Pseudo-Junction Tree Method for Cooperative Localization in Wireless Sensor Networks

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

Abstract – *Nonparametric belief propagation (NBP) is well-known probabilistic method for cooperative localization in sensor networks. However, due to the double counting problem, NBP convergence is not guaranteed in the networks with loops or even if NBP converges, it could provide us less accurate estimates. The well-known solution for this problem is nonparametric generalized belief propagation based on junction tree (NGBP-JT). However, there are two problems: how to efficiently form the junction tree in an arbitrary network, and how to decrease the number of particles while keeping the good performance. Therefore, in this paper, we propose the formation of pseudo-junction tree (PJT), which represents the approximated junction tree based on thin graph. In addition, in order to decrease the number of particles, we use a set of very strong constraints. The resulting localization method, NGBP based on PJT (NGBP-PJT), overperforms NBP in terms of accuracy and communication cost in any arbitrary network.*

Keywords: cooperative localization, belief propagation, nonparametric generalized belief propagation, clique tree, junction tree, wireless sensor networks.

1 Introduction

We consider the case in which some small number of *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. Typical measurements techniques [1] are time of arrival (TOA), time difference of arrival (TDOA), received signal strength (RSS) and angle of arrival (AOA).

Nonparametric belief propagation (NBP) [2] is well-known probabilistic method for cooperative localization in sensor networks. It is capable to provide information about location estimation with appropriate uncertainty and to accommodate non-Gaussian distance measurement errors. This is the main advantage of NBP comparing with well-known deterministic methods [1]. However, due to the *double counting* problem, NBP convergence is not guaranteed in the networks with loops [3] or even if NBP converges, it could provide us less accurate estimates. Since we have already provided detailed description of this problem in [5], we'll skip it in this paper.

The well-known solution for the loopy graphs is generalized belief propagation based on junction tree (GBP-JT) method [4], which is a standard method for exact inference in graphical models. In our previous work [5], we applied nonparametric approximation of this method (NGBP-JT) for the localization in small-scale network and showed that it can overperform NBP in terms of accuracy. However, there remained two main problems: how to efficiently form the junction tree in an arbitrary network, and how to decrease the number of particles while keeping the good performance. Therefore, in this paper, we propose the formation of *pseudo-junction tree* (PJT), which represents the approximated junction tree based on thin graph. In addition, in order to decrease the number of particles, we use a set of very strong constraints: the measured distances and the bounded boxes. The resulting localization method, NGBP based on PJT (NGBP-PJT), overperforms NBP in terms of accuracy and communication cost in any arbitrary network.

The remainder of this paper is organized as follows. In Section 2, we provide the background on junction tree formation. In Section 3, we propose pseudo-junction tree formation. Localization using NGBP-PJT method for an arbitrary graph is proposed in Section 4. Simulation results are presented in Section 5. Finally, Section 6 provides some conclusions and future work perspective.

2 Junction Tree Formation

We start by describing the basics of graphical models. A graphical model is a probabilistic model for which a

graph denotes the conditional independence structure between random variables. There are two main types: *directed* graphical models (or Bayesian networks) and *undirected* graphical models (or Markov networks). For the cooperative localization problem, we use Markov networks.

An undirected graph $G = (V, E)$ consists of a set of nodes V that are joined by a set of edges E. A *loop* is a sequence of distinct edges forming a path from a node back to itself. A *clique* is a subset of nodes such that for every two nodes in clique, there exists an link connecting the two. A *tree* is a connected graph without any loops, and a *spanning tree* is an acyclic subgraph that connects all the nodes of the original graph. Regarding directed graphs, we define a *root node*, which is a node without parent, and *leaf node*, which is a node without children. In order to define a graphical model, we place at each node a random variable taking values in some space. Each edge in the graph represents the information about conditional dependency between two connected nodes. In this way, we reduced the complexity of computing the joint probability density function (pdf) of the graph. In case of cooperative localization, this random variable represents the 2D or 3D position, and each edge, which indicates that measurement is available, represents probabilistic information about distance. If we exclude the anchors nodes, the graph is obviously undirected.

Junction tree (JT) algorithm is a method for the exact inference in arbitrary graph. That can be proved elimination procedure [4]. It' based on *triangulated* graph, i.e., a graph with additional "virtual" edges so that every loop of length more than 3 has a chord. In triangulated graph, each 3-node loop (which is not part of any larger clique) represents 3-node clique, and each edge (which is not part of any 3-node clique) represents 2-node clique. If possible, larger cliques (> 3) should be avoided using optimal triangulation procedure. Using these cliques as hypernodes, we can define a *cluster graph* [6] by connecting each pair of cliques with minimum one common node (i.e., non-empty intersection). Using cluster graph, we can create a lot of clique trees, but just very few of them represent the junction tree. The junction tree is a *maximum spanning tree* of the cluster graph, with weights given by the cardinality of the intersections between cliques. It is already proved [6] that this is a way to satisfy the main property of the junction tree, the *running intersection property* (RIP). The RIP is satisfied if and only if each node, which is in two cliques C_i and C_j, is also in all cliques on the unique path between C_i and C_j. If the RIP is not satisfied for any node, there is no theoretical guarantee that the belief of that node in one clique is the same as its belief in another clique.

