Sensor localization using nonparametric generalized belief propagation in network with loop

Vladimir Savic and Santiago Zazo

Post Print

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

©2009 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.


Postprint available at: Linköping University Electronic Press

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-81430
Sensor Localization using Nonparametric Generalized Belief Propagation in Network with Loops

Vladimir Savic
Signal Processing Application Group
Polytechnic University of Madrid
Ciudad Universitaria S/N, 28040 Madrid,
vladimir@gaps.ssr.upm.es

Santiago Zazo
Signal Processing Application Group
Polytechnic University of Madrid
Ciudad Universitaria S/N, 28040
santiago@gaps.ssr.upm.es

Abstract - Belief propagation (BP) is one of the best-known graphical model for inference in statistical physics, artificial intelligence, computer vision, etc. Furthermore, a recent research in distributed sensor network localization showed us that BP is an efficient way to obtain sensor location as well as appropriate uncertainty. However, BP convergence is not guaranteed in a network with loops. In this paper, we propose localization using generalized belief propagation based on junction tree method (GBP-JT) and nonparametric (particle-based) approximation of this algorithm (NGBP-JT). We illustrate it in a network with loop where BP shows poor performance. In fact, we compared estimated locations with Nonparametric Belief Propagation (NBP) algorithm. According to our simulation results, GBP-JT resolved the problems with loops, but the price for this is unacceptable large computational cost. Therefore, our approximated version of this algorithm, NGBP-JT, reduced significantly this cost, with little effect on accuracy.

Keywords: Localization, generalized belief propagation, junction tree, particle filters, loops.

1 Introduction

The localization consists in obtaining the relative or absolute position of a sensor node together with the uncertainty of its estimate. Equipping every sensor with a GPS receiver or equivalent technology may be expensive, energy prohibitive and limited to outdoor applications. Therefore, 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. If unknown nodes were capable of high-power transmission, they would be able to make measurements with all anchor nodes (single-hop technique). However, we prefer to use energy-conserving devices without power amplifier, with lack the energy necessary for long-range communication. In this, multi-hop, case, each sensor has available noisy measurements only to several neighboring sensors.

A recent direction of research in distributed sensor network localization is the use of particle filters [1, 2]. In [3], Ihler et al. formulated the sensor network localization problem as an inference problem on a graphical model and applied particle based variant of belief propagation (BP) methods [4], the so-called nonparametric belief propagation (NBP) algorithm, to obtain an approximate solution to the sensor locations. Comparing with deterministic algorithms [5, 6, 7], the main advantages of this statistical approach are its easy implementation in a distributed fashion and sufficiency of a small number of iterations to converge. Furthermore, NBP is capable of providing information about location estimation uncertainties and accommodating non-Gaussian distance measurement errors. However, NBP convergence is not guaranteed in a network with loops [4, 8], or even if NBP converges, it could provide us less accurate estimates.

Therefore, in this paper, we present a new variant of the NBP method which solves problem with loops. We propose localization using generalized belief propagation based on junction tree (GBP-JT) and nonparametric (particle based) approximation of this algorithm (NGBP-JT). Junction tree model is a generalization of belief propagation (BP) that is correct for arbitrary graphs. Jordan proved it using elimination procedure [9]. Comparing with Ihler’s Nonparametric Belief Propagation (NBP) algorithm, GBP-JT converges well in network with loops, but the price for this is unacceptable large computational cost. Therefore, we implemented approximated version of this algorithm, NGBP-JT, by drawing high-dimensional particles from appropriate cliques in the network. Moreover, in order to draw samples in high-probabilistic area, we used improved sampling procedure which utilizes information from first phase of the algorithm as an a priori. This version reduces significantly computational cost (over a 100 times), with little effect on accuracy.

The remainder of this paper is organized as follows. In Section 2, we review standard BP and condition for its convergence. In Sections 3 and 4, we propose GBP-JT and NGBP-JT algorithms, respectively. Simulation results are presented in Section 5. Finally, Section 6 provides some conclusions and future work perspective.
2 Convergence of Belief Propagation

In the standard BP algorithm, the belief at a node \( i \) is proportional to the product of the local evidence at that node \( \psi_i(x_i) \), and all the messages coming into node \( i \):

\[
M_i(x_i) = k \psi_i(x_i) \prod_{j \in N(i)} m_{ji}(x_i) \tag{1}
\]

where \( k \) is a normalization constant and \( N(i) \) denotes the neighbors of node \( i \). The messages are determined by the message update rules:

\[
m_{ji}(x_i) = \sum_{x_j} \psi_j(x_j) \psi_{ij}(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_j) \tag{2}
\]

where \( \psi_{ij}(x_i, x_j) \) is pairwise potential between nodes \( i \) and \( j \). On the right-hand side, there is a product over all messages going into node \( j \) except for the one coming from node \( i \).

