EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Nonparametric generalized belief propagation based on pseudo-junction tree for cooperative localization in wireless networks

Vladimir Savic[1,2*] and Santiago Zazo[1]

Abstract

Non-parametric belief propagation (NBP) is a well-known message passing method for cooperative localization in wireless networks. However, due to the over-counting problem in the networks with loops, NBP's convergence is not guaranteed, and its estimates are typically less accurate. One solution for this problem is non-parametric generalized belief propagation based on junction tree. However, this method is intractable in large-scale networks due to the high-complexity of the junction tree formation, and the high-dimensionality of the particles. Therefore, in this article, we propose the non-parametric generalized belief propagation based on pseudo-junction tree (NGBP-PJT). The main difference comparing with the standard method is the formation of pseudo-junction tree, which represents the approximated junction tree based on thin graph. In addition, in order to decrease the number of high-dimensional particles, we use more informative importance density function, and reduce the dimensionality of the messages. As by-product, we also propose NBP based on thin graph (NBP-TG), a cheaper variant of NBP, which runs on the same graph as NGBP-PJT. According to our simulation and experimental results, NGBP-PJT method outperforms NBP and NBP-TG in terms of accuracy, computational, and communication cost in reasonably sized networks.

1 Introduction

Obtaining location estimates of each node position in wireless network as well as accurately representing the uncertainty of that estimate is a critical step for a number of applications, including sensor networks [1], cellular networks [2], and robotics [3]. We consider the case in which a small number of *anchor* nodes, obtain their coordinates via Global Positioning System or by installing them at points with known coordinates, and the rest, *unknown* nodes, must determine their own coordinates. Since we want to use energy-conserving devices, with lack the energy necessary for long-range communication, we suppose that all unknown nodes obtain a noisy distance measurements of the nearby subset of the other nodes (not necessarily anchors) in the network. Typical measurement techniques [1,4,5] are time of arrival (TOA), time difference of arrival, received signal strength (RSS), and angle of arrival. This localization technique is well known as *cooperative (or multi-hop)* localization.

Most of the state-of-the-art methods for cooperative localization estimate the point estimate of the sensor positions by applying classical least squares, multidimensional scaling, multilateration, or other optimization methods. These methods, also known as *deterministic methods* [1,6-8], lack a statistical interpretation, and as one consequence do not provide an estimate of the remaining uncertainty in each sensor location. On the other hand, Gaussian probabilistic methods (such as multilateration by Savvides et al. [9], or variational method by Pedersen et al. [10]) assume a Gaussian model for all uncertainties, which may be questionable in practice. Non-Gaussian uncertainty is a common occurrence in real-world sensor localization problems, where typically there is a fraction of highly erroneous (outlier) measurements. This problem can be solved using *non-parametric probabilistic* (or *Bayesian*) methods [11-14], which take into account uncertainty of the measurements. They estimate the particle-based approximation of the posterior probability density function (PDF) of the positions of all

*Correspondence: vladimir.savic@liu.se
[1]Signal Processing Applications Group, Universidad Politécnica de Madrid, Madrid, Spain
[2]Department of Electrical Engineering (ISY), Linköping University, Linköping, Sweden

unknown nodes, given the likelihood and a prior PDF of the positions of all unknown nodes. However, the main drawback of these methods is high complexity of marginalization of the joint posterior PDF, especially in large-scale networks. Nevertheless, an appropriate factorization of the joint PDF using some message-passing technique make these methods tractable. Non-parametric belief propagation (NBP), proposed by Ihler et al. [11,12], is a well-known particle-based message passing method for cooperative localization in wireless networks. It is capable to provide information about location estimation with appropriate uncertainty, to accommodate nonlinear models, and non-Gaussian measurement errors.

However, due to the *over-counting* problem in the networks with loops, NBP's convergence is not guaranteed, and its estimates are typically less accurate [15]. Our previous proposals, using NBP based on spanning trees [16] and uniformly-reweighted NBP [17], can mitigate this problem in highly connected networks, but with very small benefit comparing with NBP. Another solution is generalized belief propagation based on junction tree (GBP-JT) method [18], which is a standard method for the exact inference in graphical models. In [19], non-parametric generalized belief propagation based on junction tree (NGBP-JT) has been applied for the localization in a small-scale network, where it has been showed that it can outperform NBP in terms of accuracy, but with an additional cost. However, there remained two main problems: (i) how to efficiently form the junction tree in an arbitrary network, and (ii) how to decrease the number of particles. Therefore, in this article, we propose non-parametric generalized belief propagation based on pseudo-junction tree (NGBP-PJT). The main difference comparing with the standard method is 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 high-dimensional particles, we use a more informative importance density function, and reduce the dimensionality of the messages. As by-product, we also propose NBP based on thin graph (NBP-TG), a cheaper variant of NBP, which runs on the same graph as NGBP-PJT. According to our simulation and experimental results (using measurements from indoor office environment), NGBP-PJT method outperforms NBP and NBP-TG in terms of accuracy, computational, and communication cost in reasonably sized networks. On the other hand, the main drawback of this method is the high cost in large-scale networks.

The remainder of this article is organized as follows. In Section 2, we provide the background on graphical models, correctness of belief propagation, and junction tree formation. In Section 3, we propose an algorithm for PJT formation. Cooperative 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 proposals for the future work. The summary of notation is provided in Table 1.

2 Background and related work

2.1 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 (also known as *Markov random field*).

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

Table 1 Summary of notation

C_i	Clique i
S_{ij}	Separator set between cliques C_i and C_j
G_t	Set of neighbors of node t
G_{C_i}	Set of neighbors of C_i
x_t	Random variable for position of node t
X_a	Position of anchor node a
x_{C_i}	Random variable for position of C_i
$x_{S_{ij}}$	Random variable for position of S_{ij}
d_{tu}	Distance between nodes t and u
p_v	Noise distribution of the measured distance
R	Transmission radius
$\psi_t(x_t)$	Single potential (prior) of node t
$\psi_{tu}(x_t, x_u)$	Pairwise potential between nodes t and u
$\psi_{C_i}(x_{C_i})$	Potential of clique i
$M_t(x_t)$	Belief of node t
$m_{tu}(x_u)$	Message from node t to node u
$M_i^m(x_{C_i})$	Belief of C_i at iteration m
$m_{ij}^m(x_{S_{ij}})$	Message from C_i to C_j at iteration m
$M_{ij}^m(x_{C_j})$	Joint message from C_i to C_j at iteration m
$q_{C_i}^m(x_{C_i})$	Importance density function of C_i at iteration m
$X_{C_i}^{k,m}$	Particle k from $M_i^m(x_{C_i})$
$W_{C_i}^{k,m}$	Weight of particle k from $M_i^m(x_{C_i})$
$X_{S_{ij}}^{k,m}$	Particle k from $m_{ij}^m(x_{S_{ij}})$
$W_{S_{ij}}^{k,m}$	Weight of particle k from $m_{ij}^m(x_{S_{ij}})$

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 case of cooperative localization, each random variable represents the 2D (or 3D) position, and each edge, which indicates that the measurement is available, represents the likelihood function of that measurement. If we exclude the anchors nodes, the graph can be considered as undirected.

2.2 Correctness of belief propagation

In the standard belief propagation (BP) algorithm (also known as sum-product), proposed by Pearl [20], the belief at node t, which represents the estimate of the posterior marginal PDF, is proportional to the product of the local evidence at that node $\psi_t(x_t)$, and all the messages coming into node t:

$$M_t(x_t) \propto \psi_t(x_t) \prod_{u \in G_t} m_{ut}(x_t) \tag{1}$$

where x_t is a random variable for the state of node t (e.g., 2D position), and G_t denotes the neighbors of node t. The messages are determined by the message update rule:

$$m_{ut}(x_t) = \sum_{x_u} \psi_u(x_u)\psi_{tu}(x_t, x_u) \prod_{g \in G_u \setminus t} m_{gu}(x_u) \tag{2}$$

where $\psi_{tu}(x_t, x_u)$ is the pairwise potential between nodes t and u. On the right-hand side, there is a product over all messages going into node u except for the one coming from node t. This product is marginalized in order to form the particular information that we want to send to the destination node. In case of continuous functions, the sum over x_u have to be replaced with the integral.

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 [15] that each message needs to be computed only once for tree-like graphs, meaning that the whole computation takes a time proportional to the number of links in the graph, which is significantly less that the exponentially large time that would be required to compute the marginal PDFs naively. In other words, BP is a way of organizing the global computation of marginal beliefs in terms of smaller local computations. For the localization problem, this is not sufficient, so we need to represent the messages and beliefs in non-parametric (particle-based) form, as done in [11]. The resulting method, NBP, is capable to approximate the posterior marginal PDFs in non-Gaussian form.

The BP/NBP algorithm does not make a reference to the topology of the graph that it is running on. However, if we ignore the existence of loops, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium [20]. One can find examples of loopy graphs, where, for certain parameter values, the BP/NBP algorithm fails to converge or predicts beliefs that are inaccurate. On the other hand, the BP/NBP algorithm could be successful in graphs with loops, e.g., error-correcting codes defined on Tanner graphs that have loops [21].