We illustrate the whole procedure in Fig. 1. We first triangulate the graph by adding the edge between nodes 2 and 5 (Fig. 1a). Then we form the cluster graph (Fig. 1b) with cliques $C_i(t, u, v)$ and separator sets $S_{ij}(q, r)$ ($S_{ij} = C_i \cap C_j$), where t, u, v are the nodes in the clique, and q, r are the *separator nodes*. Finally, any spanning tree represents the clique tree like in Fig. 1c and Fig. 1d. The tree in Fig. 1d is maximum spanning tree ($|S_{12}| > |S_{13}|$), so it represents the junction tree of the initial graph. Note that the tree in Fig. 1c does not satisfy RIP since the node 6, which is in C_1 and C_2, is not in C_3.

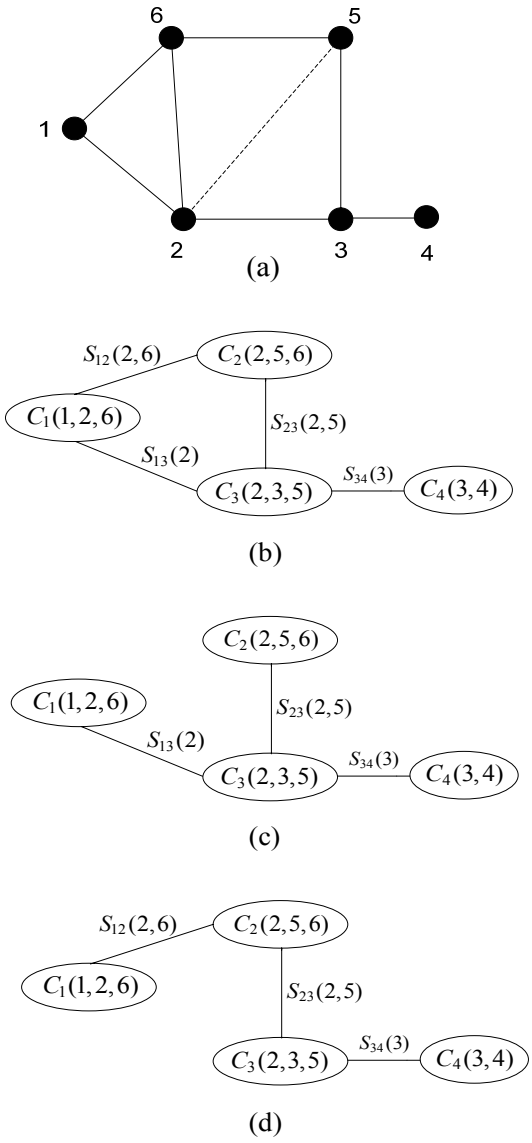


Figure 1: (a) Triangulated 6-node graph, and corresponding (b) cluster graph, (c) clique tree, and (d) junction tree.

The described procedure represents the exact formation of junction tree, also called *chordal graph* method. The main problem of this approach is the triangulation phase. Finding, a minimum triangulation, i.e., one

Algorithm 1 Searching for thin graph and cliques using modified Breadth First Search (BFS) method

Input: node list Q and root node $root$
Create more node lists: $Nodes, NewVisit \leftarrow Q$
Set current root: $r \leftarrow root$
Create list of neighbors for all nodes $n \in Q$: G_n
while $Nodes$ is not empty **do**
  for all nodes $t \in G_r$ **do**
    if $t \in Nodes$ **then**
      Remove t from $Nodes$
      Insert t in $WaitingRoots$
      Insert d_{rt} in T
    else if $d_{rt} \notin T$ *and* $r \in NewVisit$ **then**
      Insert d_{rt} in T
      Remove r from $NewVisit$
      Create 3-node cliques:
      for all $q \in PreviousRoots$ **do**
        if $\{d_{rq}, d_{tq}\} \in T$ **then**
          $C^{3nodes} \leftarrow \{r, t, q\}$
        end if
      end for
    end if
  end for
  Insert r in $PreviousRoots$
  Set current root: $r \leftarrow$ first unused node from $WaitingRoots$
end while
Create 2-node cliques C^{2nodes}: each edges in T which is not subset of C^{3nodes}
Output: thin graph $\{Q, T\}$ and cliques $C = C^{2nodes} \cup C^{3nodes}$

where the largest clique has minimum size, is *NP*-hard problem due to the number of permutations that must be checked. Of course, there exist approximate methods which are less expensive, but still too costly. For more details, see Chapter 10 in [6].