In practical computation, one starts with nodes at the edge of the graph, and only computes a message when one has available all the messages required. It is easy to see [4] that each message needs to be computed only once for single connected graphs. That means that whole computation takes a time proportional to the number of links in the graph, which is dramatically less that the exponentially large time that would be required to compute marginal probabilities naively. In other words, BP is a way of organizing the "global" computation of marginal beliefs in terms of smaller local computations.

The BP algorithm, defined by equations (1) and (2), does not make a reference to the topology of the graph that it is running on. Thus, there is nothing to stop us from implementing it on a graph that has loops. One starts with some initial set of messages, and simply iterates the message-update rules (2) until they eventually converge, and then can read off the approximate beliefs from the belief equations (1). But 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. One can indeed find examples of graphical models with loops, where, for certain parameter values, the BP algorithm fails to converge or predicts beliefs that are inaccurate. On the other hand, the BP algorithm could be successful in graphs with loops, e.g. error-correcting codes defined on Tanner graphs that have loops [10].

This can be proved using Bethe approximation to the "free energy" [4, 8]. The fixed points of the BP algorithm correspond to the stationary points of the Bethe "free energy". To make this more clear, let's define for one graphical model, a joint probability function \( p(x) \). If we have some other approximate joint probability function \( b(x) \), we can define a "distance" between \( p(x) \) and \( b(x) \), called Kullback-Leibler (KL) distance, by:

\[
D(b | p) = \sum_x b(x) \ln \frac{b(x)}{p(x)}
\]

where \( Z \) is a normalization constant, and the "temperature" \( T \) is just a parameter that defines a scale of units for the "energy" \( E \). For simplicity, we can choose \( T = 1 \). Using eqs. (3) and (4), we find the KL distance:

\[
D(b | p) = \sum_x b(x) E(x) + \sum_x b(x) \ln b(x) + \ln Z
\]

So we see that this KL distance will be zero when approximate probability function \( b(x) \) will equal to the exact probability function \( p(x) \). The Bethe approximation is the case when joint belief \( b(x) \) is function of single-node beliefs \( b(x_i) \) and two-node beliefs \( b(x_i, x_j) \). Yedidia et al. proved [4] that for a single-connected graph, values of these beliefs that minimize the Bethe free energy, will correspond to the exact marginal probabilities. For graph with loops, these beliefs will only be approximations, although a lot of them are quite good.

3 Localization using Generalized Belief Propagation

Our goal in this section is to develop new localized algorithm using generalized belief propagation based on junction tree method (GBP-JT). Junction tree algorithm is a standard method for exact inference in graphical model [9]. The graph is first triangulated (added "virtual" edges so that every loop of length more than 3 has a chord). Given a triangulated graph, with cliques \( C_i \) and potentials \( \psi_{C_i}(x_{C_i}) \), and given corresponding junction tree which defines links between the cliques, we send the following message from clique \( C_i \) to clique \( C_j \) by the message update rule:

\[
m_{ij}(x_{S_i}) = \sum_{x_{S_j}} \psi_{C_i}(x_{C_i}) \prod_{k \in N(i) \setminus j} m_{kj}(x_{S_k}) \tag{6}
\]

where \( S_i = C_i \cap C_j \), and where \( N(i) \) are the neighbors of clique \( C_i \) in the junction tree. The belief at clique \( C_i \) is proportional to the product of the local evidence at that clique and all the messages coming into clique \( i \):
The equations (6), (7), and (8) represent generalized belief propagation algorithm which is valid for arbitrary graphs. The BP algorithm defined with (1) and (2) is a special case of GBP-JT, obtaining by noting that the original tree is already triangulated, and has only pairs of nodes as cliques. In this case, sets $S_i$ are single nodes, and marginalization using eq. (8) is unnecessary.