In order for BP/NBP to be successful, it needs to avoid *over-counting* [20,22], a situation in which the same evidence is passed around the network multiple times and mistaken for new evidence. Of course, this is not the case in tree-like graphs because a node can receive some evidence only through one path. In a loopy network, over-counting could not be avoided. However, BP/NBP could still lead to nearly exact inference if all evidence is over-counted in equal amounts. This could be formalized by *unwrapped network* [22] corresponding to a loopy network. The unwrapped network is a tree-like network constructed such that performing BP/NBP in the unwrapped network is equivalent to performing BP/NBP in the loopy network. The importance of the unwrapped network is that since it is tree-like, BP/NBP on it is guaranteed to give the correct beliefs. However, usefulness of this beliefs depends on the similarity between the probability distribution induced by the unwrapped network and the original loopy network. If the distributions are not similar, then the unwrapped network is not useful and the results will be erroneous as in original loopy network.

For the extensive analysis of this problem, we refer the readers to [15,22,23].

2.3 Junction tree formation

Junction tree (JT) algorithm is a method for the exact inference in arbitrary graphs. That can be proved by elimination procedure [18]. It is based on *triangulated* graph, i.e., a graph with additional "virtual" edges, which ensure 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. Larger cliques (> 3) should be avoided, but this is usually not possible even with the optimal triangulation procedure. Using these cliques as hypernodes, we can define a *cluster graph* [24] by connecting each pair of the cliques with minimum one common node (i.e., non-empty intersection). Using cluster graph, we can create a lot of clique trees, but just one (or very few) of them represent the JT. The JT is a *maximum spanning tree* of the cluster graph, with weights given by the cardinality of the intersections between cliques. It is already proved [24] that this is a way to satisfy the main property of the JT, 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 one node, there is no theoretical guarantee that its belief in one clique is the same as its belief in another clique.

We illustrate the whole procedure in Figure 1. We first triangulate the graph by adding the edge between nodes 2 and 5 (Figure 1a). Then we form the cluster graph (Figure 1b) with cliques $C_i(t,u,v)$ and the 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, such as ones in Figure 1c,d. The tree in Figure 1d is the maximum spanning tree ($|S_{12}| > |S_{13}|$), so it represents the JT of the initial graph. Note that the tree in Figure 1c does not satisfy RIP since the node 6, which is in C_1 and C_2, is not in C_3.

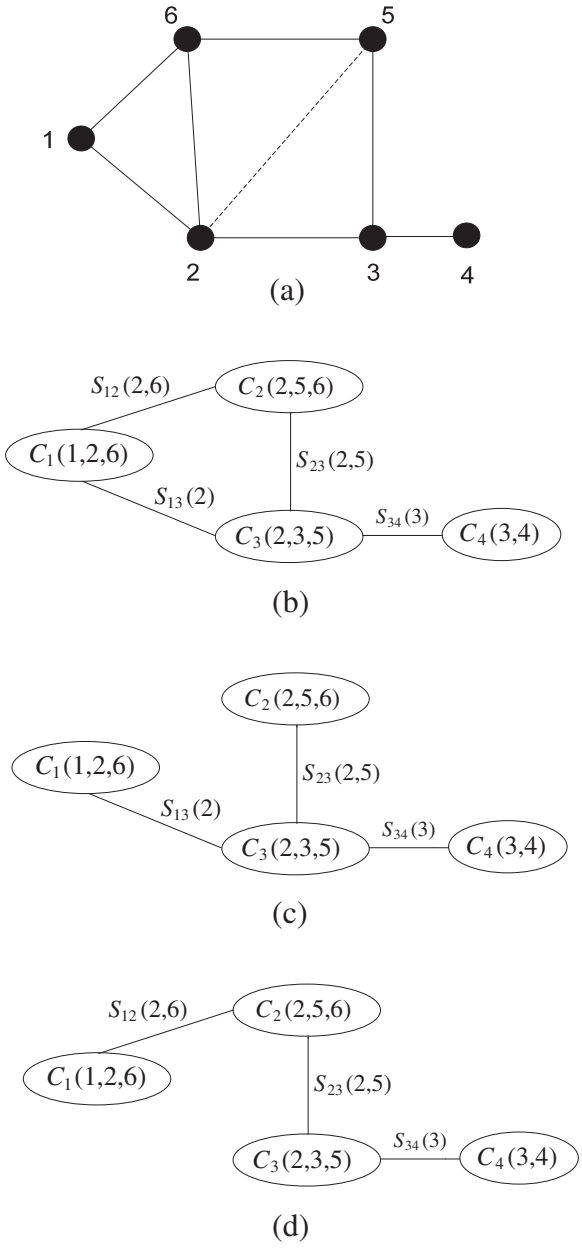

Figure 1 Illustration of junction tree formation (a) Triangulated 6-node graph, and corresponding (b) cluster graph, (c) clique tree, and (d) junction tree.

The described procedure represents the exact formation of JT, also called *chordal graph* method. The main problem of this approach is the triangulation phase. Finding, a minimum triangulation, i.e., one 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 an approximate methods (e.g., [25]) which are less expensive, but still too costly according to authors. For more details, see Chapter 10 in [24].

3 PJT

Due to the high complexity of the optimal JT formation, it is necessary to find some approximation that will be suitable for the cooperative localization problem. Therefore, our goal is to achieve the following.

(a) The number of cliques should be reasonable (i.e., in the order of number of nodes).
(b) In order to reduce the dimensionality of the problem, each clique should include no more than three 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 the junction tree (measured by the percentage of the nodes that satisfies RIP), we name it PJT.

3.1 Thin graph formation

In order to satisfy the conditions (a) and (b), we need to decrease the number of the edges in the graph by formation of *thin graph*. Assuming that each edge provides the same (or sufficiently similar) amount of information, it can be done using a modified version of *breadth first search* (BFS) method. The standard BFS method [26] begins at randomly chosen root node and explores all the neighboring nodes. Then each of those neighbors explores their unexplored neighbors, and so on, until all the nodes are explored. In this way, there will not be a loop in the graph because all the nodes will be explored just once. Thus, the final result of BFS is the 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 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 an additional visit to the node that was already visited by some of the previous roots. All edges found by first and second visits, along with all the nodes from the 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 the edges that appear in thin graph, but not in any 3-node clique. The worst complexity is $O(\nu + e + \nu \cdot (\nu - 1)) \approx O(\nu^2)$, since for each of the additional visit, we need to check all previous roots (all the nodes minus one, in the worst case). The detailed pseudocode is shown in Algorithm 1, and an example of the original graph and the corresponding thin graph are shown in Figure 2a,b, respectively.

Algorithm 1. Searching for thin graph and cliques using modified BFS method

Input: node list Q and root node *root*
Copy to 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 edge in T which is not subset of C^{3nodes}
Output: thin graph $\{Q, T\}$ and cliques $C = C^{2nodes} \cup C^{3nodes}$

The main benefit of the thin graph is that it mainly includes 3-node loops. The number of these loops, which is obviously always less than the total number of nodes, is nearly constant with respect to connectivity, so the number of cliques will nearly be constant as well. On the other hand, the main drawback is that there exist the loops which include more than three nodes.[a] 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. Another problem can be caused by the nodes which cannot determine their positions due to the possible non-rigidity of the thin graph (e.g., the nodes with less than three neighbors). However, these nodes can be still located since we bounded the estimate within its bounding box (see Section 4.1), created using original (not thin) graph. Therefore, the estimates will never be out

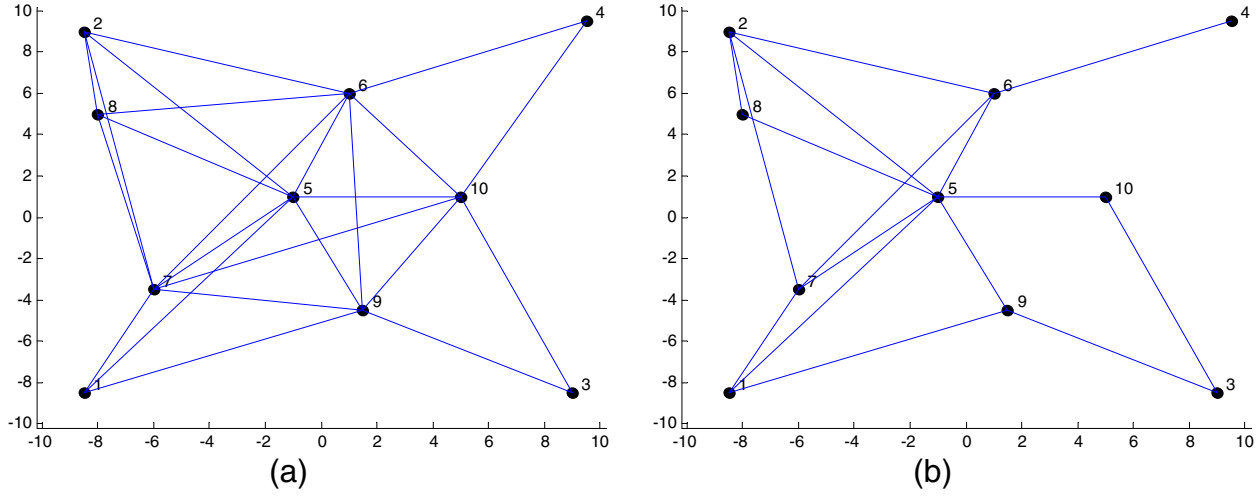

Figure 2 Illustration of thin graph formation (a) Example of 10-node graph, and (b) corresponding thin graph. The initial root is node 1.

of these boxes, which means that we ensured a coarse estimate in the worst case scenario. Finally, we note that anchor-unknown links are not excluded, so it would be useful if the anchors are placed as close as possible to the edges of the deployment area, where the leaf nodes are expected.