3 Pseudo-Junction Tree Formation

Due to the high complexity of the optimal junction tree formation, it's necessary to find some approximation that will be suitable for localization scenario. Thus, we are going to make the following assumptions:

(a) The number of cliques should be reasonable (i.e., in the order of number of nodes).

(b) In order to decrease the dimensionality of the problem, each clique will include no more than 3 nodes.

(c) Since the triangulation is expensive procedure, we are going to avoid it, even if it causes the break of RIP for some small percentage of the nodes.

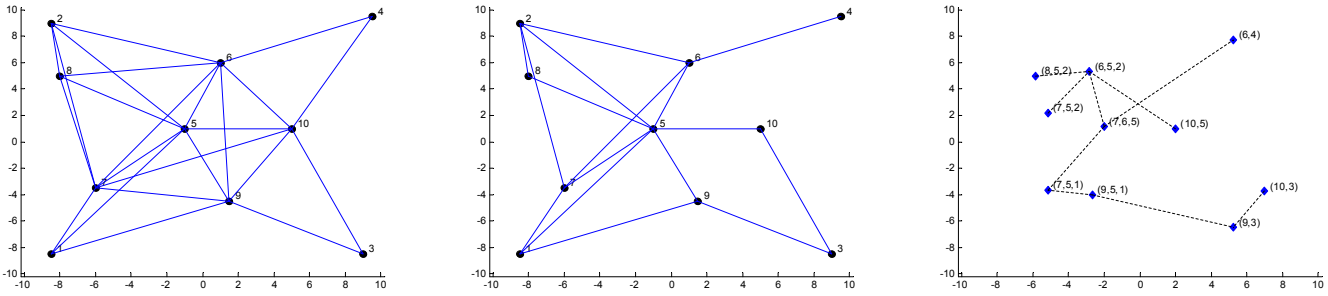
After these approximations, the final result represents, strictly speaking, the clique tree. However, since it is very close to junction tree (measured by percentage of nodes that satisfies RIP), we call it *pseudo-junction tree* (PJT).

Algorithm 2 Pseudo-Junction tree formation using Prim's algorithm

Input: node list Q and cliques C
Create weighted cluster graph:
for all pairs $\{i, j\}$ **do**
  $Weights(i, j) = |C_i \cap C_j|$
end for
Insert random root clique in $CurrentList$
while $|CurrentList| < |C|$ **do**
  Choose edge $\{m, n\}$ with maximal weight, such that C_m is in $CurrentList$ and C_n is not
  Insert C_n in $CurrentList$
  Insert edge $\{m, n\}$ in D
end while
Output: Pseudo-junction tree $\{C, D\}$

In order to satisfy the conditions (a) and (b), we need to decrease the number of edges in the graph by formation of *thin graph*. That can be easily done using modified version of *breadth first search* (BFS) method. Standard BFS method [7] begins at randomly chosen 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. Thus, the final result of BFS is spanning tree. 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.

Nevertheless, the spanning tree is very coarse approximation of the original graph since it excludes a lot of edges from the graph. For example, in any spanning tree, one communication failure breaks the graph into the two parts. As a consequence, we need more spanning trees in order to have reasonable accurate inference in graphical models. Therefore, we modify standard BFS method by permitting each root node to make the additional visit to the node that was already visited by some of the previous roots. All edges found by first and second visit, together with all nodes from original graph, represent the thin graph. In addition, the second visit will automatically form a loop, so we use it to form 3-node clique. The 2-node cliques can be found easily by taking all edges that appear in thin graph, but which are not already subset of any 3-node clique. The worst complexity is $O(v+e+v \cdot n_r)$, since for each of the additional visit we need to check all previous roots (i.e., n_r represents the average number of previous roots). The detailed pseudocode is shown in Alg. 1, and an example of original graph and corresponding thin graph is shown in Fig. 2a and Fig. 2b, respectively.



(a) (b) (c)

Figure 2: (a) Example of 10-node graph, and corresponding (b) thin graph, and (c) pseudo-junction tree

The main benefit of thin graph is that it mainly includes 3-node loops. The number of these loops, which is obviously always less than number of nodes, is nearly constant with respect to connectivity, so the number of cliques will be constant as well. On the other hand, the main drawback is that there exist the loops which include more that 3 nodes, but just very few of them. These loops should be triangulated, but we prefer to avoid it in order to keep reasonable complexity. Thus, for n-node loops ($n > 3$), we form maximum n 2-node cliques, using each edge (which is not already subset of any 3-node clique) of the loop as a clique.