Let’s show how it works in our example in Figure 1. The network has 10 nodes, 5 anchors (nodes 6-10) and 5 unknowns (nodes 1-5). There is a loop 1-2-4-5-3, so we have to triangulate it by adding two more edges (2-3 and 3-4). Then we can define 8 cliques in the graph: $C_1 = \{x_1, x_2, x_3\}$, $C_2 = \{x_2, x_3, x_4\}$, $C_3 = \{x_3, x_4, x_5\}$, $C_4 = \{x_4, x_5\}$, $C_5 = \{x_5, x_6\}$, $C_6 = \{x_6, x_7\}$, $C_7 = \{x_7, x_8\}$, $C_8 = \{x_8, x_9\}$. The appropriate potentials of 3-node cliques are given by:

$$
\psi_{C_1}(x_1, x_2, x_3) = \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \\
\psi_{C_2}(x_2, x_3, x_4) = \psi_{24}(x_2, x_4) \\
\psi_{C_3}(x_3, x_4, x_5) = \psi_{35}(x_3, x_5) \psi_{45}(x_4, x_5) \\
$$

(9)

Note that “virtual” edges do not appear in these equations since they are used only to define cliques. Other cliques, defined over pairs of nodes, are nothing else than potential functions between two nodes already known from standard BP:

$$
\psi_{C_4}(x_4, x_5) = \psi_{45}(x_4, x_5), \quad \psi_{C_5}(x_5, x_6) = \psi_{56}(x_5, x_6), \quad \psi_{C_6}(x_6, x_7) = \psi_{67}(x_6, x_7), \quad \psi_{C_7}(x_7, x_8) = \psi_{78}(x_7, x_8), \quad \psi_{C_8}(x_8, x_9) = \psi_{89}(x_8, x_9). \\
$$

The junction tree corresponding to the network in Figure 1 is shown in Figure 2. As we can see, “anchor cliques” ($C_4 - C_8$) do not receive messages, so this graph does not contain loops. Actually, these “anchor cliques” also include one unknown node so we can send them messages, but this node also could be located marginalizing the belief of some other clique.

In the next step, we can compute all messages using equation (6). The complete set of messages is given by:

$$
m_{61}(x_1) = \psi_{16}(x_1, x_6^*) , \quad m_{53}(x_3) = \psi_{53}(x_3, x_6^*) \\
m_{71}(x_2) = m_{72}(x_2) = \psi_{27}(x_2, x_7^*) \\
m_{42}(x_4) = m_{43}(x_4) = \psi_{49}(x_4, x_9^*) \\
m_{81}(x_3) = m_{82}(x_3) = m_{83}(x_3) = \psi_{38}(x_3, x_8^*) \\
m_{12}(x_2, x_3) = \psi_{27}(x_2, x_7^*) \psi_{38}(x_3, x_8^*) \sum_{x_5} \psi_{16}(x_1, x_6^*) \psi_{C_1}(x_1, x_2, x_3) \\
m_{13}(x_3, x_4) = \psi_{49}(x_4, x_9^*) \psi_{38}(x_3, x_8^*) \sum_{x_5} \psi_{53}(x_3, x_6^*) \psi_{C_1}(x_3, x_4, x_5) \\
m_{21}(x_2, x_3) = \psi_{27}(x_2, x_7^*) \psi_{38}(x_3, x_8^*) \sum_{x_4} \psi_{49}(x_4, x_9^*) \psi_{C_1}(x_2, x_3, x_4) m_{23}(x_3, x_4) \\
m_{23}(x_3, x_4) = \psi_{49}(x_4, x_9^*) \psi_{38}(x_3, x_8^*) \sum_{x_2} \psi_{27}(x_2, x_7^*) \psi_{C_1}(x_2, x_3, x_4) m_{23}(x_2, x_3) \\
$$

where asterisk denotes the known location of the anchor node and the messages from "anchor cliques" are directly replaced by appropriate potential function. The beliefs of cliques are computed using equation (7):

$$
M_1(x_2, x_3, x_4) = \psi_{C_1}(x_1, x_2, x_3) \psi_{16}(x_1, x_6^*) \psi_{27}(x_2, x_7^*) \psi_{38}(x_3, x_8^*) m_{21}(x_2, x_3) \\
M_2(x_2, x_3, x_4) = \psi_{C_2}(x_2, x_3, x_4) \psi_{27}(x_2, x_7^*) \psi_{38}(x_3, x_8^*) \psi_{49}(x_4, x_9^*) m_{23}(x_3, x_4) \\
M_3(x_3, x_4, x_5) = \psi_{C_3}(x_3, x_4, x_5) \psi_{38}(x_3, x_8^*) \psi_{49}(x_4, x_9^*) \psi_{53}(x_3, x_6^*) m_{32}(x_2, x_3) \\
$$