3.2 PJT formation

Having defined the cliques, we can form the cluster graph by connecting all pairs of the cliques with non-empty intersection (see Figure 3a). As we already mentioned, the JT, as well as PJT, is the maximum spanning tree of the cluster graph. It can be found using, e.g., Prim's algorithm [27], as shown in Algorithm 2. The Prim's algorithm is a method that finds a maximum (or minimum) spanning tree for a connected weighted undirected graph, meaning that the total weight of all the edges in the final tree is maximized (or minimized). In our case, the algorithm starts with a list (i.e., *CurrentList* in Algorithm 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 the maximum weight and increases the list by adding the explored clique. Finally, it stops when all the cliques are spanned. The example of PJT is shown in Figure 3b. The worst case complexity is $O(e \cdot log(v))$ [27], but in our case the weights are binary ($|S_{ij}| = 1$, or $|S_{ij}| = 2$), so the execution will be very fast.

Algorithm 2. PJT formation using Prim's algorithm

1: **Input**: node list Q and cliques C
2: Create weighted cluster graph:
3: **for all** pairs $\{i, j\}$ **do**
4: $\quad Weights(i, j) = |C_i \cap C_j|$
5: **end for**
6: Insert random root clique in *CurrentList*
7: **while** $|CurrentList| < |C|$ **do**
8: $\quad$ Choose edge $\{m, n\}$ with maximal weight, such that C_m is in *CurrentList* and C_n is not
9: Insert C_n in *CurrentList*
10: Insert edge $\{m, n\}$ in D
11: **end while**
12: **Output**: PJT $\{C, D\}$

The BP/GBP methods are naturally distributed through the network which means that there is no central unit (fusion center) which will handle all computations. Thus, the proposed PJT formation should be done in a distributed way. It is already well known that there is a straightforward distributed way to form any spanning tree, so we refer the readers to [28,29].

Having defined PJT, it remains to define the communication between neighboring cliques. Since the separator sets, between each pair of the neighboring cliques, are always non-empty, the separator nodes are responsible to perform the communication. Practically, these nodes represents the cluster heads. For example, in Figure 3b, 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.

Finally, the previous approximations will likely break the RIP for some small number of the nodes. For instance, in the PJT in Figure 3b, the node 10 (due to the non-triangulated 4-node loop: 3–9–5–10), and the node 7 (due to the appearance of 4-node clique: 2–6–5–7) do not satisfy the RIP. Therefore, we do not have a guarantee that the belief of that node in one clique is the same as its belief in another clique [24]. Nevertheless, for cooperative localization, this is not a problem since we used the

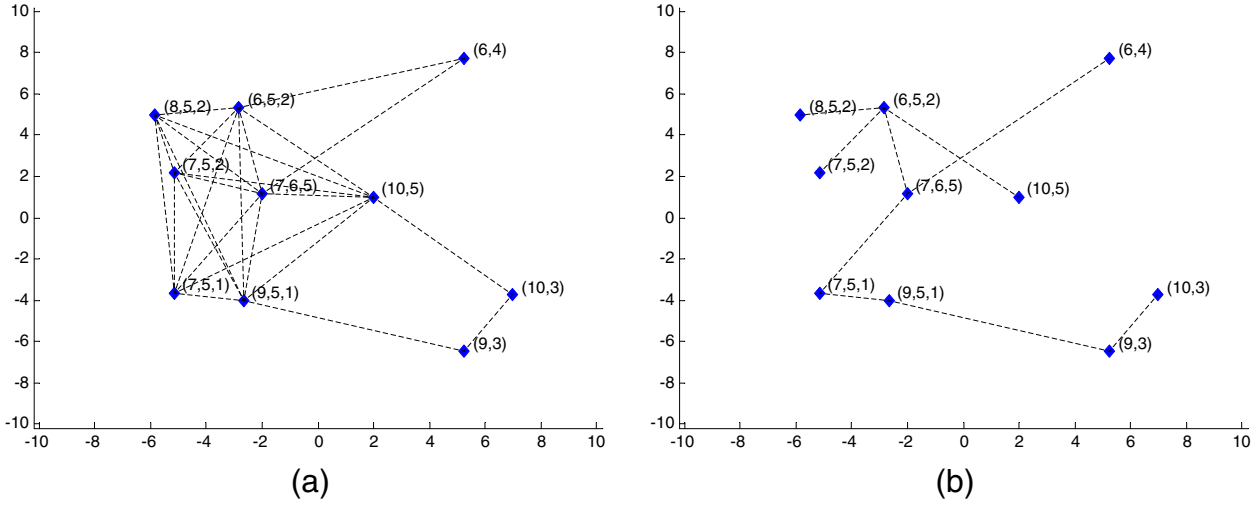

Figure 3 Illustration of PJT formation (a) Cluster graph based on thin graph from Figure 2b, and (b) corresponding PJT (maximum spanning tree of given cluster graph). Each clique is placed at the centroid of its node positions. The root clique is (10,5).

bounded boxes (see Section 4.1) for the initial set of particles. Regarding other applications, this method might be useful if all edges provide the same (or sufficiently similar) amount of information.

3.3 Possible alternatives

Although we provided a tractable solution for formation of the approximated junction tree, we cannot claim that it is an optimal one. In literature, there are available alternatives that could be (with some adaption) applied for this problem. For example, Dechter et al. [30] propose iterative join-graph propagation, which runs on the cluster graph with bounded cluster size, created without discarding any edges. Similar solution, thin junction tree with bounded cluster size, is available in [31]. However, for both approaches, a distributed implementation is not provided, so they cannot directly be applied for our problem. Finally, it is worth mentioning a distributed method [32], which creates a rigid subgraph from the fully connected (complete) graph in a tractable way. However, this method can be applied for cooperative localization, only if adapted for non-complete graphs.

4 Nonparametric generalized belief propagation

GBP-JT is a standard message passing method for the exact inference in graphical models. This can be proved using *elimination* procedure [18]. 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:

$$m_{ij}(x_{S_{ij}}) = \sum_{C_i \setminus S_{ij}} \psi_{C_i}(x_{C_i}) \prod_{k \in G_{C_i} \setminus j} m_{ki}(x_{S_{ki}}) \tag{3}$$

where $S_{ij} = S_{ji} = C_i \cap C_j$, and where G_{C_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:

$$M_i(x_{C_i}) \propto \psi_{C_i}(x_{C_i}) \prod_{j \in G_{C_i}} m_{ji}(x_{S_{ji}}) \tag{4}$$

Finally, the single-node beliefs can be obtained via further marginalization.

$$M_i(x_i) = \sum_{C_i \setminus i} M_i(x_{C_i}) \quad \text{for } i \in C_i \tag{5}$$

Equations (3), (4), and (5) represent GBP-JT algorithm which is valid for any arbitrary graphs. The standard BP algorithm [11] is a special case of GBP-JT, obtaining by noting that the original tree is already triangulated, and has only pairs of the nodes as cliques. In that case, sets S_{ij} are single nodes, and the marginalization is unnecessary.

In order to adapt GBP-JT to iterative scenario for cooperative localization, Equations (3), (4), at iteration $m+1$ can be written as

$$m_{ij}^{m+1}(x_{S_{ij}}) = \frac{1}{m_{ji}^{m}(x_{S_{ji}})} \sum_{C_i \setminus S_{ij}} M_i^m(x_{C_i}) \tag{6}$$

$$M_i^{m+1}(x_{C_i}) \propto \psi_{C_i}(x_{C_i}) \prod_{j \in G_{C_i}} m_{ji}^{m+1}(x_{S_{ji}}) \tag{7}$$

At the beginning, it is necessary to initialize $m_{ij}^1 = 1$, and $M_i^1 = \psi_{C_i}$. The clique potential ψ_{C_i} is given as a product of all single-node and pairwise potentials. The potentials of 2-node clique $C_i(t,u)$ and 3-node clique $C_j(t,u,v)$ are, respectively, given by

$$\psi_{C_i}(x_{C_i}) = \psi_{tu}(x_t, x_u)\psi_t(x_t)\psi_u(x_u) \tag{8}$$

$$\begin{aligned}\psi_{C_j}(x_{C_j}) = {} & \psi_{tu}(x_t, x_u)\psi_{tv}(x_t, x_v)\psi_{uv}(x_u, x_v)\psi_t(x_t) \\ & \times \psi_u(x_u)\psi_v(x_v)\end{aligned} \tag{9}$$

The single-node potential (the prior) of node t is given by

$$\psi_t(x_t) = \begin{cases} 1, & \text{within } b.box \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The bounding box (b. box) of node t, created using approximated distances to anchors (only 1-hop and 2-hop) as constraints [33], represents the region of the deployment area where the node t is localized. The pairwise potential ψ_{tu}, which represents the likelihood function about the distance between nodes t and u, ($\psi_{tu}(x_t, x_u) \propto p(d_{tu}|x_t, x_u)$) is given by

$$\psi_{tu}(x_t, x_u) = p_v(d_{tu} - \|x_t - x_u\|) \tag{11}$$

where d_{tu} represents the measured distance between nodes t and u, $p_v(\cdot)$ the noise distribution of the measured distance, and R the transmission radius. More general model, which incorporates the probability of detection, can be found in [11,33].