Having defined cliques, we can form the cluster graph by connecting all pairs of cliques with non-empty intersection. As we already mentioned, the junction-tree, as well as pseudo-junction tree, is the maximum spanning tree of the cluster graph. It can be found using e.g., Prim's algorithm [8], as shown in Alg. 2. The Prim's algorithm is a method that finds a maximum (minimum) spanning tree for a connected weighted undirected graph. That means that the the total weight of all the edges in the final tree is maximized (minimized). In our case, the algorithm starts with a list (i.e., *CurrentList* in Alg. 2) which initially includes only randomly chosen root clique. At each step, among all the edges between the cliques in the list and those not in the list yet, it chooses the one with maximum weight and increases the list by adding the explored clique. Finally, it stops when all the cliques are spanned. The example of pseudo-junction tree is shown in Fig. 2c. The worst case complexity is $O(e \cdot log(v))$ [8], but in our case the weights are binary, so it will be significantly faster.

The BP/GBP methods are naturally distributed through the graph which means that there is no central unit which will handle all computations. Thus, the proposed pseudo-junction tree formation has to be done in a distributed way. It's already well-known that there is straightforward distributed way to form any spanning tree. Since this part is out of topic of this paper, we refer the reader to [9]. However, due to the modification of standard BFS algorithm, we explain here how to form the cliques in distributed way. This can be easily done, when root node visit one of the already visited nodes, by adding node's *IDs* in the (initially empty) clique list. This list must be broadcasted to all the nodes in the list, and to the next root node. Having defined all cliques, it remains to define the communication between neighboring cliques. Since, the separator sets, between each pair of neighboring cliques, are always non-empty, the separator nodes are responsible to perform the communication. For example, in Fig. 2c, the node 3 will request all the data from node 9, and upon receiving, it will send the data to node 10, and vice versa.

The approximations we made usually break the RIP for some small number of nodes. For instance, in the pseudo-junction tree in Fig. 2c, the node 10 (due to the non-triangulated 4-node loop: 3-9-5-10), and node 7 (due to the appearance of 4-node clique: 2-6-5-7) do not satisfy the RIP. Therefore, we don't have a guarantee that the belief of that node in one clique is the same as its belief in another clique [6]. Anyway, for the localization, this is not a problem since we incorporate additional constraints (see Section 4.1) for the initial set of particles. Regarding other applications, this algorithm could be used as well, but we recommend the potential users to test it before starting any implementation.

4 Nonparametric Generalized Belief Propagation

Generalized Belief Propagation based on junction tree (GBP-JT) is a standard method for exact inference in graphical model. This can be proved using *elimination* procedure. [4]. Given cliques C_i and its potentials $\psi_{C_i}(x_{C_i})$, and given the 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:

$$m_{ij}(x_{S_{ij}}) = \sum_{C_i \setminus S_{ij}} \psi_{C_i}(x_{C_i}) \prod_{k \in G_i \setminus j} m_{ki}(x_{S_{ki}}) \quad (1)$$

where $S_{ij} = S_{ji} = C_i \cap C_j$, and where G_i are the neighbors of clique C_i (including anchor nodes, which are not part of PJT). 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:

$$M_i(x_{C_i}) \propto \psi_{C_i}(x_{C_i}) \prod_{j \in G_i} m_{ji}(x_{S_{ji}}) \quad (2)$$

Finally, the single-node beliefs can be obtained via further marginalization. Equations (1), (2) represent GBP-JT algorithm which is valid for arbitrary graphs. The standard BP algorithm [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 that case, sets S_{ij} are single nodes, and marginalization is unnecessary.

In order to adapt GBP-JT to iterative scenario for cooperative localization, the previous equations, at iteration $m+1$, can be written as:

$$m_{ij}^{m+1}(x_{S_{ij}}) = \frac{1}{m_{ji}^{m}(x_{S_{ji}})} \sum_{C_i \setminus S_{ij}} M_i^m(x_{C_i}) \quad (3)$$

$$M_i^{m+1}(x_{C_i}) \propto \psi_{C_i}(x_{C_i}) \prod_{j \in G_i} m_{ji}^{m+1}(x_{S_{ji}}) \quad (4)$$

At the beginning, it's necessary to initialize $m_{ij}^1 = 1$, and $M_i^1 = \psi_{C_i}$. The clique potentials are given as a product of all single-node and pairwise potentials. The single-node potential (the prior) of node t is given by $\psi_t(x_t) = 1$ within the bounding box, and otherwise $\psi_t(x_t) = 0$. The bounding box of node t, created using approximated distances to anchors as constraints [10], represents the region of the deployment area where the node t is localized. The pairwise potential ψ_{tu}, which represents the probabilistic information about the distance between nodes t and u, is given by:

$$\psi_{tu}(x_t, x_u) = \begin{cases} p_v(d_{tu} - \|x_t - x_u\|), & if\ d_{tu} < R, \\ 0, & otherwise. \end{cases} \quad (5)$$

where d_{tu} represents the distance between nodes t and u, p_v the noise distribution of the measured distance, and R the transmission radius. The more general model, which incorporates the probability of detection, can be found in [2, 10].

