Belief consensus algorithms for fast distributed target tracking in wireless sensor networks

Vladimir Savic, Henk Wymeersch, and Santiago Zazo

***Abstract*—In distributed target tracking for wireless sensor networks, agreement on the target state can be achieved by the construction and maintenance of a communication path, in order to exchange information regarding local likelihood functions. Such an approach lacks robustness to failures and is not easily applicable to ad-hoc networks. To address this, several methods have been proposed that allow agreement on the global likelihood through fully distributed belief consensus (BC) algorithms, operating on local likelihoods in distributed particle filtering (DPF). However, a unified comparison of the convergence speed and communication cost has not been performed. In this paper, we provide such a comparison and propose a novel BC algorithm based on belief propagation (BP). According to our study, DPF based on metropolis belief consensus (MBC) is the fastest in loopy graphs, while DPF based on BP consensus is the fastest in tree graphs. Moreover, we found that BC-based DPF methods have lower communication overhead than data flooding when the network is sufficiently sparse.**

***Index Terms*—Belief consensus, belief propagation, distributed target tracking, particle filtering, wireless sensor networks.**

I. Introduction

Distributed tracking in wireless sensor networks (WSN) [1] is an important task for many applications in which central unit is not available. For example, in emergency situations, such as a fire, a nuclear disaster, or a mine collapse, a WSN can be used to detect these phenomena. Once a phenomenon is detected (e.g., increased temperature or radioactivity), the sensors start to sense their neighbourhood and cooperatively track people and assets. As sensors are low-cost devices that may not survive during tracking, it is important to track in a manner that is fully robust to sensors failures, and in such a way that every sensor has the same belief of the target location. Then, the rescue team can access the estimates, even if just one sensor survives. As another potential application, sensor nodes can also serve as actuators, which perform a specific action (e.g., move towards the target) as a function of estimated target's position. In this case, to ensure compatible actions, a unified view of the target's position is crucial.

The traditional approach to target tracking is based on Kalman filtering (KF) [2]. However, due to nonlinear relationships and possible non-Gaussian uncertainties, a particle filter (PF) is preferred [3] in many scenarios. Therefore, the focus of this paper will be on PF-based distributed tracking. Many PF-based methods are based on the construction and maintenance of a communication path, such as a spanning tree or a Hamiltonian cycle. For example, in [4], low-power sensors pass the parameters of likelihood functions to high-power sensors, which are responsible to manage the low-power nodes. In [5], a set of uncorrelated sensor cliques is constructed, in which slave nodes have to transmit Gaussian mixture parameters to the master node of the clique. The master node performs the tracking, and forward estimates to another clique. In [6], a Markov-chain distributed PF is proposed, which does not route the information through the graph during tracking. However, it requires that each node knows the total number of communication links and the number of communication links between each pair of nodes, which can be obtained only by aggregating the data before tracking. In [7], the authors propose an incremental approach, in which the parameters of the likelihood are communicated from sensor to sensor in order to approximate the posterior of interest. Finally, there is also a different class of methods [8], [9] that maintain disjoint sets of particles at different nodes, and propagate them towards the predicted target position. These type of methods, also know as *leader-agent algorithms* (see [1] for an overview), lack robustness to failures, cause excessive delays due to the sequential estimation, and do not provide the estimates at each sensor without additional post-processing routing phase.

These problems can be solved if each node broadcasts observations until all the nodes have complete set of observations. Then, each node (acting like a fusion center) performs the tracking. This method, known as *data flooding* and used in non-centralized PF (NCPF) [10], is not scalable, but can be competitive in some scenarios. Other solutions consider distributed particle filtering (DPF) methods based on consensus algorithms [11]–[17]. In [11], the global posterior distribution is approximated with a Gaussian mixture, and consensus is applied over the local parameters to compute the global parameters. Similarly, [12], [13] use a Gaussian approximation instead of a Gaussian mixture, and [14] can use any distribution that belongs to an exponential family. Randomized gossip consensus was used in [15] for distributed target tracking. The main problem with these approaches

V. Savic was with the Signal Processing Applications Group, Universidad Politecnica de Madrid, Madrid, Spain, and he is now with the Dept. of Electrical Engineering (ISY), Linköping University, Sweden (e-mail: vladimir.savic@liu.se). H. Wymeersch is with the Dept. of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden (e-mail: henkw@chalmers.se). S. Zazo is with the Signal Processing Applications Group, Universidad Politecnica de Madrid, Madrid, Spain (e-mail: santiago@gaps.ssr.upm.es)