Regarding the model for mobile scenario, assuming that information is moving only forward in time, there are no loops between different time frames. Therefore, a dynamic model should be defined on the original graph with nodes. One option is to include this information using single-node potential, i.e.

$$\psi_t(x_{t,\tau}) = p(x_{t,\tau}|x_{t,\tau-1}), \tag{12}$$

where a dynamic model $p(x_{t,\tau}|x_{t,\tau-1})$ defines the possible positions of the unknown node $x_{t,\tau}$ in current time instant τ, given the estimated position from the previous time instant. It is also necessary to create the PJT at each time instant, except if the structure in the graph remains the same. All other computations (within the same time instant) are the same as in the static scenario. Thus, for clarity, we discard subscript τ in all equations, and focus

on the static scenario. More details on mobile positioning can be found in [13,14,34].

Due to the high complexity, the presence of nonlinear relationships, and potentially highly non-Gaussian uncertainties, GBP-JT method is not appropriate for cooperative localization [11]. Thus, we need to use 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 the arbitrary networks. Note that an analysis of NGBP-JT for the small-scale network has already been provided in [19].

4.1 Drawing particles from the cliques

Let us draw N_C weighted particles, $\left\{W_{C_i}^{k,m}, X_{C_i}^{k,m}\right\}$ $(k = 1,\ldots,N_C, m = 1)$, from clique i. Since it is 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 $C_i(t,u)$ is given by[b]:

$$q_{C_i}^1(x_{C_i}) = q_{tu}^1(x_t, x_u) = \begin{cases} \psi_t(x_t)\psi_u(x_u), & \text{if } \|\mu_{tu} - \|x_t - x_u\|\| < \delta \\ \epsilon, & \text{otherwise} \end{cases} \tag{13}$$

where μ_{tu} is the mean value of measured distance (we assumed that we obtained more measurements per link). The parameter δ should be chosen so as to encompass nearly the whole PDF. Otherwise, if we cut out significant part of the PDF, the final beliefs will be overconfident. For instance, if p_v is a Gaussian with standard deviation σ_d, $\delta = 3\sigma_d$ could be a good choice since it will encompass about 99% of the PDF. If the constraint is not satisfied, there is very small probability ϵ for the particle in that area.[c] Finally, it is straightforward to show, using (13), that the importance density function, for 3-node clique $C_j(t,u,v)$, can be found as

$$q_{C_j}^1(x_{C_j}) = \sqrt{q_{tu}^1(x_t,x_u)q_{tv}^1(x_t,x_v)q_{uv}^1(x_u,x_v)} \tag{14}$$

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 easily be computed by

$$W_{C_i}^{k,1} = \psi_{C_i}(X_{C_i}^{k,1})/q_{C_i}^1(X_{C_i}^{k,1}) \tag{15}$$

Then these weights (as well as all the weights in the following subsections) are normalized

$$W_{C_i}^{k,1} = W_{C_i}^{k,1} / \sum_k W_{C_i}^{k,1} \tag{16}$$

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 an initial set of particles in Figure 4.

4.2 Computing messages

Having computed the initial set of particles from the beliefs, we can compute the particles from the messages. According to Equation (6), we first need to marginalize the belief from previous iteration, then divide it by the incoming message from the 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 particles of separator nodes (from the clique that send the message), and compute the weight as reminder of (6). The separator sets can include one or two nodes, so there exist 1-node and 2-node messages. Therefore, the weighted particles of the 2-node message from $C_i(t,u,v)$ to $C_j(t,u,r)$, at iteration $m+1$, are given by

$$X_{S_{ij}}^{k,m+1} = \{X_t^{k,m}, X_u^{k,m}\} \tag{17}$$

$$W_{S_{ij}}^{k,m+1} = \frac{W_{C_i}^{k,m}}{m_{ji}^m(X_t^{k,m}, X_u^{k,m})} \tag{18}$$

The 1-node messages can be found in analog way. As we can see, we need to approximate the parametric form of

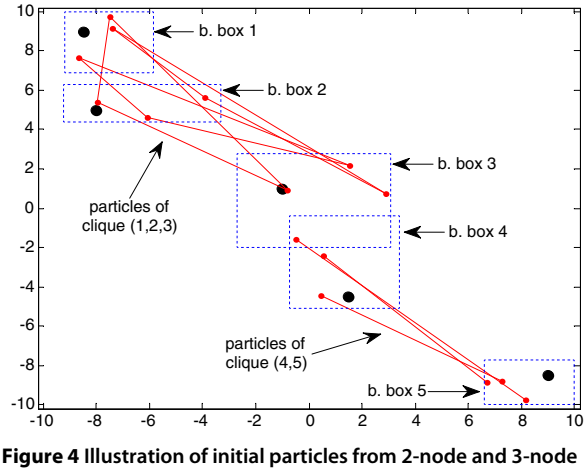

Figure 4 Illustration of initial particles from 2-node and 3-node cliques (for clarity, we show only 3 particles). Note that all the particles are originally high-dimensional (6D or 4D), but they can be shown in 2D space thanks to the distance constraint. The true node positions are marked with black circles.

the message m_{ji}^m, so we estimate it using spherically symmetric Gaussian kernel [12,35]. The bandwidth, parameter which controls the smoothness of this kernel density estimate (KDE), can be found using "rule of thumb" [12], or some advanced method [36]. In case of 2-node message, it is too complex to estimate the parametric form directly from high-dimensional (4D) particles. However, thanks to the dependency between the nodes within the message (the noisy distance), we can reduce the dimension of the message by

$$m_{ji}^m(x_t, x_u) = m_{ji}^m(x_t)\psi_{tu}(x_t, x_u)\psi_u(x_u) \tag{19}$$

Note that in PJT (in contrast to JT), there is always observed distance between each pair of the nodes within the clique (i.e., no additional edges added by triangulation). Thus, it is sufficient to transmit the particles over one node, and upon receiving, shift them in a random direction for the observed distance. Finally, the messages from any anchor a to any neighboring unknown node t, are simply given in the parametric form

$$m_{at}(x_t) = \psi_{at}(X_a, x_t) \tag{20}$$

where we assumed that the position of the anchor node is perfectly known (i.e., defined by Delta Dirac function). However, if anchors' positions are uncertain (as in [37]), the message can be computed in the same way as the messages from the unknown nodes.

4.3 Computing beliefs

According to (7), the belief of clique i is a product of its clique potential and all the messages coming into the clique. Before drawing the particles, we need to solve two problems: (i) the messages include information about different nodes within the clique, and (ii) it is intractable to draw the particles from the product.

The first problem can be solved by filling the message with the information about the nodes which appears in the 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*:

$$\begin{aligned} &M_{ij}^{m+1}(x_t, x_u, x_r) = \\ &m_{ij}^{m+1}(x_t, x_u)\psi_{tr}(x_t, x_r)\psi_{ur}(x_u, x_r)\psi_r(x_r) \end{aligned} \tag{21}$$

Taking Equations (19), (8), and (9) into account, the joint message can be written as

$$M_{ij}^{m+1}(x_{C_j}) = m_{ij}^{m+1}(x_t)\psi_{C_j}(x_{C_j}) \tag{22}$$

where node t must be in appropriate separator set ($t \in S_{ij}$), and if $|S_{ij}| > 1$, we can pick one node randomly. Thanks to the particles from the 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 a random direction for an amount which represents the observed distance, and by checking (only in case of 3-node clique) another distance constraint. The weights of the particles from the joint messages are equal to the weights of the particles from the standard messages. However, due to the sample depletion, we *resample with replacement* [38,39] so as to produce the particles with same weights: $\{1/N_C, X_{ij}^{k,m+1}\}$. The most of the particles, especially in case of small noise, will be the same. This can cause very poor representation of the beliefs. Therefore, to each of these particles, we add a small jitter ω drawn from p_ν

$$X_{ij}^{k,m+1} = X_{ij}^{k,m+1} + \omega \cdot [\cos(\theta) \quad \sin(\theta)] \tag{23}$$

where θ represents a random direction ($\theta \sim Unif[0, 2\pi)$). Finally, we solve problem (ii), by making the sum (instead of the product) of the joint messages (i.e., using *mixture importance sampling* (MIS) [11]). Therefore, the final importance density for the belief of clique j, and corresponding particles, are, respectively, given by

$$q_{C_j}^{m+1}(x_{C_j}) = \sum_{i \in G_{C_j}} M_{ij}^{m+1}(x_{C_j}) \tag{24}$$

$$\{W_{C_j,q}^{k,m+1}, X_{C_j,q}^{k,m+1}\Big|_{k=1}^{|G_{C_j}|}\} = \{\frac{1}{|G_{C_j}| \cdot N_c}, \bigcup_{i \in G_{C_j}} X_{ij}^{k,m+1}\} \tag{25}$$