Due to the high complexity, the presence of nonlinear relationships, and potentially highly non-Gaussian uncertainties, we use nonparametric (particle-based) approximation of GBP-JT method (NGBP-JT). Moreover, due to the problems explained in previous sections, we are going to use PJT instead of JT. Therefore, in following subsections, we propose NGBP based on PJT (NGBP-PJT) for any arbitrary network. Before continuing, we recommend the reader to read [5], where we provided the analysis of GBP-JT and NGBP-JT for the small-scale network.

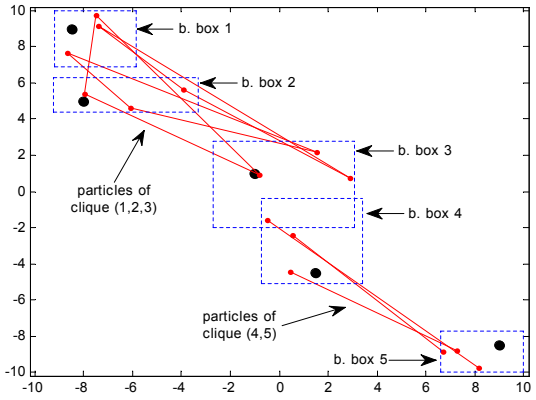


Figure 3: Illustration of initial particles from 2-node and 3-node cliques (to make plot visible, we set $N_C = 3$). The true node positions are marked with black circles.

4.1 Drawing particles from the cliques

Let us draw N_C weighted particles, $\{W_{C_i}^{k,m}, X_{C_i}^{k,m}\}$ ($k = 1, ..., N_C, m = 1$), from clique i. Since it's computationally very expensive to draw particles from $M_i^1 = \psi_{C_i}$, we need to find appropriate importance density function. Thus, for the initial particles, we are going to use two constraints: i) each particle of the node must be inside its bounding box, and ii) the distance between each pair of the nodes in clique should be close to the mean value of the measured distance. Taking this into account, our importance density function $q_{C_i}^m$ ($m = 1$) for clique i, which includes nodes t and u, is given by:

$$q_{C_i}^1(x_{C_i}) = q_{tu}^1(x_t, x_u) = \begin{cases} \psi_t(x_t)\psi_u(x_u), & if\ \|\mu_{tu} - \|x_t - x_u\|\| < 2\sigma \\ 0, & otherwise \end{cases} \quad (6)$$

where μ_{tu} is the mean value of measured distance, and σ is the standard deviation of the error distribution. The importance density for 3-node clique j, which includes nodes t, u, and v, is given by:

$$q_{C_j}^1(x_{C_j}) = \sqrt{q_{tu}^1(x_t, x_u) q_{tv}^1(x_t, x_v) q_{uv}^1(x_u, x_v)} \quad (7)$$

To draw clique particle, we need to draw node particles within its boxes and accept the particle if the constraint is satisfied. If not, we reject the sample, and try again. The weights of the particles can be easily computed by $W_{C_i}^{k,1} = \psi_{C_i}(X_{C_i}^{k,1}) / q_{C_i}^1(X_{C_i}^{k,1})$ and then normalized. In this way, we have created two types of particles: the edges (for 2-node cliques), and the triangles (for 3-node cliques). We illustrated initial set of particles in Fig. 3.

4.2 Computing messages

Having computed initial particles from beliefs, we can compute the particles from the messages. According to equation (3), we first need to marginalize the belief from previous iteration and then to divide the belief by the incoming message from previous iteration. Since all node particles within the clique have one common weight (e.g., $\{W_{C_i}^{k,m}, X_{C_i}^{k,m}\} = \{W_{C_i}^{k,m}, X_t^{k,m}, X_u^{k,m}\}$), we can simply pick the node which appears in the message (from clique that send the message), and compute the weight as reminder of (3). Since the separators sets can include one or two nodes, there exist 1-node and 2-node messages. For example, the weighted particles of the 1-node message from $C_i(t,u)$ to $C_j(t,v)$, at iteration $m+1$, are given by:

$$\{X_{S_{ij}}^{k,m+1}, W_{S_{ij}}^{k,m+1}\} = \{X_t^{k,m}, \frac{W_{C_i}^{k,m}}{m_{ji}^m(X_t^{k,m})}\} \quad (8)$$

The 2-node message can be found in same way. As we can see, we need the parametric form of the message m_{ji}^m, so we estimate it using spherically symmetric Gaussian kernel [11]. For 2-node message, it is very expensive to estimate the parametric form directly from high-dimensional (4D) particles. However, there is dependency between the nodes within the message (noisy distance), so it can be written as:

$$m_{ji}^m(x_t, x_u) = m_{ji}^m(x_t)\psi_{tu}(x_t, x_u) \quad (9)$$

Finally, the messages from any anchor a to an unknown node t, are simply given by parametric form: $m_{at}(x_t) = \psi_{at}(X_a, x_t)$.

