Nonparametric Belief Propagation Based on Spanning Trees for Cooperative Localization in Wireless Sensor Networks

Vladimir Savic and Santiago Zazo

Post Print

N.B.: When citing this work, cite the original article.

©2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

http://dx.doi.org/10.1109/VETECF.2010.5594105

Postprint available at: Linköping University Electronic Press
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-81082
Nonparametric Belief Propagation based on Spanning Trees for Cooperative Localization in Wireless Sensor Networks

Vladimir Savic and Santiago Zazo
Signal Processing Application Group, Polytechnic University of Madrid
Avda. Complutense 30, 28040 Madrid, Spain
Email: {vladimir, santiago}@gaps.ssr.upm.es

Abstract—Nonparametric belief propagation (NBP) is one of the best-known methods for cooperative localization in sensor networks. It is capable of providing information about location estimation with appropriate uncertainty and to accommodate non-Gaussian distance measurement errors. However, the accuracy of NBP is questionable in loopy networks. In this paper, we propose a novel approach, NBP based on spanning trees (NBP-ST) created by breadth first search (BFS) method. According to our simulation results, NBP-ST performs better than NBP in terms of accuracy, computational and communication cost in the networks with high connectivity (i.e., highly loopy networks).

I. INTRODUCTION

The belief propagation (BP) algorithm [1] is a way of organizing the “global” computation of marginal beliefs in terms of smaller local computations within the graph. It is one of the best-known message-passing method for distributed inference in statistical physics, artificial intelligence, computer vision, localization, etc. The whole computation takes a time proportional to the number of links in the graph, which is dramatically less than the exponentially large time that would be required to compute marginal probabilities naively.

Due to the presence of nonlinear relationships and highly non-Gaussian uncertainties the standard BP is undesirable. In addition, to obtain acceptable spatial resolution for the nodes, the discrete space (grid) in the deployment area must be made too large for BP to be computationally feasible. However, using particle-based representations via nonparametric belief propagation (NBP) [2] enables the application of BP to inference in sensor networks. The main advantages of this approach are i) its easy implementation in a distributed fashion thanks to the message-passing method, and ii) sufficiency of a small number of iterations (e.g., 3-5) to converge. Furthermore, NBP is capable of providing information about location estimation uncertainties and accommodating non-Gaussian distance measurement errors. This is the main advantage of NBP comparing with well-known deterministic methods, e.g., [3]–[5].

However, BP convergence is not guaranteed in a network with loops [1], [6] or even with convergence, it could provide us less accurate estimates. Regarding localization using NBP, there is no significant convergence problem, but the accuracy is questionable. In the current state of the art, there are few solutions for networks with loops, but mostly they have not been used for the localization. Well-known solutions based on generalized belief propagation (GBP) [6]–[9], which are based on clusters or cliques, are still very complex for the large-scale networks. Another option, tree-based reparameterization (TRP) [10] is the method based on the message-free version of BP which requires formation of two-node trees and then merging them via an update rule. The problem with this method is the lack of the nonparametric representation. Nevertheless, the idea for our method comes from the TRP, but the core of our method is standard NBP method (i.e., message-passing version) based on optimal spanning tree formation.

Therefore, in this paper, we propose NBP based on spanning trees (NBP-ST) created by breadth first search (BFS) method [11] which is optimal for the unweighted graphs. NBP-ST algorithm represents two (or more) independent runnings of the NBP algorithm based on formed spanning trees. According to our simulation results, NBP-ST performs better than NBP in terms of accuracy, computational and communication cost in the networks with high connectivity (i.e., highly loopy networks). Furthermore, the computational and communication cost have the smaller curve slope with respect to the transmission radius. However, the drawback of proposed method is a poor performance in low connected networks. Anyway, this is not a problem since for the low-connected networks we can keep using NBP.

The remainder of this paper is organized as follows. In Section II, we review NBP method for cooperative localization in sensor networks. In Section III, we propose NBP method based on spanning trees. Simulation results are presented in Section IV. Finally, Section V provides some conclusions and future work perspective.