(12)

Now it’s easy to compute beliefs of single nodes by marginalizing beliefs of cliques using eq. (8). Obviously, it's sufficient to know beliefs of $C_1$ and $C_3$ since these cliques include all unknown nodes. Marginalization of $C_2$ provides degree of freedom and could be used to check the estimated positions of some nodes (in our case, for nodes 2, 3 and 4).
Finally, in order to use this method for localization, we have to define potential functions. In our case, we assumed that we didn't obtain a priori information about node position, so single-node potentials are equal to 1 (in opposite case, beliefs computed using eq. (7) have to be multiplied by their own potentials). The pairwise potential between nodes $t$ and $u$ is given by:

$$
\psi_{tu}(d) = \begin{cases} 
\frac{1}{d^2} & \text{if } d \leq R, \\
0 & \text{otherwise.}
\end{cases}
$$

$$
\psi_{tu}(d) = \begin{cases} 
\frac{1}{d^2} & \text{if } d \leq R, \\
0 & \text{otherwise.}
\end{cases}
$$

where $d$ is the probability of detecting nearby sensors; in our case we used improved model which assumes that the probability of detecting nearby sensors falls off exponentially with squared distance:

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

where $R$ is the transmission radius. The binary variable $o_{tu}$ indicates whether this observation is available ($o_{tu} = 1$) or not ($o_{tu} = 0$). And the last remaining parameter is measured distance. The unknown node $t$ obtains a noisy measurement $d_{tu}$ of its distance from detected node $u$:

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

In our case, we used Gaussian distribution for $P_t$, but, as we can see, it's very easy to change it to any desired distribution (e.g. obtained by running training experiment in the deployment area).

The proposed GBP-JT algorithm is not unique. There are a lot of variations of this method; the best known is cluster variation method [8]. However, it can be shown that they are quite similar. For example, in [8] is described the relationship between different region-based approximations. The main goal is achieved in all of them: estimated beliefs are correct in network with loops. However, the price for this is unacceptable large computational cost, so we are going to implement approximated version of GBP-JT algorithm.

4 Nonparametric Generalized Belief Propagation

In order to obtain acceptable spatial resolution for unknown nodes, number of discrete points in the deployment area (e.g. $N_x \times N_y$ for 2D grid) must be too large for GBP to be computationally feasible [3]. Besides, the presence of nonlinear relationships and potentially highly non-Gaussian uncertainties in sensor localization makes GBP undesirable. However, using particle-based representations via non-parametric generalized belief propagation (NGBP), enables the application of GBP to inference in sensor networks. In this section we propose NGBP-JT, particle-based approximation of GBP-JT method for the same example of network from previous section (Figures 1 and 2).

4.1 Drawing initial particles

Let’s draw $N_C$ weighted particles from cliques $C_i$ and $C_j$:

$$
\{W_i^j, X_i^j\} = \{W_i^j, X_i^{1, 2, 3^j} \}
$$

where $W_i^j$ represents weight of 6-dimensional particle $X_i^j$ from clique $C_i$ which consists from three 2-dimensional particles from node $i$ ($X_i^{1, 2, 3^j}$). For now, we don’t need particle from clique $C_j$. There is a lot of ways to draw these particles. In general, we can draw all particles uniformly within the deployment area, but it requires
significant large number of particles (e.g. 100 particles drawn for each node, corresponds to \(100 \times 100 \times 100 = 10^6\) particles of its clique). Therefore, we will immediately include all information available within the clique: potential functions given by (9) and (13) which represent our information about distance between nodes within the clique. First, we draw particles of node \(i\) uniformly within the deployment area. To draw particle of any neighboring node \(u\), we shift particle of node \(i\) in the random direction for an amount which represents the observed distance between these two nodes:

\[
X_{n,u} = X_{i,u} + (d_u + v)\sin(\theta^i) \cos(\theta^i), \quad \theta^i \sim \text{Unif}(0, 2\pi), \quad j = 1, \ldots, N_C
\] (17)

We will use simplified notation of the above equation:

\[
X'_{i,u} = \text{shift}(X'_{i,u}, d_u)
\] (18)