4.3 Computing beliefs

According to (4), the belief of clique i is a product of its clique potential and all the messages coming into the clique. Before drawing particles, we need to solve two problems: i) the messages include information about different nodes within the clique, and ii) it is very expensive to draw samples from the product.

The first problem can be solved by filling the message with information about nodes which appear in destination clique, but not in the message. For example, for the message $m_{ij}^{m+1}(x_t, x_u)$, from $C_i(t,u,v)$ to $C_j(t,u,r)$, we can form the *joint message*: $M_{ij}^{m+1}(x_t, x_u, x_r) = m_{ij}^{m+1}(x_t, x_u)\psi_{tu}(x_t, x_r)\psi_{tu}(x_u, x_r)$. Using (9) and the definition of clique potential, the joint message can be written as:

$$M_{ij}^{m+1}(x_{C_j}) = m_{ij}^{m+1}(x_t)\psi_{C_j}(x_{C_j}) \quad (10)$$

where node t must be in appropriate separator set ($t \in S_{ij}$), and if $|S_{ij}| > 1$ we can pick one node randomly. Taking this into account, it can be easily proved, that equation (10) is valid for any clique. Thanks to the particles from standard messages, we already have few (one or two) node particles from each joint message. The remained node particles can be drawn by shifting given node particles in random direction for an amount which represents the observed distance, and by checking (only in case of 3-node clique) another distance constraint (more details in [5]). Of course, the weights of the particles from joint messages are equal to the weights of the particles from standard messages. However, due to the sample depletion, we *resample with replacement* [2] so as to produce the particles with same weights: $\{1/N_C, X_{ij}^{k,m+1}\}$.

Finally, due to the problem ii), instead of product, we make the sum of joint messages (i.e., using *mixture importance sampling* (MIS) [2]) Therefore, the final importance density for the belief of clique j, and corresponding particles, are respectively given by:

$$q_{C_j}^{m+1}(x_{C_j}) = \sum_{i \in G_j} M_{ij}^{m+1}(x_{C_j}) \quad (11)$$

$$\{W_{C_j,q}^{k,m+1}, X_{C_j,q}^{k,m+1}\} = \{\frac{1}{|G_j| \cdot N_c}, \bigcup_{i \in G_j} X_{ij}^{k,m+1}\} \quad (12)$$

Finally, we can find the set of particles from the beliefs $\{W_{C_j}^{k,m+1}, X_{C_j}^{k,m+1}\}$ $(k = 1, ..., N_C)$:

$$X_{C_j}^{k,m+1} = choose(\, X_{C_j,q}^{k,m+1}\Big|_{k=1}^{|G_j|}) \quad (13)$$

$$W_{C_j,corr}^{k,m+1} = W_{C_j,q}^{k,m+1} \frac{\prod_{i \in G_j} m_{ij}^{m+1}(X_{S_{ij}}^{k,m+1})}{q_{C_j}^{m+1}(X_{C_j}^{k,m+1})} \quad (14)$$

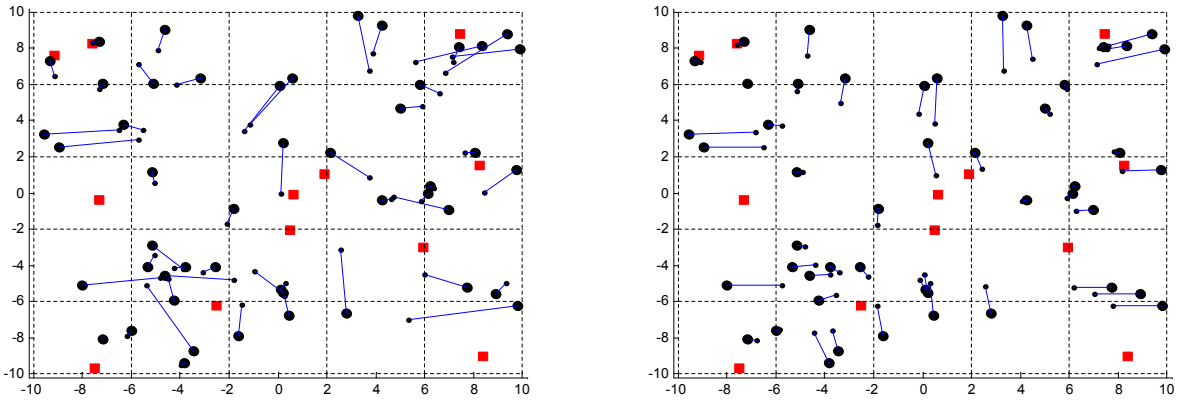
$$W_{C_j}^{k,m+1} = W_{C_j,corr}^{k,m+1} \cdot \psi_{C_j}(X_{C_j}^{k,m+1}) \prod_{\substack{t \in C_j \\ a \in G_j}} m_{at}(X_t^{k,m+1}) \quad (15)$$

where $W_{C_j,corr}^{k,m+1}$ is correction of the weights due to the MIS, $X_t^{k,m+1}$ particle from node t, and function *choose* chooses randomly one particle from $|G_j|$.