II. NONPARAMETRIC BELIEF PROPAGATION FOR COOPERATIVE LOCALIZATION IN WSN

We start with the description of the statistical framework for cooperative localization using BP/NBP proposed by Ihler et al. [2]. We consider the problem in which some small number of sensors, called anchor nodes, obtain their coordinates via GPS or by installing them at points with known coordinates, and the rest, unknown nodes, must determine their own coordinates. We suppose that all sensors with unknown positions obtain noisy distance measurements of nearby subset of the other sensors in the network. Typically, this measurement procedure
can be accomplished using a broadcast transmission from each sensor as all other sensors listen. This, multi-hop, technique allows us to use low-power sensors with lack the energy necessary for long-range communication.

Let us assume that we have $N_s$ sensors ($N_a$ anchors and $N_u$ unknowns) scattered randomly in a planar region, and denote the two-dimensional (2D) location of sensor $t$ by $x_t$. The unknown node $u$ obtains a noisy measurement $d_{tu}$ of its distance from node $t$ with some probability $P_d(x_t, x_u)$:

$$d_{tu} = \|x_t - x_u\| + v_{tu}, \quad v_{tu} \sim p_v(x_t, x_u)$$

(1)

where, for the noise $v_{tu}$, we can use the log-normal (for received-signal-strength - RSS) or Gaussian (for time-of-arrival - TOA) distribution ($p_v$). However, it is advisable to use real measurements done in appropriate deployment area. In that case it is not necessary to use any parametric form of the error distribution.

The binary variable $o_{tu}$ will indicate whether this observation is available ($o_{tu} = 1$) or not ($o_{tu} = 0$). Finally, each sensor $t$ has some prior distribution denoted $p_t(x_t)$. This prior could be an uninformative one for the unknowns and the Dirac impulse for the anchors. Then, the joint distribution is given by:

$$p(x_1, ..., x_{N_u}, \{o_{tu}\}, \{d_{tu}\}) = \prod_{(t,u)} p(o_{tu} | x_t, x_u) \prod_{(t,u)} p(d_{tu} | x_t, x_u) \prod_t p_t(x_t)$$

(2)

For large-scale sensor networks, it is reasonable to assume that only a subset of pairwise distances will be available, primarily between sensors which are located within some radius $R$. This is an ideal model of probability of detection. However, in this paper, we use the exponential model, which represents a good approximation of the real-world systems:

$$P_d(x_t, x_u) = \exp \left(-\frac{1}{2} \|x_t - x_u\|^2 / R^2 \right)$$

(3)

In addition, it is necessary to exchange information between the nodes which are not directly connected (n-step neighbors; $n > 1$) because they contain some information about the distance between them. Therefore, if two nodes do not observe the distance between them, they should be far away from each other. In our case, we will include all 1-step and 2-step neighbors, others could be neglected without losing accuracy of the results.

The relationship between the graph and joint distribution may be quantified in terms of potential functions $\psi$ which are defined over each of the graph’s cliques. A clique $(C)$ is a subset of nodes such that for every two nodes in $C$, there exists an edge connecting the two. So the joint distribution is given by:

$$p(x_1, ..., x_{N_u}) \propto \prod_{\text{cliques } C} \psi_C(\{x_i : i \in C\})$$

(4)

We can now define potential functions which can express this joint posterior distribution. This only requires potential functions defined over variables associated with single nodes and pairs of nodes. Single-node potential (prior information about position) at each node $t$, and the pairwise potential (probabilistic information about distance) between nodes $t$ and $u$, are respectively given by:

$$\psi_t(x_t) = p_t(x_t),$$

(5)

$$\psi_{tu}(x_t, x_u) = \begin{cases} P_d(x_t, x_u) p_v(d_{tu} - \|x_t - x_u\|), & \text{if } o_{tu} = 1, \\ 1 - P_d(x_t, x_u), & \text{otherwise.} \end{cases}$$

(6)

Finally, the joint posterior distribution is given by:

$$p(x_1, ..., x_{N_u} | \{o_{tu}, d_{tu}\}) \propto \prod_t \psi_t(x_t) \prod_{t,u} \psi_{tu}(x_t, x_u)$$

(7)

Having defined a statistical framework for sensor localization, we can now estimate the sensor locations by applying the BP algorithm. We apply BP to estimate each sensor’s posterior marginal, and use the minimum mean square error estimate (MMSE) (i.e., mean value) of this marginal and its associated uncertainty to characterize the sensor positions.