Assuming that we have already drawn samples, e.g. from nodes 1 and 5, we can compute particles of other nodes:

\[
\begin{align*}
X'_{i,2} &= \text{shift}(X'_{i,1}, d_{i2}), \quad X'_{i,3} = \text{shift}(X'_{i,1}, d_{i3}) \\
X'_{i,4} &= \text{shift}(X'_{i,5}, d_{i4}), \quad X'_{i,5} = \text{shift}(X'_{i,5}, d_{i5})
\end{align*}
\]

Since these particles are drawn from \(\psi_C\) and \(\psi_C\), respectively, and that we already included all information which place these particles in high-probabilistic regions with respect to \(X'_{i,1}\) and \(X'_{i,5}\) (see eqs. (13) and (19)), all clique’s weights can be approximated with the same value:

\[
W'^j_i = W'^j_i = \frac{1}{N_C}, \quad j = 1, \ldots, N_C
\] (20)

Note that all particles of nodes within the clique have one common weight, e.g. \([W'^j_i, [X'_{i,2}, X'_{i,3}, X'_{i,5}]]\). Our initial set of particles for clique \(C_1\) is illustrated in Figure 3.

![Figure 3. Initial set of particles for clique \(C_1\)](image)

### 4.2 Computing messages

Having drawn all particles, we can now compute all messages. Messages \(m_{12}\) and \(m_{32}\) are function of \(m_{12}\) and \(m_{32}\) respectively (see eqs. (11)), so they will be computed later. Also, messages from the “anchor cliques” will be directly replaced with appropriate potential function. So we start with messages \(m_{12}\) and \(m_{32}\) which depends on \(\psi_C\) and \(\psi_C\), from which we already have drawn particles. Let’s represent these two messages in slightly different form:

\[
m_{12}(x_1, x_2, x_3) = \sum_{x_i} M_{12}(x_1, x_2, x_3)
\]

\[
M_{12}(x_1, x_2, x_3) = \psi_{23}(x_2, x_3) \psi_{13}(x_1, x_3) \psi_{12}(x_1, x_2) \psi_{11}(x_1, x_2)
\]

\[
m_{32}(x_3, x_4) = \sum_{x_i} M_{32}(x_3, x_4, x_5)
\]

\[
M_{32}(x_3, x_4, x_5) = \psi_{45}(x_4, x_5) \psi_{35}(x_3, x_4) \psi_{34}(x_3, x_4) \psi_{33}(x_3, x_4, x_4)
\]

Defined factors, \(M_{12}\) and \(M_{32}\), are some kind of unmarginalized messages, so we’ll call them joint messages. Now it’s very easy to compute weighted particles from these joint messages (\([W'^{\text{joint}}_i, X'^{\text{joint}}_i]\)):

\[
X'^{\text{joint}}_i = X'^j_i = [X'_{i,1}, X'_{i,2}, X'_{i,3}]
\]

\[
W'^{\text{joint}}_i = \psi_{23}(X'_{i,1}, x_2) \psi_{13}(X'_{i,2}, x_3) \psi_{12}(X'_{i,3}, x_4) \psi_{11}(X'_{i,4}, x_5)
\]

Before computing final messages, we noticed usual problem, sample depletion [2], the problem when one, or few, of the weights are much larger than the rest. This means that any sample-based estimate will be unduly dominated by the influence of just few particles. In our case, it’s expected because we are working in 6-dimensional space where it’s very hard to draw good sample (clique with position and shape similar to the right one – see Figure 3). Therefore, we resample with replacement [1, 3], which will produce \(N_C\) equal-weight particles (\(W''_i = 1 / N_C\)). In our case, we have to resample from cliques, thus the easiest way is to resample from single nodes using standard resampling procedure [1], and then to synchronize indexes in order to keep original shapes of the particles. This procedure is illustrated for \(M_{12}\), by the following pseudocode:

\[
[W_{12}, X_{11}, \text{index}] = \text{resample}(W_{12}, X_{11});
\]

\[
\text{for } j=1:N_C;
\]

\[
X'^{\text{index}(j)}_{12} = X'^{\text{index}(j)}_{12}; \quad X'^{\text{index}(j)}_{13} = X'^{\text{index}(j)}_{13};
\]

\[
\text{end}
\]

1 Note the difference between \(X'^{i}_{12}\) and \(X'^{i}_{12}\)}
where \( \{W_{12}, X_{1:2}\} \) is the vector of \( N_C \) particles from node 1 (part of joint message), and index is the vector of old (pre-resampled) indexes of new particles.