We now find the set of particles from the beliefs $\{W_{C_j}^{k,m+1}, X_{C_j}^{k,m+1}\}(k = 1, \ldots, N_C)$:

$$X_{C_j}^{k,m+1} = \text{choose}\ (X_{C_j,q}^{k,m+1}\Big|_{k=1}^{|G_{C_j}|}) \tag{26}$$

$$W_{C_j,\text{corr}}^{k,m+1} = W_{C_j,q}^{k,m+1} \frac{\prod_{i \in G_{C_j}} m_{ij}^{m+1}(X_{S_{ij}}^{k,m+1})}{q_{C_j}^{m+1}(X_{C_j}^{k,m+1})} \tag{27}$$

$$W_{C_j}^{k,m+1} = W_{C_j,\text{corr}}^{k,m+1} \cdot \psi_{C_j}(X_{C_j}^{k,m+1}) \prod_{\substack{t \in C_j \\ a \in G_{C_j}}} m_{at}(X_t^{k,m+1}) \tag{28}$$

where $W_{C_j,\text{corr}}^{k,m+1}$ is the correction of the weights due to the MIS, $X_t^{k,m+1}$ particle from node t, m_{at} is the message from the anchor node a to unknown node t, and function *choose* chooses randomly one particle from $|G_{C_j}|$.

As a convergence parameter, we use approximated Kullback-Leibler (KL) divergence between the beliefs in two consecutive iterations, which is given by:

$$KL_j^{m+1} = \sum_k W_{C_j}^{k,m+1} \log[\, W_{C_j}^{k,m+1} / M_j^m(X_{C_j}^{k,m+1})] \tag{29}$$

where we used the approximation $M_j^{m+1}(X_{C_j}^{k,m+1}) \approx W_{C_j}^{k,m+1}$. The algorithm stops when KL_j^{m+1} (for all j) drops below the predefined threshold. However, it is also possible to predefine the number of iterations, given diameter of the graph and transmission radius. We choose the latter approach in simulations.

The final estimation of each node within the cliques is given as the mean of the particles from the belief in last iteration. Since the most of the nodes appear in more than one clique, we simply average multiple estimates. Other options are also possible, such as choosing the belief with the smallest entropy. We summarize the NGBP-PJT algorithm in Algorithm 3.

Algorithm 3. NGBP-PJT method for cooperative localization

for all *unknown nodes* **do**
  Obtain distance measurements to all neighbors
  Construct the bounded box
  Set all parameters to the initial values
end for
Form PJT using Algorithms 1 and 2
for all *cliques* **do**
  Draw initial particles from the importance density function (13) or (14)
  for $m = 1 : N_{iter}$ **do**
    Compute particles for outgoing messages via (17)-(18)
    Compute (eventual) messages from anchors via (20)
    Compute KDE of the messages
    Draw particles from the joint messages (22)
    Resample with replacement
    Add small jitter to all particles via (23)
    Compute particles from the beliefs via (24)-(28)
  end for
end for
Compute final location estimates

Finally, it is worth noting that a special case of NGBP-PJT method is NBP method based on thin graph (NBP-TG) assuming that the thin graph has only the pairs of the nodes as cliques. NBP-TG is very important by-product since it runs on the same graph as NGBP-PJT, which makes this method cheaper than NBP. It also helps to understand (as shown in the following section) how much the removed edges from the original graph affect the performance of the method.

5 Performance evaluation

5.1 Scenario

We assume that there are $N_a + N_u = 60$ nodes in a $20 \times 20\,\mathrm{m}^2$ area. Four anchors are placed near the edges. This, usually 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 iterations is set to $N_{\text{iter}} = 3$, which means that each node/clique will have available all the information 3-hop away from itself. The transmission radius is set to $R = 8\,\mathrm{m}$. Simulations are performed using $N_a = 6$ and $N_a = 12$ anchor nodes. We assume that the distance is obtained from RSS measurements using log-normal model, since this is usually the worst case scenario [1]. Thus, we choose $\sigma_{\text{dB}} = 5\,\text{dB}$ as standard deviation of RSS (i.e., the parameters[d] of log-normal distribution are $\mu = \log(d)$ and $\sigma = \sigma_{\text{dB}}/10n_p = 0.25$, where $n_p = 2$ is the path-loss exponent.[e]). Previous parameters are same both for NBP, NBP-TG, and NGBP-PJT. However, the number of particles is set to 100 (for NBP), 290 (for NBP-TG), and 210 (for NGBP-PJT), in order to make nearly the same computational time for all three methods (see Table 2). For the KDE of the messages, the bandwidth is found using "rule of thumb", which is the simplest option. Finally, the following simulation results represent the average over 20 Monte Carlo runs (in each of them, there are N_u estimates available). Note that all defined parameters are valid only if not otherwise stated in the following text.

5.2 Comparison of accuracy and convergence

Using the defined scenario, we compare the accuracy and the convergence of NBP, NBP-TG, and NBP-PJT algorithms. The error is defined as Euclidean distance between the true and the estimated location. First, we illustrate the results of these methods in Figure 5. We can see that NBP-PJT method significantly outperforms both NBP and NBP-TG methods, and also that NBP slightly outperforms NBP-TG.

Then, for randomly chosen node, we illustrate its initial and final belief in Figure 6. First, the initial beliefs of NBP and NBP-TG represent nearly uniform distribution within its bounded box, but the initial belief of NGBP-PJT (which is also within its bounded box) is not uniform due to the distance constraints within appropriate cliques (see (13)). Thus, the initial belief of NGBP-PJT is more informative than the belief of NBP. Second, we can see that the final NBP belief is the tightest, but this information is overconfident comparing with NBP-TG and NGBP-PJT. Since NBP-TG and NGBP-PJT run on the same graph, we can say that NBP-TG is overconfident, comparing

Table 2 Comparison of the computational cost (measured in MFlops)

R	NBP	NBP-TG	NGBP-PJT	PJT
5m	0.56	0.56	0.56	0.007
8m	0.71	0.59	0.67	0.011
12m	0.82	0.62	0.73	0.013

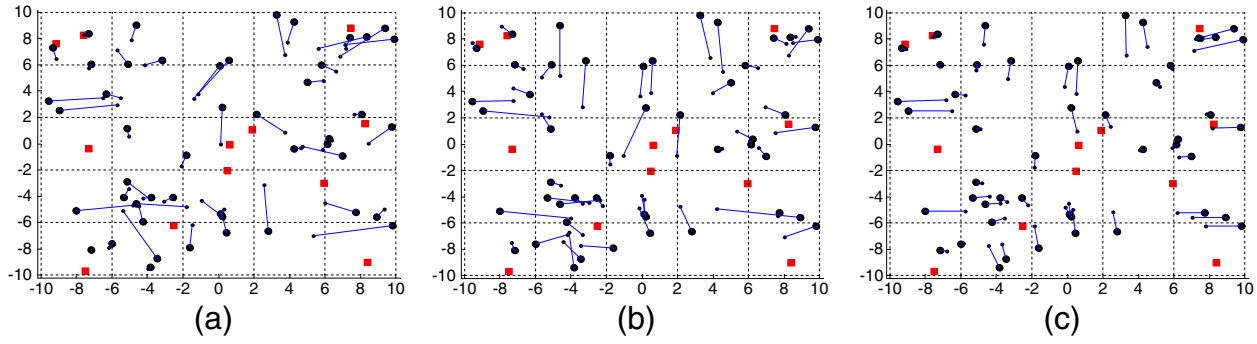

Figure 5 Illustration of results for the 60-node network. (a) NBP, **(b)** NBP-TG, and **(c)** NGBP-PJT. The anchors are marked with red squares, the unknowns with black circles, and the estimated locations with black dots.

with NGBP-PJT, due to the loops. Regarding comparison between NBP and NBP-TG, we note that the latter one has less number of the edges and less number of the loops. This increase the uncertainty, so NBP-TG provides more uncertain belief. The level of overconfidence can be also analyzed using the true position of the node. Hence, in case of overconfident belief, the true position can be located in the area with probability close to zero (as shown in Figure 6). Therefore, we can conclude that NGBP-PJT is less informative, but more trustful. Moreover, in order to obtain more precise conclusion about accuracy, we also consider cumulative distribution function (CDF) of the error in position. We can see in Figure 7 that NGBP-PJT outperforms all other methods in terms of any percentile.

