in using this method.

An interesting problem that this research prompted is given perfect distance readings, how must the radius of communication scale as a function of the number of nodes within a constant area to be able to bound, with constant probability, the ability to solve the localization problem given three known nodes? It seems that this property would help a lot with the intuition about what values of R are reasonable for a specific graph.

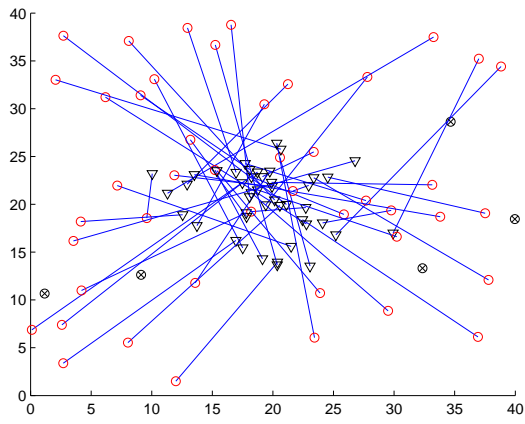
One way to alleviate our problems with unlucky initializations is to have all nodes ignore any neighbors that it has not received messages from, instead of just using randomly initialized messages. In this case, reasonable initial values will propagate outward from the anchor nodes. We suspect that this may reduce the variance in our experiments as well as produce optimal solutions more frequently. We would like to explore this more.

11 Acknowledgment

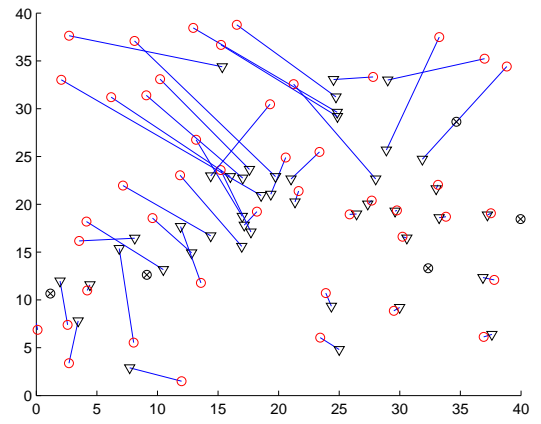
We would like to thank Alex Dimakis and David Chu for their assistance in formulating this problem. We would also like to thank professor Martin Wainwright for his initial inspiration of adapting Loopy Belief Propagation to sensor networks.

References

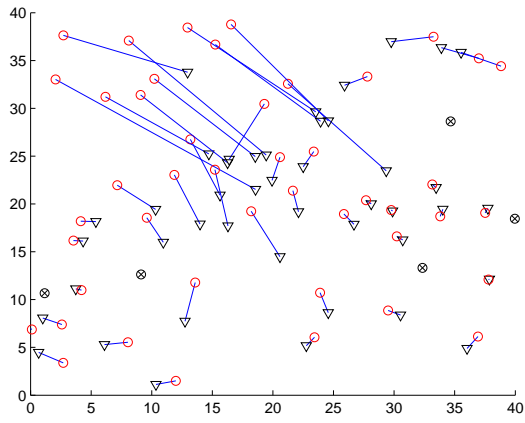
- [1] A Localization System using Wireless Network Sensors: A Comparison of Two Techniques. Mark terwilliger, ajay gupta, vijay bhuse, zille huma kamal, and mohammad ali salahuddin. In *The Proceedings of the First Workshop on Positioning, Navigation and Communication, Hannover, Germany, March 2004*.
- [2] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location support system. *Proceedings of the 6th Annual ACM International conference on Mobile Computing and Networking*, August 2000.
- [3] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky. Nonparametric belief propagation for self-localization in sensor networks. *IEEE Journal of Selected Areas in Communication*, 2005.
- [4] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS), April 2005*.
- [5] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2004*.
- [6] G. Tolle et. al. A macroscope in the redwoods. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2005*.
- [7] D. Cochran, D. Sinno, and A. Clausen. Source detection and localization using a multi-mode detector: A bayesian approach. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Phoenix, March 1999*.
- [8] Shawn R. Jeffery, Gustavo Alonso, Michael J. Franklin, Wei Hong, and Jennifer Widom. Virtual devices: An extensible architecture for bridging the physical-digital divide. Technical Report UCB/CSD-05-1375, EECS Department, University of California, Berkeley, 2005.
- [9] Songhwai Oh, Inseok Hwang, Kaushik Roy, and Shankar Sastry. A fully automated distributed multiple-target tracking and identity management algorithm. In *Proc. of the AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, August 2005*.
- [10] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, and David Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2004*.
- [11] K. Plarre and P. Kumar. Extended message passing algorithm for inference in loopy gaussian graphical models, 2002.
- [12] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in AI*, pages 467–475, 1999.
- [13] J M Mooij and H J Kappen. Sufficient conditions for convergence of loopy belief propagation, 2005.
- [14] Tom Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Comput.*, 16(11):2379–2413, 2004.
- [15] T. Heskes. Stable fixed points of loopy belief propagation are minima of the bethe free energy. In *Proceedings of Advances in Neural Information Processing Systems, volume 15, Vancouver, CA, 2003*.
- [16] M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation via pseudo-moment matching. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.
- [17] K. Whitehouse. Mastering thesis. Master’s thesis, UC Berkeley University, 2002.
- [18] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, October 2001.
- [19] Mark Paskin, Carlos Guestrin, and Jim McFadden. A robust architecture for distributed inference in sensor networks. In *Fourth International Conference on Information Processing in Sensor Networks (IPSN’05)*, 2005.
- [20] Danny Bickson, Danny Dolev, and Yair Weiss. Modified belief propagation algorithm for energy savings in wireless and sensor networks. Technical Report TR-2005-85, School of Computer Science and Engineering, The Hebrew University, 2005.