Now we are ready to compute particles from messages \( \{w_{1:i}, x_{1:i}\} \). The marginalization of joint messages is very easy since we already have weighted particles from them. So we just need to discard one data, and keep the same weights. Thus, they are given by:

\[
\begin{align*}
x_{1:2}^j &= X_{1:2}^j (2 : 3) = [X_{1:2}^j, X_{1:4}^j], \quad w_{1:2}^j = W_{1:2}^j = 1/ N_C \\
x_{1:2}^j &= X_{1:2}^j (1 : 2) = [X_{1:2}^j, X_{1:4}^j], \quad w_{1:2}^j = W_{1:2}^j = 1/ N_C \\
\end{align*}
\]  

Finally, we can compute particles of other two messages, \( m_{12} \) and \( m_{32} \). According to eqs. (11), they are function of \( \psi_{C, m_{12}} \) and \( \psi_{C, m_{32}} \), respectively, so we will draw particles from these products and then re-weight by remainder of eq. (11). Actually, two single-node particles of messages \( m_{12} \) and \( m_{32} \) are already computed (see eqs. (26)), so we have just to draw missing particle using information from \( \psi_{C_{1}} \), the observed distance between nodes 2 and 4. The result of this procedure are particles of joint messages \( M_{21} \) and \( M_{23} \). Marginalizing them, we obtain final messages \( m_{21} \) and \( m_{23} \). The complete procedure is given as follows:

\[
\begin{align*}
X_{2:1} &= \{\text{shift}(x_{2:1}^j (2), d_{2:4}), x_{1:2}^j (1), x_{1:2}^j (2)\} \\
W_{2:1} &= \psi_{20}(X_{2:1}^j (3), x_{1:2}^j (2)) \psi_{21}(X_{2:1}^j (2), x_{1:2}^j (1)) \psi_{22}(X_{2:1}^j (1), x_{1:2}^j) w_{1:2}^j \\
\text{resample and synchronize} \\
x_{2:1} &= X_{2:1}^j (1 : 2) = [X_{2:1}^j (1), X_{2:1}^j (2)], \quad w_{2:1} = W_{2:1} = 1/ N_C \\
X_{2:3} &= \{x_{1:2}^j (1), x_{1:2}^j (2), \text{shift}(x_{1:2}^j (1), d_{2:4})\} \\
W_{2:3} &= \psi_{20}(X_{2:3}^j (3), x_{1:2}^j (2)) \psi_{21}(X_{2:3}^j (2), x_{1:2}^j (1)) \psi_{22}(X_{2:3}^j (1), x_{1:2}^j) w_{1:2}^j \\
\text{resample and synchronize} \\
x_{2:3} &= X_{2:3}^j (2 : 3) = [X_{2:3}^j (2), X_{2:3}^j (3)], \quad w_{2:3} = W_{2:3} = 1/ N_C
\end{align*}
\]

**4.3 Computing final beliefs**

To estimate beliefs of unknown nodes, we are going to compute beliefs of cliques using already computed particles of messages. According to eqs. (12), beliefs \( M_1, M_2 \) and \( M_3 \) are function of \( \psi_{C, m_{21}} \), \( \psi_{C, m_{32}} \) and \( \psi_{C, m_{32}} \), respectively, so we will draw particles from these products and then re-weight by remainder of eqs. (12).

Let’s start with \( M_1 \) and its corresponding product \( \psi_{C, m_{21}} \). As we can see in eqs. (9), \( \psi_{C_{1}} \) includes information about distance between nodes 2 and 3, as well as between nodes 1 and 3. Besides, message \( m_{21} \) includes information about positions of nodes 2 and 3. So we just need to locate node 1, using available positions and distances. It could be done geometrically by intersecting circles, but we prefer statistical approach which is quite faster. It’s done by following pseudocode:

\[
\text{for } j=1: N_C \\
X_{1:5}^j = x_{1:5}^j (1); \quad X_{1:5}^j = x_{1:5}^j (2) \\
\text{while } (k < k_{\text{max}}) \\
X_{1:5}^j = \text{shift}(X_{1:5}^j, d_{1:5}); \\
\text{if } \text{abs}(X_{1:5}^j, X_{1:5}^j) \in (d_{1:5} - \varepsilon, d_{1:5} + \varepsilon) \text{ break } (\text{29}) \\
\text{end} \\
k = k + 1; \\
W_{1:5}^j = \psi_{10}(X_{1:5}^j, x_{1:5}^j) \psi_{27}(X_{2:5}^j, x_{2:5}^j) \psi_{30}(X_{3:5}^j, x_{3:5}^j) w_{1:2}^j
\]