Furthermore, we provide the analysis of the RMS error with respect to transmission radius. According to Figure 8, the NGBP-PJT significantly (5–10%) outperforms NBP and NBP-TG, for all R and both values of N_a. It is also worth noting that the number of anchors significantly affects the accuracy. For instance, NGBP-PJT with 6 anchors performs similar as NBP with 12 anchors. Therefore, given nearly the same accuracy, one can decrease the equipment cost by removing 6 anchors (which are usually very expensive). It is also interesting to note the performance difference between NGBP-PJT and NBP-TG (since they use different message passing, but the same graph),

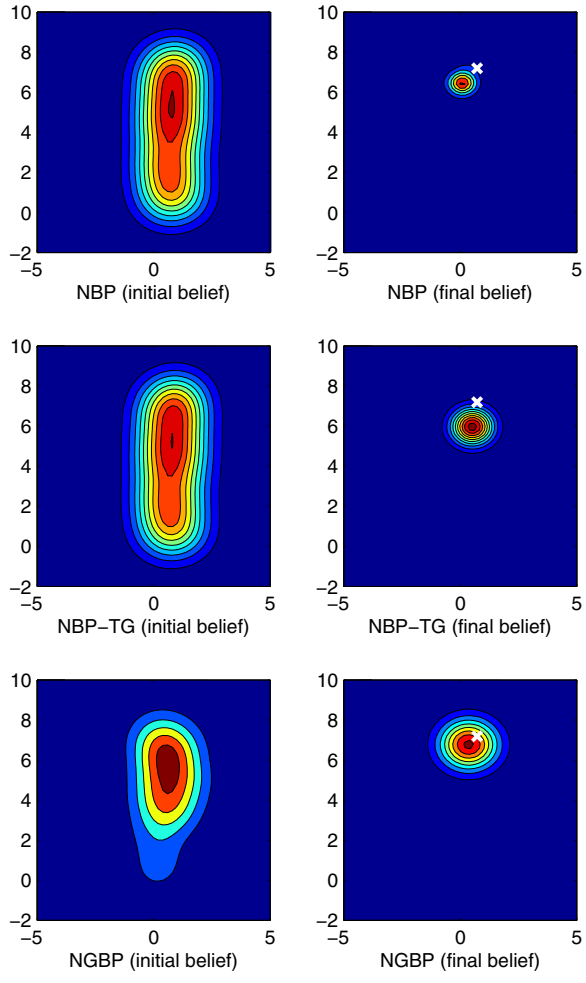

Figure 6 Comparison of NBP, NBP-TG and NGBP-PJT beliefs in the first and last iteration. True position of the node is marked with *X*.

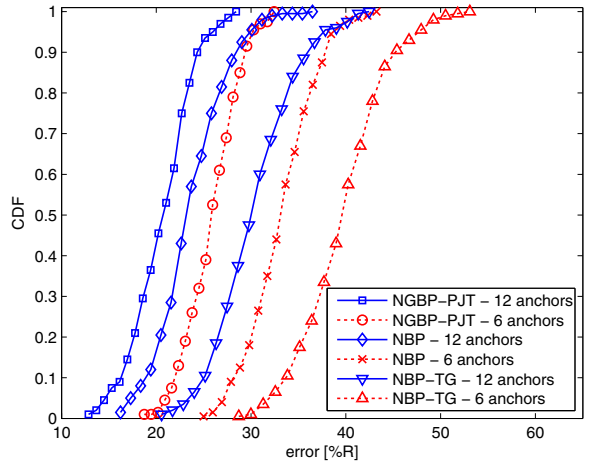

Figure 7 CDF of root mean square (RMS) error in position.

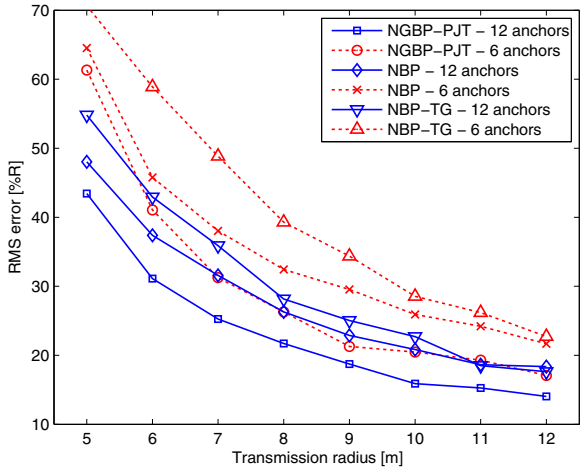

Figure 8 The effect of transmission radius on RMS error in position.

and the difference between NBP and NBP-TG (since they use the same message passing, but different graph).

Regarding convergence, we can see in Figure 9 that all algorithms converge sufficiently after second iteration. This is expected since we set $R = 8$ m, so almost all information is maximum 2-hop away from each clique. Finally, we can see that all algorithms, especially NBP-TG, cannot perfectly converge (i.e., $KL \to 0$) after reasonable number of iterations. This is, of course, caused by the existence of loops (for NBP and NBP-TG), and missing edges in thin graph (for NBP-TG and NGBP-PJT).

5.3 Comparison of computational and communication cost

As we already mentioned, we set the same computational cost for $R = 5$ m by choosing appropriate number of particles for all three methods. It was not possible to set the same cost for all methods since the cost is more sensitive to R in case of NBP method. On the other hand, NGBP-PJT and NBP-TG costs are less sensitive to R due to the nearly same number of edges with respect to (w.r.t.) R, in formed thin graph. We provide the average cost per node for different values of R in Table 2. We can see that the cost of NGBP-PJT is the same or less for all considered values of R. We can also see that the complexity of the PJT formation is neglectable comparing with full algorithms.

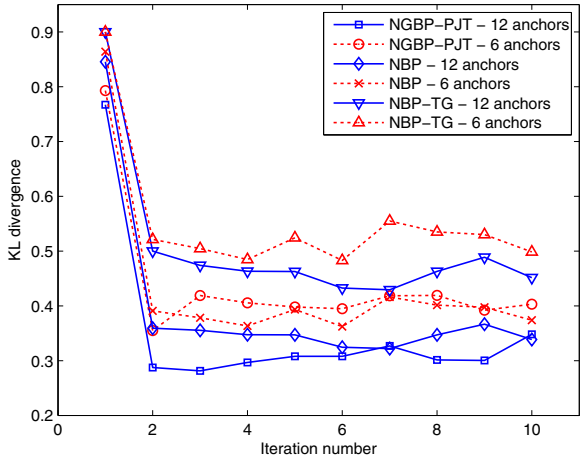

Figure 9 Comparison of KL divergence in each iteration.

Regarding communication cost, which is directly related with the battery life of the wireless devices, we provide a simplified[f] analysis by counting *elementary messages*, where one elementary message is defined as a scalar value (e.g., one coordinate of one particle). We will consider the effect of transmission radius and number of unknowns, since their variations obviously affect the cost. First, we analyze the cost of PJT formation (Algorithms 1 and 2). As we can see in Figure 10, it is a linear function of transmission radius, and a quadratic function of number of unknowns. Second, we analyze the cost of all considered algorithms w.r.t. R, for two different number of unknowns. According to Figure 11, we can conclude the following

- NGBP-PJT significantly outperforms NBP and NBP-TG methods, for reasonable number of unknowns.
- Comparing with NBP, the improvement of NGBP-PJT is increasing as transmission radius increasing. This is achieved thanks to the thin graph.
- The cost of NBP-TG is slightly less than NGBP-PJT due to the redundancy in PJT graph (i.e., when the same node appears in more than one clique).
- Increasing the number of unknowns will decrease the benefit of NGBP-PJT. This is caused by quadratic dependency of PJT formation w.r.t. number of unknowns. Using results from Figures 10b and 11, we estimate that NGBP-PJT will reach the same cost as NBP, for 140 unknown nodes.

Finally, we can conclude that the proposed NGBP-PJT method is cheaper for reasonably sized networks. However, it can also be cheap for very large-scale networks if the network is divided into regions, and one PJT created for each of them.