The final estimates of the nodes within the cliques, are given as the mean of the beliefs in last iteration. Since the most of the nodes appear in more than one clique, we simply average multiple estimates.

5 Simulation Results

We placed $N_a + N_u = 60$ nodes in 20m x 20m area. The minimum number of anchors, which are placed near the edges, is $N_{a,min} = 4$. This realistic constraint helps the unknown nodes near the edges which suffer from low connectivity. The rest of the anchors and the unknowns are randomly deployed within the area. The number of iteration is set to $N_{iter} = 3$, which means that any node/clique will receive all the information 3-hop away from itself. We assume that the distance is obtained using RSS measurements, so we choose the standard deviation σ_{dB} =5dB (i.e., the parameter of log-normal



(a) (b)

Figure 4: Illustration of results for the 60-node network: (a) NBP (b) NGBP-PJT. The anchors are marked with red squares, the unknowns with black circles, and the estimated locations with black dots.

distribution of the distance is $\sigma_{log(d)} = \sigma_{dB}/10n_p = 0.25$, where $n_p \approx 2$). All simulations are performed for $N_a = 6$ and $N_a = 12$ anchor nodes, and with respect to transmission radius R, which varies from 5m to 12m. All previous parameters are same both for NBP and NGBP-PJT. However, the last parameter, the number of particles, is set to $N_{nbp} = 100$ and $N_{pjt} = 210$ [1], so as to make the same computational time for both methods[2]. Finally, all simulations results represent the average over 20 random networks.

Using the defined scenario, we compared accuracy of NBP and NBP-PJT algorithms. The error is defined as Euclidean distance between true and estimated location. We illustrated the results of both methods in Fig. 4. It's obvious that, for almost all unknown nodes, NBP-PJT method overperforms the NBP method. Regarding RMS error and coverage (the percentage of located nodes with error less than predefined tolerance) shown in Fig. 5 and Fig. 6, the NBP-PJT significantly (5-10%) overperforms NBP, for all R and both values of N_a. It's also worth noting that number of anchors significantly affects accuracy and coverage.

To measure the communication cost, we count *elementary messages*, where one elementary message is defined as simple scalar data (e.g., one coordinate of one particle). According to Fig. 7, the NGBP-PJT not only overperforms NBP, but also this improvement is increasing as transmission radius increases. This is achieved thanks to the thin graph, which ensures low communication cost, and makes it nearly constant with respect to R. This feature provides us more precise

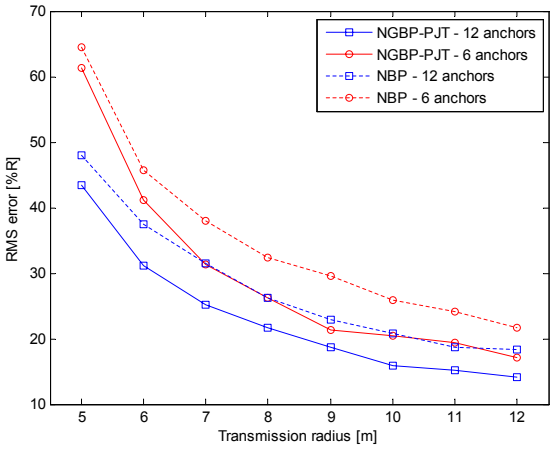


Figure 5: Comparison of the accuracy

information about battery life. Regarding number of anchors, the larger fraction of anchors decreases communication cost due to the simple message that each anchor has to transmit. However, it's interesting that for $R < 7$, the conclusion is opposite. This is caused by our constraint that the network is always connected (i.e., there is a path between each pair of the nodes), which is realistic constraint in the most applications. Basically, when the network is disconnected (it mostly happened for $R < 7$, $N_a = 12$), we replaced that network with connected one, and as a consequence increase the communication between nodes.

6 Conclusions and Future Work

In this paper, we presented NGBP-PJT, a novel probabilistic approach for cooperative localization in loopy

[1] Theoretically, $N_{pjt} = N_{nbp}^n$ for n-node clique, but thanks to the constraints that we included, this number is significantly less.