where \( \text{abs}(X_{1:5}^j, X_{1:5}^j) \) is estimated distance between these two particles, and \( \varepsilon \) is predefined tolerance. If we do not obtain “good particle” after \( k_{\text{max}} \) iterations, that’s mean that these two circles cannot intersect, so our particle is the position shifted for \( d_{1:5} \) in random direction. This is not a problem because this wrong particle will obviously have later a very small weight (filtered by potential functions from anchors). The other problem is bimodality, if the circles intersect in two points; but the wrong particle will be also filtered in same way. The same procedure is done for \( M_3 \) since we have distance between nodes 5 and 4, as well as between nodes 3 and 5, and message \( m_{23} \) which includes information about positions of nodes 3 and 4.

As we already mentioned, the belief \( M_2 \) is not necessary since other two cliques include positions of all unknown nodes. Anyway, we will show the procedure because it’s slightly different. We have to draw particles from product of two 4-dimensional messages and as result we expect 6-dimensional message. So, if we want to avoid to draw randomly missing particles (e.g. for message \( m_{12}(x_2, x_3) \)), we would have to draw single-node particles from \( x_4 \), we will directly draw particles from the product \( \psi_{C_{1}}(x_2, x_3, x_4)m_{21}(x_2, x_5)m_{32}(x_3, x_4) \) and then re-weight by remainder of eq. (12). The following procedure shows it:

\[
\begin{align*}
X_{4:1}^j &= \{\text{shift}(x_{4:1}^j (2), d_{4:5}), x_{1:2}^j (1), x_{1:2}^j (2)\}; \\
X_{4:2}^j &= \{x_{1:2}^j (1), x_{1:2}^j (2), \text{shift}(x_{1:2}^j (1), d_{4:5})\}; \\
X_{4:3}^j &= \text{choose}(X_{4:3}^j \cup X_{4:2}^j , N_C); \quad W_{4:1}^j = 1/ N_C \\
W_{4:2}^j &= \frac{m_{12}(X_{1:2}^j, X_{1:4}^j)m_{32}(X_{2:3}^j, X_{2:4}^j)}{m_{12}(X_{2:3}^j, X_{2:4}^j)m_{21}(X_{2:3}^j, X_{2:4}^j)} W_{4:1}^j \\
W_{4:3}^j &= \psi_{27}(X_{2:3}^j, x_{1:2}^j) \psi_{30}(X_{2:4}^j, x_{1:2}^j) W_{4:1}^j
\end{align*}
\]

where function choose \( (X_{4:3}^j \cup X_{4:2}^j, N_C) \) chooses randomly \( N_C \) particles from \( 2N_C \). This procedure is known as importance sampling [3], the approximation of original distribution \( (m_{12}m_{32}) \) with proposal one \( (m_{21} + m_{32}) \) from which is easy to draw samples \( (X_{4:3}^j \cup X_{4:2}^j) \), and then re-weighting \( (m_{21} + m_{32}) \) to compensate the error.

\[2\text{ For the simplicity, updated and old particles are denoted by same symbols.}\]
The final estimated positions of unknown nodes are given by mean values of particles from clique $C_m$:

$$x_n^{\text{est}} = \frac{\sum_{j=1}^{N_C} W^j_m x_{w,n}^j}{\sum_{j=1}^{N_C} W^j_m}$$  \hspace{1cm} (31)

### 4.4 Improved sampling procedure

There is one important modification to this algorithm that can reduce significantly the initial number of particles. As we already mentioned, if we draw $N$ particles from one node, generally it corresponds to $N_C = N^3$ particles of a 3-node clique. However, we included information about distance, so our new number for the same clique is $N_C = NN_\theta^3$ where $N_\theta$ represents the number of possible angles. But this number is still very large, so we would like to include additional information.