Each node $t$ computes its belief $M_t^i(x_t)$, the posterior marginal distribution of 2D position $x_t$ at iteration $i$, by taking a product of its local potential $\psi_t$ with the messages from its set of neighbors $G_t$:

$$M_t^i(x_t) \propto \psi_t(x_t) \prod_{u \in G_t} m_{ut}^i(x_t)$$

(8)

The messages $m_{ut}$, from node $u$ to node $t$, are computed by:

$$m_{ut}^i(x_t) \propto \sum_{x_u} \psi_{ut}(x_t, x_u) \frac{M_{ut}^{i-1}(x_u)}{m_{tu}^{i-1}(x_u)}$$

(9)

In the first iteration of this algorithm it is necessary to initialize $m_{ut}^1 = 1$ and $M_t^1 = p_t$ for all $u, t$, and then repeat computation using (8) and (9) until sufficiently converge. As convergence parameter, we use the KL divergence [12]. However, even without computing the KL divergence, we know that it is sufficient to run until all unknown nodes obtain information from minimum 3 non-colinear anchor nodes.

Since we use NBP, the belief and message update equations, (8) and (9), are performed using stochastic approximations, in two stages: first, drawing $N$ weighted particles ($\{W^{t,i}_t, X^{t,i}_t\}$) from the belief $M_t^i(x_t)$, then using these particles to approximate each outgoing message $m_{tu}^i$. The detailed description of the algorithm can be found in [2]. In addition, using the results from our previous paper [12], we constrain the area from which the particles are drawn by building a box that covers the region where anchors ranges overlap, and, in each iteration, we filter out all particles which get out of the appropriate box. This modification increases the accuracy even after very small number of iterations (i.e., < 5).

III. SPANNING TREE FORMATION

If we ignore the existence of loops and permit the nodes to continue communicating with each other, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium, or even if it converges,
it could provide us less accurate estimates. That is the main reason why we are going to implement NBP based on spanning trees (NBP-ST).

We start by describing the basics of graphical models. An undirected graph \( G = (V, E) \) consists of a set of nodes or vertices \( V \) that are joined by a set of edges \( E \). A loop or cycle is a sequence of distinct edges forming a path from a node back to itself. A tree is a connected graph without any loop. A spanning tree is an acyclic subgraph that connects all the nodes of the original graph. A root node is a node without parent and leaf node is a node without children. In order to define an undirected graphical model, we place at each node a random variable \( x \) taking values in some space. In case of localization, this random variable represents the 2D position and each edge represents the measured distance. If we exclude anchor nodes, the graph is obviously undirected, but only for the first phase (spanning tree formation) we assume that it is directed (starting from chosen root node).

The optimal method for spanning tree formation for unweighted graphs is breadth first search (BFS). It begins at the root node and explores all the neighboring nodes. Then each of those neighbors explores their unexplored neighbor nodes, and so on, until all nodes are explored. In this way, there will be a path in the graph and all nodes will be explored just once. The detailed pseudocode is shown in Alg. 1. The worst case complexity is \( O(v + e) \), where \( v \) is the number of nodes and \( e \) is the number of edges in the graph, since every node and every edge will be explored in the worst case.

In case of NBP localization, we exclude all the anchors from the BFS algorithm since they do not form the loops in the graph (they just send, and never receive the messages). A graph generally has a large number of spanning trees, but since our graph is unweighted we choose few (minimum 2) of them in a partly random way. The first root node we choose randomly from the set of all unknown nodes. In order to maximize the difference between two spanning trees, the second root node has to be as far as possible from the first root node. Thus, it should be one of the leaf nodes which is the maximum-hop away from the root. If we want to form more spanning trees, the analog constraint will be used. An example of a loopy graph and two corresponding spanning trees, formed by BFS with mentioned constraints, are illustrated in Figure 1. Note that, using BFS, it is not possible to form two spanning trees with completely different edges and that usually some of the edges will be out of both spanning trees. If we want to include all edges, we have to add more spanning trees but it is usually not necessary since it will only provide us a redundant information. It is especially the case in the networks with high connectivity.

The NBP method is naturally distributed through the graph which means that there is no central unit which will handle all computations. Therefore, the proposed BFS method has to be done in a distributed way. This can be simply done if each unknown node initially broadcast its \( ID \) to all neighbors, which will continue to broadcast to others, and so on, until each unknown node aggregates a list of all unknown nodes in the graph. One node (e.g. with lowest \( ID \)) has to be assigned to choose the root node from the list and give him a permission (by multihop broadcasting) to start BFS algorithm. Then, the chosen root node has all initial data to start BFS algorithm, and, when it is necessary, has only to broadcast all data (i.e. variables from Alg. 1) to all its neighbors.