[2] We set the same comp. time for $R = 5$. For $R > 5$ the comp. time is even less for NGBP-PJT.

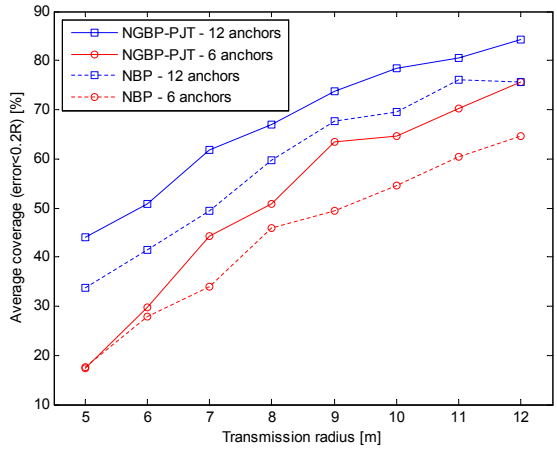


Figure 6: Comparison of the coverage

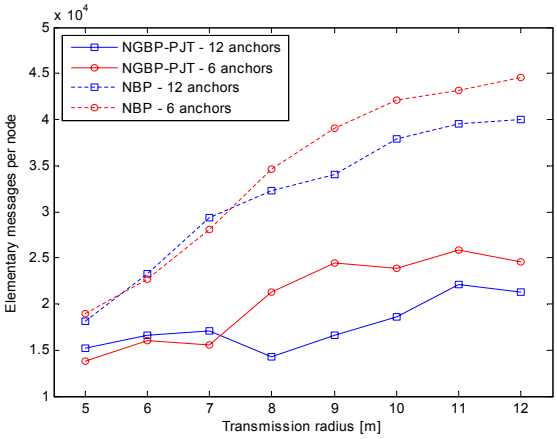


Figure 7: Comparison of the communication cost

networks. Since the exact formation of junction tree is not tractable, we proposed the formation of pseudo-junction tree (PJT), which represents the approximated junction tree based on thin graph. In addition, in order to decrease the number of particles for NGBP-PJT method, we used a set of very strong constraints. The resulting localization method, NGBP-PJT, overperforms NBP in terms of accuracy and communication cost in any arbitrary network. Moreover, thanks to the thin graph, NGBP-PJT has nearly constant communication cost with respect to transmission radius. There remain many open directions for the future work. A comparison with other BP-based methods for loopy networks could be very useful. We are currently working on two alternative solutions for this problem: NBP based spanning tree, and tree-reweighted NBP [12]. Furthermore, it's important to check if there is some cheaper (non-particle-based) message representation, which should be capable to handle all realistic uncertainties. Finally, target tracking using this method could be an interesting direction.

Acknowledgments

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

[1] N. Patwari, J.N. Ash, S. Kyperountas, A.O. Hero III, R.L. Moses and N.S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks", *Signal Processing Magazine, IEEE*, vol. 22, issue 4, pp. 54-69, July 2005.

[2] A.T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self- localization of sensor networks", *IEEE Journal On Selected Areas In Communications*, vol. 23, issue 4, pp. 809-819, April 2005.

[3] J.S. Yedidia, W.T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations", *Exploring artificial intelligence in the new millennium*, vol. 8, pp. 239-269, January 2003.

[4] M.I. Jordan and Y. Weiss, "Graphical model: probabilistic inference", *The Handbook of Brain Theory and Neural Networks*, 2nd edition, Cambridge, MA, MIT Press, 2002.

[5] V. Savic and S. Zazo, "Sensor localization using nonparametric generalized belief propagation in network with loops", in *IEEE Proc. of Information Fusion*, pp. 1966 - 1973, July 2009.

[6] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[7] D.A.Bader, K. Madduri, "Designing Multithreaded Algorithms for Breadth-First Search and st-connectivity on the Cray MTA-2", in *IEEE Proc. of Parallel Processing*, pp. 523-530, August 2006.

[8] X. Wang; X. Wang; D.M.Wilkes, "A Divide-and-Conquer Approach for Minimum Spanning Tree-Based Clustering", in *IEEE Transactions on Knowledge and Data Engineering*, vol.21, no.7, pp.945-958, July 2009.

[9] A.J. Mooij, N. Goga, W. Wesselink, "A distributed spanning tree algorithm for topology-aware networks", in *Proc. of the Conference on Design, Analysis, and Simulation of Distributed Systems*, 2004.

[10] V. Savic and S. Zazo, "Nonparametric boxed belief propagation for localization in wireless sensor networks", *In IEEE Proc. of SENSORCOMM*, pp. 520-525, June 2009.

[11] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York, 1986.

[12] M.J. Wainwright, T.S. Jaakkola and A.S. Willsky "Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching", in *Workshop on Artificial Intelligence and Statistics,*, January, 2003 .