5.4 Experimental results

We now test NBP, NBP-TG, and NGBP-PJT using real RSS data obtained in indoor environment. The experiments are performed by Patwari et al. [40] and the data are available online [41]. They marked 44 points in a $14 \times 13\,\text{m}^2$ office area (see Figure 12), with many obstacles typical for that environment (cubicle walls, desks, computers, etc.). The measurement system includes one transmitter and receiver which uses a wideband direct-sequence spread-spectrum (DS-SS). The transmit power was 10 mW, and

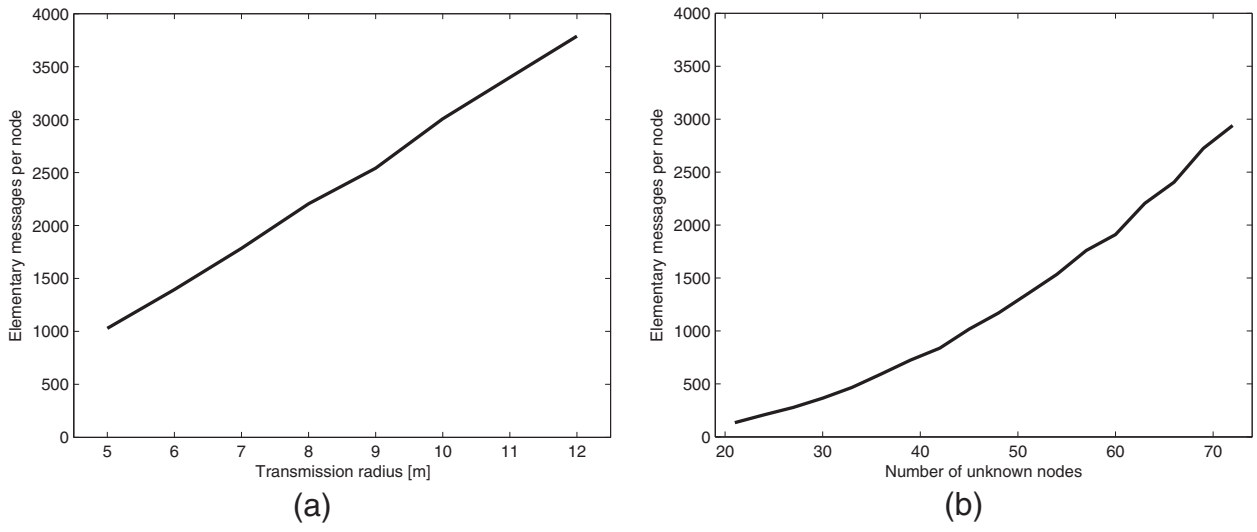

Figure 10 Communication cost for PJT formation w.r.t. (a) transmission radius, (b) number of nodes.

the center frequency 2443 MHz. The measurements are conducted by placing the transmitter and receiver at each pair of the points, and taking 5 measurements for each combination (totally, 9460 measurements). Then, for each measurement, TOA and RSS have been estimated, and for each link, the average has been found. More details about the experiments can be found in [40].

In this article, we use averaged RSS samples, shown in Figure 13. Taking $P_0(d_0 = 1m) = -37$ dBm, the parameters of the log-normal model can be found: $n_p = 2.3$ and $\sigma_{dB} = 3.92$ dB. Then, we compare the accuracy of NGBP-PJT, NBP-TG and NBP as a function of transmission radius (which varies from 5 to 10 m). The number of particles is the same as for the test in Section 5.2, and the number of anchors is set to $N_a = 5$ (marked as red squares in Figure 12). The results are averaged over 20 Monte Carlo runs (in each run, we use different seed for the particles, but keep the same network and the corresponding measurements). As we can see in Figure 14, the conclusion is the same as for the previous test (Section 5.2) based on synthetic data: NGBP-PJT outperforms NBP and NBP-TG, for all considered R. However, comparing Figures 8 and 14, we note that the values of the errors has changes.

6 Conclusion and future work

In this article, we presented NGBP-PJT, a novel message passing approach for cooperative localization in wireless networks. Since the exact formation of junction tree

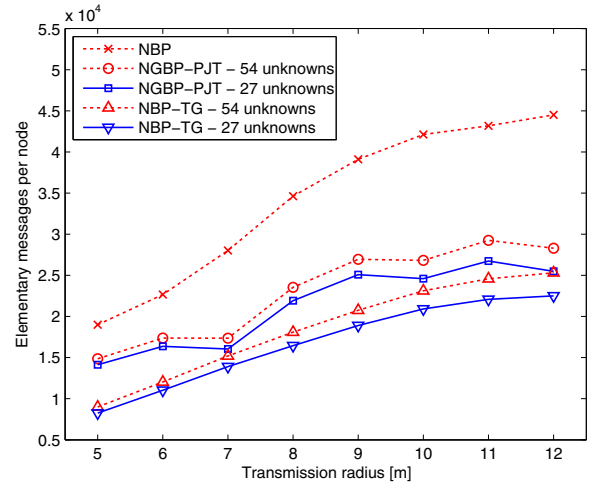

Figure 11 The effect of transmission radius on communication cost (for two different number of unknowns). NBP cost is the same for any number of unknowns, so we just plot one curve.

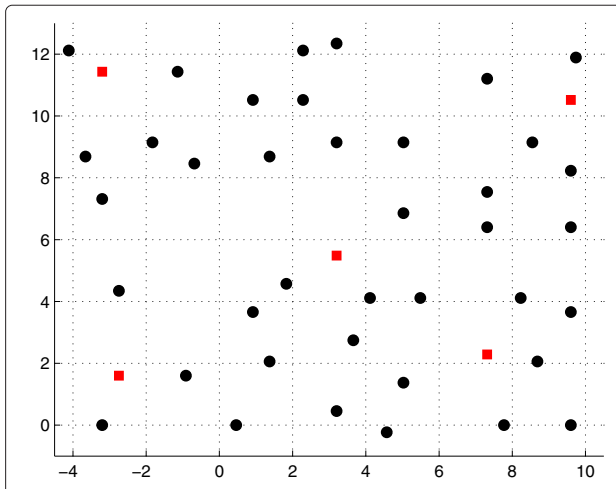

Figure 12 The deployment of the sensors in a $14 \times 13\,m^2$ (obstacles are not shown). The photo of the area can be found in [41].

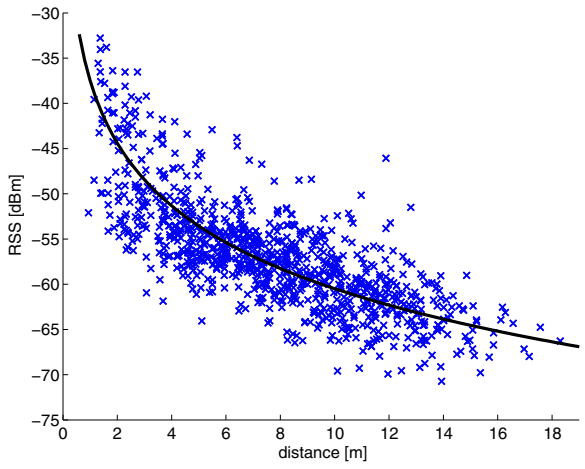

Figure 13 Obtained measurements (marked with X) and the corresponding log-normal model (black line).

is intractable, we proposed the formation of PJT, which represents the approximated JT based on thin graph. In addition, in order to decrease the number of particles for NGBP-PJT method, we proposed a more informative importance density function, and also reduce the dimensionality of the messages. As by-product, we also proposed NBP-TG, a cheaper variant of NBP, which runs on the same graph as NGBP-PJT. According to our simulation and experimental results, NGBP-PJT, outperforms NBP and NBP-TG in terms of accuracy, computational and communication cost in reasonably sized networks. In addition, NGBP-PJT beliefs are not overconfident as NBP beliefs, so we can obtain online more trustful information about the position uncertainty. Finally, all algorithms converge sufficiently after very small number of iterations, but the convergence is not perfect.

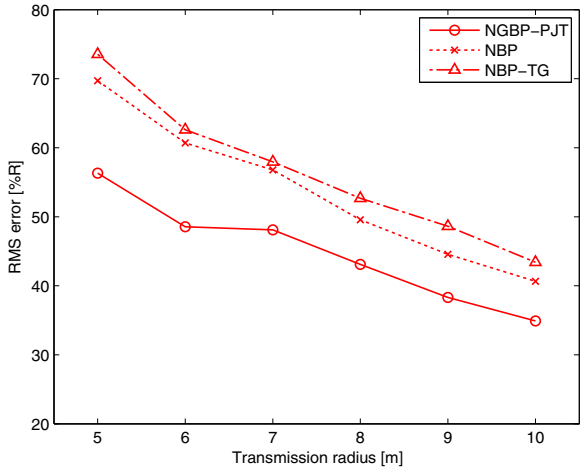

Figure 14 The effect of transmission radius on RMS error, for the network shown in Figure 12, and the measurement model shown in Figure 13.

There remain a number of open directions for the future work. One direction could be to find an alternative method (e.g., modified variants of the methods described in Section 3.3), which will be tractable in large-scale networks. It would be also useful to find in which graphs, and under which conditions, the benefit gained by using a NGBP-PJT instead of NBP outweighs the penalty caused by discarding edges. Moreover, an important research line is to investigate if there is some cheaper (non-particle-based) message representation, which should be capable to handle all realistic uncertainties. Finally, distributed target tracking in sensor network [42,43] could be an interesting direction since this application requires a number of sensor nodes with known (or estimated) positions.

7 Endnotes

[a]According to our empirical analysis, the number of these loops is relatively small (e.g., just one 4-node loop in Figure 2b).

[b]We implicitly assumed that $q^1_{C_i}(x_{C_i}) = 0$ if the state of one of the clique nodes is out of the deployment area.

[c]In practical circumstances, we can set $\epsilon = 0$.