Finally, NBP-ST algorithm represents two (or more) independent runnings of the NBP algorithm based on formed spanning trees. Each running will provide us weighted particles of the node beliefs computed by (8). The simplest way to fuse these beliefs is to resample with replacement [13] from weighted particles from all spanning trees, which produces the particles with same weights. The collection of particles from all spanning trees represents our final output, from which we can easily extract any parameter that we need (e.g., mean value for location estimate). The pseudocode in Alg. 2 illustrates NBP-ST method.

IV. SIMULATION RESULTS

In the simulations using Matlab, we placed 100 unknowns and 10 anchor nodes randomly in 20m x 20m area. Since the unknown nodes near the edges of deployment area suffer from low connectivity, we include one realistic constraint: four unknown nodes and each edge represents the measured distance. If we exclude anchor nodes, the graph is obviously undirected, but only for the first phase (spanning tree formation) we assume that it is directed (starting from chosen root node).

The optimal method for spanning tree formation for unweighted graphs is breadth first search (BFS). It begins at the root node and explores all the neighboring nodes. Then each of those neighbors explores their unexplored neighbor nodes, and so on, until all nodes are explored. In this way, there will not be a loop in the graph because all nodes will be explored just once. The detailed pseudocode is shown in Alg. 1. The worst case complexity is \( O(v + e) \), where \( v \) is the number of nodes and \( e \) is the number of edges in the graph, since every node and every edge will be explored in the worst case.

In case of NBP localization, we exclude all the anchors from the BFS algorithm since they do not form the loops in the graph (they just send, and never receive the messages). A graph generally has a large number of spanning trees, but since our graph is unweighted we choose few (minimum 2) of them in a partly random way. The first root node we choose randomly from the set of all unknown nodes. In order to maximize the difference between two spanning trees, the second root node has to be as far as possible from the first root node. Thus, it should be one of the leaf nodes which is the maximum-hop away from the root. If we want to form more spanning trees, the analog constraint will be used. An example of a loopy graph and two corresponding spanning trees, formed by BFS with mentioned constraints, are illustrated in Figure 1. Note that, using BFS, it is not possible to form two spanning trees with completely different edges and that usually some of the edges will be out of both spanning trees. If we want to include all edges, we have to add more spanning trees but it is usually not necessary since it will only provide us a redundant information. It is especially the case in the networks with high connectivity.

The NBP method is naturally distributed through the graph which means that there is no central unit which will handle all computations. Therefore, the proposed BFS method has to be done in a distributed way. This can be simply done if each unknown node initially broadcast its \( ID \) to all neighbors, which will continue to broadcast to others, and so on, until each unknown node aggregates a list of all unknown nodes in the graph. One node (e.g. with lowest \( ID \)) has to be assigned to choose the root node from the list and give him a permission (by multihop broadcasting) to start BFS algorithm. Then, the chosen root node has all initial data to start BFS algorithm, and, when it is necessary, has only to broadcast all data (i.e. variables from Alg. 1) to all its neighbors.

Finally, NBP-ST algorithm represents two (or more) independent runnings of the NBP algorithm based on formed spanning trees. Each running will provide us weighted particles of the node beliefs computed by (8). The simplest way to fuse these beliefs is to resample with replacement [13] from weighted particles from all spanning trees, which produces the particles with same weights. The collection of particles from all spanning trees represents our final output, from which we can easily extract any parameter that we need (e.g., mean value for location estimate). The pseudocode in Alg. 2 illustrates NBP-ST method.

### Algorithm 1 Breadth First Search (BFS)

1. **Input**: list of nodes \( Q \) and root node \( root \)
2. **Set current root**: \( r \leftarrow root \)
3. **while** \( Q \) is not empty **do**
   4. **for all** nodes \( t \in G \), **do**
      5. **if** \( t \in Q \) **then**
         6. Remove \( t \) from \( Q \)
         7. Insert \( t \) in auxiliary list \( Q_r \)
         8. Insert \( d_{rt} \) in \( S \)
      9. **end if**
   10. **end for**
11. **Set current root**: \( r \leftarrow \) first unused node from \( Q_r \)
12. **end while**
13. **Output**: spanning tree \( \{Q, S\} \)