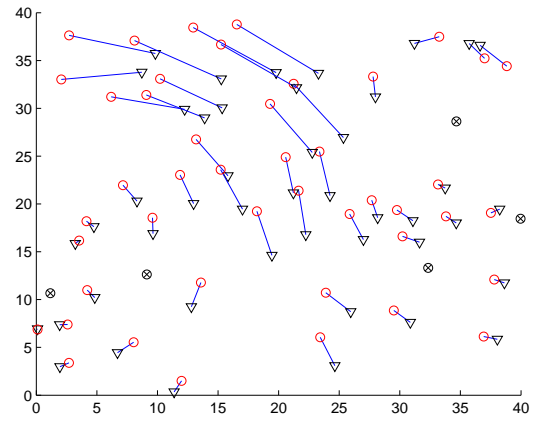
Iteration 1



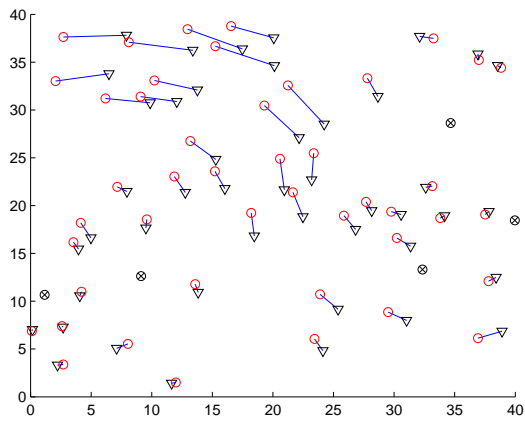
Iteration 10



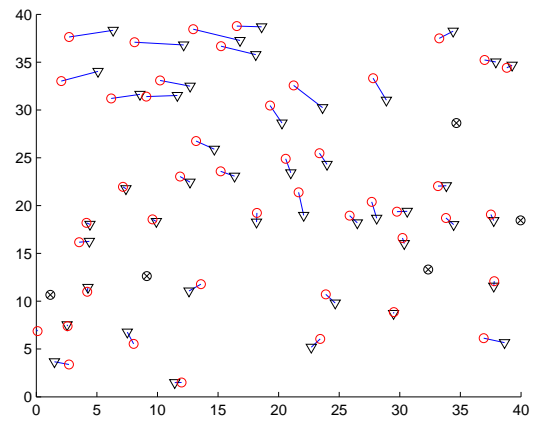
Iteration 20



Iteration 30



Iteration 40



Iteration 50

Figure 13: This illustrates the localization estimates, where circles represent true locations, and triangles represent estimated locations.