We assumed that there is no a priori information about node position. However, after very first phase of algorithm, we computed joint messages $M_{12}$ and $M_{32}$ which include current information about positions of cliques $C_1$ and $C_3$. At this point, particles are concentrated in a smaller region (except for very few of them), so we can draw new set of particles around single-node particles of joint messages. For $C_1$, it’s done by following procedure:

$$d^j = \text{Unif}(0,r), \quad X^j_{1,1} = \text{shift}(X^j_{1,1},d^j)$$

$$X^j_{1,2} = \text{shift}(X^j_{1,1},d_{12}), \quad X^j_{1,3} = \text{shift}(X^j_{1,1},d_{13})$$

$$W^j_1 = 1/N_C$$

- compute messages again

where $r$ is the radius of deployment area of new particles. Computing messages again is very important since we draw new set of particles, which means that we have to run algorithm from the beginning. Of course, for clique $C_1$, we use analog procedure. This improved procedure allows us to decrease initial number of samples to $N_C = NN_\theta^3/n$ where $n$ is the reducing factor that could be found experimentally. Theoretically, it’s proportional to the ratio of the new to the old deployment area.

### 5 Simulation Results

We simulated the network from Figure 1 using NBP, GBP-JT and NGBP-JT algorithms. We placed 10 nodes in $2m \times 2m$ area, 5 anchors and 5 unknowns. We set the values of transmission radius ($R = 25\%$ of diagonal length of the deployment area) and standard deviation of measured distance ($\sigma = 0.1m = 14\%R$). Number of iteration for NBP is set to the length of the longest path in the graph ($N_{\text{iter}} = 5$), and for GBP-JT/NGBP-JT there is obviously, in our example, just one iteration. Number of particle for NBP is set to $N = 400$, so corresponding number of grid points for GBP-JT is $N_G = 20 \times 20$. For NGBP-JT we used improved sampling procedure with radius $r = 0.1$ and experimentally we found out reducing factor which do not change the accuracy ($n \approx 4$). Assuming that minimum number of possible angles could be approximated with $N_\theta = 10$, we set the number of clique’s particles to $N_C = NN_\theta^3/n = 10000$.

We ran the simulation for NBP and NGBP-JT, and obtained results shown in Figure 4. Obviously, the location estimates for the NGBP-JT are more accurate since this algorithm is correct for network with loops. NBP algorithm does not converge well for a few nodes, but for some other values of parameters, or with different positions of some nodes, it provides estimates with almost same accuracy as NGBP-JT. However, comparing uncertainties for NBP and NGBP-JT (contours in Figures 4a and 4b), we can see that NBP provides us better guarantees of its estimate.
Finally, we checked the averaged accuracy with respect to the deviation of measured distance for all three methods (Figure 5). The accuracy of GBP-JT is always higher than accuracy of NBP and NGBP-JT. NGBP-JT provides us better accuracy than NBP for some usual values of distance deviation (e.g. for measurements using time of arrival, the error is 5-20 %R [11]), and unexpectedly worse accuracy for higher values of mentioned deviation. Anyway, this accuracy could be increased, using larger number of particles (e.g. increasing $N_\theta$), until the “bottom line” defined by the accuracy of GBP-JT.

Comparing with NBP/NGBP-JT, the computational cost of GBP-JT is, of course, very large (311 MFlops) and absolutely unacceptable. Nonparametric approximation of this algorithm decreased it around 25 times, and improved sampling additional 4 times. So the final computational cost of NGBP-JT in simulated example is 7.89 MFlops, just double as many comparing with NBP (3.38 MFlops).

6 Conclusions

As presented in this article, junction tree model is a generalization of belief propagation that is correct for arbitrary graphs. We proposed localization using generalized belief propagation based on junction tree method and nonparametric approximation of this algorithm (NGBP-JT). Our main goal was to solve the problem with loops with some acceptable computational cost and we achieved it using NGBP-JT approach. We can conclude that this algorithm could provide higher accuracy with acceptable computational cost. The main open direction for future work is generalizing this algorithm for an ad-hoc network. This would probably require additional computation necessary for “construction” of junction tree cliques within the network. Moreover, communication cost has to be considered since it’s obvious that we can not exchange thousands of particles without any compression. This will be a part of our future research.

Acknowledgment

This work has been performed in the framework of the ICT project ICT-217033 WHERE, which is partly funded by the European Union and partly by the Spanish Education and Science Ministry under Grant TEC2007-67520-C02-01/02/TCM. Furthermore, we thank partial support by program CONSOLIDER-INGENIO 2010 CSD2008-00010 COMONSENS.

References