![Fig. 1: (a) Example of loopy graph, (b), (c) Spanning trees created by BFS method.](image-url)
Algorithm 2 NBP-ST method for localization

1: \textbf{for all} nodes \textbf{do}
2: \hspace{1em} Take sensing actions
3: \hspace{1em} Set all parameters to the initial values
4: \hspace{1em} Broadcast own and all received IDs and listen for other sensor broadcasts (until receive all IDs)
5: \textbf{end for}
6: Set a list of nodes for BFS (excluding anchors): \( Q \)
7: Choose randomly the first root from the list \( Q \): root
8: \textbf{for all} spanning trees \textbf{do}
9: \hspace{1em} Run BFS (Alg. 1)
10: \hspace{1em} Run NBP on defined spanning tree
11: \hspace{1em} Choose root node as far as possible from the previous roots
12: \textbf{end for}
13: Fuse all beliefs into one and compute location estimates

anchors are randomly deployed within four areas (4m x 4m) near the edges, respectively. The standard deviation of the Gaussian noise is set to \( \sigma = 0.3m \) and the number of iteration is set to \( N_{\text{iter}} = 3 \). All simulations are done for \( N = 50 \) and \( N = 100 \) particles with respect to the transmission radius \(( R = 4m - 10m )\). The error is defined as a distance between true and estimated location. To measure the communication cost, we count \textit{elementary messages}, which represent any scalar value (e.g., one coordinate of one particle). Finally, each point in the simulations represents the average over 20 Monte Carlo trials.

Using the defined scenario, we compared NBP and NBP-ST algorithms. For NBP-ST, we used 2 spanning trees. The original network and 2 spanning trees created by BFS are illustrated in Figure 4, respectively. Regarding accuracy and coverage (percentage of located nodes with error less than predefined tolerance; in our case 0.2R) in Figures 2 and 3, NBP-ST performs better than NBP for \( R > 7m \), approximately. Obviously, for these values of \( R \) there is a large number of loops in the network which decreases the performance of NBP method. For lower values of \( R \), we could expect that NBP-ST performs with higher (or same) accuracy, but we cannot forget that, by using only 2 spanning trees, we didn’t include all information (removed edges) that we have. Thus, the NBP outperforms NBP-ST in this case.

Regarding computational/communication cost (Figures 5 and 6), NBP-ST performs better than NBP for \( R > 8m \) and \( R > 9m \), respectively. In order to explain this we have to remember two main things we have taken into account: removing the edges in order to form the spanning trees and running NBP two times in these spanning trees. First operation decreases the costs, but the second one increases it. Therefore, in low connected networks the second operation predominates, but in high connected networks the first one predominates. Furthermore, it is important that the computational and communication cost have the smaller curve slope with respect to the transmission radius. This feature provides us more precise information about battery life.

The final conclusion is that NBP-ST algorithm performs better than NBP in all terms, for \( R > R_{\text{min}} \). In our case \( R_{\text{min}} = 9m \), but this parameter depends on the density in the network (i.e., average connectivity). These values of transmission radius are available in all today’s wireless sensor nodes even if they are set to work in low-power mode.

V. CONCLUSIONS AND FUTURE WORK

As presented in this paper, NBP localization method has poor performance in highly loopy networks. Moreover, the connectivity in these networks is very high which makes computational and communication burdens for low-power applications. Therefore, we proposed NBP-ST method based on the spanning trees created by the BFS method which is optimal for the unweighted graphs. The BFS method is done in a distributed way which makes the algorithm applicable in ad-hoc/sensor networks. We can conclude that NBP-ST method performs better than NBP in terms of accuracy, computational and communication cost for high values \(( > R_{\text{min}} )\) of transmission radius. There are many open directions for the future work. The main one is the implementation of
localization algorithm based on nonparametric generalized belief propagation (NGBP). Some versions already exist, but they are still very complex for the large-scale ad hoc sensor networks. Moreover, an adaptive algorithm based on both NBP and NBP-ST could be a good idea. Finally, the hardware implementation of all methods in indoor and outdoor scenario could be useful to provide us more precise conclusions.

ACKNOWLEDGMENT

This work is supported by the FPU fellowship from Spanish Ministry of Science and Innovation. Furthermore, we thank partial support by project ICT-217033 WHERE, program CONSOLIDER-INGENIO 2010 CSD2008-00010 COMONSENS and National Project M3HF.

REFERENCES