[d]Note that these values do not represent the mean value and the standard deviation of the distance. They are respectively given by: $\mu_d = e^{\mu+\sigma^2/2}, \sigma_d = \mu_d\sqrt{e^{\sigma^2}-1}$. Consequently, these parameters are distance dependent.

[e]Typical values for n_p are between 2 and 6 [44]. For the distance estimation, the minimum value is the worst case.

[f]Exact communication cost can only be measured by knowing the hardware specifications, especially, the amount of the bytes in the package, number of reserved bytes in package, energy required to transmit a package, etc.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This study was supported in part by the FPU fellowship from Spanish Ministry of Science and Innovation; program CONSOLIDER-INGENIO 2010 under the grant CSD2008-00010 COMONSENS; the European Commission under the grant FP7-ICT-2009-4-248894-WHERE-2; the Swedish Foundation for Strategic Research (SSF) and ELLIIT.

Received: 27 October 2011 Accepted: 17 January 2013
Published: 9 February 2013

References

1. N Patwari, JN Ash, S Kyperountas, AO Hero III, RL Moses, NS Correal, Locating the nodes: cooperative localization in wireless sensor networks. IEEE Signal Process. Mag. **22**(4), 54–69 (2005)
2. AH Sayed, A Tarighat, N Khajehnouri, Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. IEEE Signal Process. Mag. **22**(4), 24–40 (2005)
3. D Fox, W Burgard, F Dellaert, S Thrun. Monte Carlo localization: efficient position estimation for mobile robots, in *Proceedings of the National Conference on Artificial Intelligence*, (July 1999)
4. F Gustafsson, F Gunnarsson, Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements. IEEE Signal Process. Mag. **22**(4), 41–53 (2005)

5. S Gezici, Z Tian, GB Giannakis, H Kobayashi, AF Molisch, HV Poor, Z Sahinoglu, Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. IEEE Signal Process. Mag. **22**(4), 70–84 (2005)
6. D Niculescu, B Nath, vol. 5. Ad hoc positioning system (APS), in *IEEE Proceedings of the GLOBECOM*, (November 2001), pp. 2926–2931
7. Y Shang, W Ruml, Y Zhang, M Fromherz, Localization from connectivity in sensor networks. IEEE Trans. Parallel Distrib. Syst. **15**(11), 961–974 (2004)
8. F Chan, HC So, W-K Ma, A novel subspace approach for cooperative localization in wireless sensor networks Using Range Measurements. IEEE Trans. Signal Process. **57**(1), 260–269 (2009)
9. A Savvides, H Park, MB Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems, in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Application*, (September 2002), pp. 112–121
10. C Pedersen, T Pedersen, BH Fleury. A Variational message passing algorithm for sensor self-localization in wireless networks, in *Proceedings of the International Symposium on Information Theory (ISIT)*, (July 2011), pp. 2158-2162
11. AT Ihler, JW Fisher I I I, RL Moses, AS Willsky, Nonparametric belief propagation for self- localization of sensor networks. IEEE J. Sel. Areas Commun. **23**(4), 809–819 (2005)
12. AT Ihler, Inference in sensor networks: graphical models and particle methods, PhD Thesis. Massachusetts Institute of Technology, June 2005
13. H Wymeersch, J Lien, MZ Win, Cooperative localization in wireless networks. Proc. IEEE. **97**(2), 427–450 (2009)
14. J Schiff, EB Sudderth, K Goldberg. Nonparametric belief propagation for distributed tracking of robot networks with noisy inter-distance measurements, in *IEEE Proceedings of the International Conference on Intelligent Robots and Systems*, (October 2009), pp. 1369–1376
15. JS Yedidia, WT Freeman, Y Weiss, Understanding belief propagation and its generalizations. Explor. Artif. Intell. New Millennium. **8**, 239–269 (2003)
16. V Savic, A Poblacion, S Zazo, M Garcia, Indoor positioning using nonparametric belief propagation based on spanning trees. EURASIP J. Wirel. Commun. Netw. **2010**, 1–12 (2010)
17. V Savic, H Wymeersch, F Penna, S Zazo. Optimized edge appearance probability for cooperative localization based on tree-reweighted nonparametric belief propagation, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (May 2011), pp. 3028–3031
18. MI Jordan, Y Weiss. 2nd edn. Graphical model: probabilistic inference, in *The Handbook of Brain Theory and Neural Networks* (MIT Press, Cambridge, MA, p. 2002
19. V Savic, S Zazo. Sensor localization using nonparametric generalized belief propagation in network with loops, in *IEEE Proceedings of Information Fusion*, (July 2009), pp. 1966–1973
20. J Pearl, *Probabilistic Reasoning in Intelligent Systems. Networks of plausible inference*. (Morgan Kaufmann, Burlington, MA, 1988)
21. BJ Frey, vol. 10. A revolution: belief propagation in graphs with cycles, in *Proceedings of Advances in Neural Information Processing Systems*, (1997), pp. 479–185
22. Y Weiss, Correctness of local probability propagation in graphical models with loops. Neural Comput. **12**(1), 1–41 (2000)
23. JM Mooij, HJ Kappen, Sufficient conditions for convergence of the sum-product algorithm. IEEE Trans. Inf. Theory. **53**(12), 4422–4437 (2007)
24. D Koller, N Friedman, *Probabilistic Graphical Models: Principles and Techniques*. (MIT Press, MA, 2009)
25. F Hutter, B Ng, R Dearden, *Incremental thin junction trees for dynamic Bayesian networks*. (Technical Report, Darmstadt University of Technology, 2004). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.8661
26. DA Bader, K Madduri. Designing multithreaded algorithms for breadth-first search and st-connectivity on the Cray MTA-2, in *IEEE Proceedings of the Parallel Processing*, (August 2006), pp. 523–530
27. X Wang, X Wang, DM Wilkes, A divide-and-conquer approach for minimum spanning tree-based clustering. IEEE Trans. Knowledge Data Eng. **21**(7), 945–958 (2009)
28. A Yoo, E Chow, K Henderson, W McLendon, B Hendrickson, UV Catalyurek. A scalable distributed parallel breadth-first search algorithm on Bluegene/L, in *Proceedings of the Supercomputing*, (November 2005)
29. AJ Mooij, N Goga. A distributed spanning tree algorithm for topology-aware networks, in *Proceedings of the Conference on Design, Analysis, and Simulation of Distributed Systems*, (2004)
30. R Dechter, K Kask, R Mateescu. Iterative join-graph propagation, in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, (2002)
31. A Chechetka, C Guestrin. Efficient principled learning of thin junction trees, in *Advances in Neural Information Processing Systems (NIPS)*, (2007)
32. TS Caetano, T Caelli, D Schuurmans, DAC Barone, Graphical models and point pattern matching. IEEE Trans. Pattern Anal. Mach. Intell. **28**(10), 1646–1663 (2006)
33. V Savic, S Zazo. Nonparametric boxed belief propagation for localization in wireless sensor networks, in *IEEE Proceedings of the SENSORCOMM*, (June 2009), pp. 520–525
34. L Mihaylova, D Angelova, DR Bull, NC Canagarajah, Localization of mobile nodes in wireless networks with correlated in time measurement noise. IEEE Trans. Mob. Comput. **10**(1), 44–53 (2011)
35. BW Silverman, *Density Estimation for Statistics and Data Analysis*. (Chapman and Hall, New York, 1986)
36. ZI Botev, *A novel nonparametric density estimator*. (Technical Report, The University of Queensland, Australia, 2006)
37. M Vemula, MF Bugallo, PM Djuric, Sensor self-localization with beacon position uncertainty. Signal Process (Elsevier). **89**(6), 1144–1154 (2009)
38. PM Djuric, JH Kotecha, J Zhang, Y Huang, T Ghirmai, MF Bugallo, J Miguez, Particle filtering. IEEE Signal Process. Mag. **20**(5), 19–38 (2003)
39. MS Arulampalam, S Maskell, N Gordon, T Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Trans. Signal Process. **50**(2), 174–188 (2002)
40. N Patwari, AO Hero, M Perkins, NS Correal, RJ O'Dea, Relative location estimation in wireless sensor networks. IEEE Trans. Signal Process. **51**(8), 2137–2148 (2003)
41. Wireless Sensor Network Localization Measurement Repository. http://web.eecs.umich.edu/~hero/localize/
42. AF Garcia-Fernandez, J Grajal, Asynchronous particle filter for tracking using non-synchronous sensor networks. Signal Process (Elsevier). **91**(10), 2304–2313 (2011)
43. PM Djuric, J Beaudeau, MF Bugallo. Non-centralized target tracking with mobile agents, in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, (May 2011), pp. 5928–5931
44. TS Rappaport, *Wireless Communications: Principles and Practice*, 2nd edn. (Prentice Hall PTR , Upper Saddle River, NJ, 2001)

doi:10.1186/1687-6180-2013-16
Cite this article as: Savic and Zazo: **Nonparametric generalized belief propagation based on pseudo-junction tree for cooperative localization in wireless networks.** *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:16.