This work is supported by the Swedish Foundation for Strategic Research (SSF) and ELLIIT; Swedish Research Council (VR), under grant no. 2010-5889; the European Research Council, under grant COOPNET No. 258418; program CONSOLIDER-INGENIO 2010 under the grant CSD2008-00010 COMONSENS; the European Commission under the grant FP7-ICT-2009-4-248894-WHERE-2; and the FPU fellowship from Spanish Ministry of Science and Innovation. Part of this work was presented at the 2012 European Signal Processing Conference.

is that the global likelihood function is represented in the same parametric form as local likelihood functions, which is questionable in certain scenarios. In [16], [17], consensus is applied instead to the weights in the DPF, so that any likelihood can be represented. However, an issue that arises with these DPF approaches is that consensus can be slow. In a setting where the target moves, only a finite time is available to perform consensus [18], so the fastest possible method should be employed. A recent and detailed overview of DPF algorithms can be found in [1], but it does not analyze the effect of different consensus techniques on convergence.

In this paper, we compare five algorithms for target tracking using distributed particle filtering (DPF) based on belief consensus (BC):

1) standard belief consensus (SBC) [17];
2) randomized gossip (RG) [16];
3) broadcast gossip (BG) [19];
4) Metropolis belief consensus (MBC) [14]; and
5) one novel algorithm based on belief propagation (BP), which we earlier proposed in [20].

To the best of our knowledge, this is the first study where these methods are compared in a common setting. According to our simulation study, DPF-MBC is the fastest in loopy graphs, while DPF-BP is the fastest in tree graphs (typical for tunnel-like environments). Moreover, we found that BC-based DPF methods have lower communication overhead than data flooding only in sparse networks.

The rest of this paper is organized as follows. In Section II, we review centralized target tracking. In Section III, we describe five BC algorithms for PF-based distributed target tracking, including the novel based on BP. Simulation results are shown in Section IV. Finally, Section V provides our conclusions and suggestions for future work.

II. Overview of Centralized Target Tracking

We assume that there is a number of static sensor nodes with known positions and one moving target (e.g., a person or vehicle) in some surveillance area. The target may be passive or not willing to reveal its state, but the sensors are assumed to periodically make observations that depend on their relative position to the target. The goal of the WSN is to track the state (e.g., position and velocity) of the target. In this section, we describe a centralized approach to solve this problem, in which all the observations are collected by a sensor that acts as a fusion center.

A. System Model

The scenario under consideration is illustrated in Figure 1. There are N_s static sensors with known two-dimensional (2D) positions, $\mathbf{l}_n$ $(n = 1, 2, \ldots, N_s)$ and one mobile target with an unknown state $\mathbf{x}_t$ at time t. The goal of the WSN is to estimate $\mathbf{x}_t$ at each (discrete) time t. We use the following state-space model:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \tag{1}$$

$$y_{n,t} = g_n(\mathbf{x}_t, v_{n,t}), \tag{2}$$

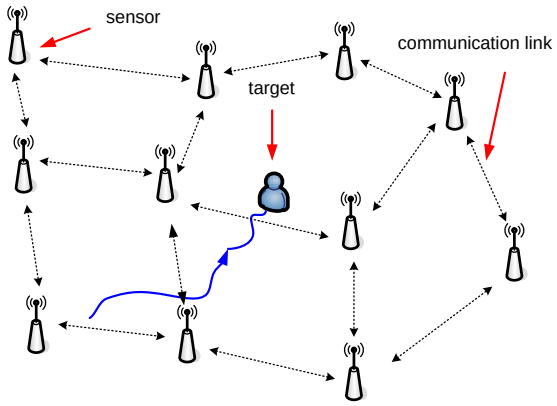


Fig. 1. Illustration of target tracking in a WSN. The goal of the WSN is to track the position and velocity of the target.

where $\mathbf{u}_t$ is process noise, $y_{n,t}$ is local observation of sensor n at time t, and $v_{n,t}$ is its observation noise. We denote the aggregation of all observations at time t by $\mathbf{y}_t$. The process noise $\mathbf{u}_t$ can be non-Gaussian, but since it is usually hard to measure [2], [21], we can assume a Gaussian approximation with sufficiently large variance, which is a common choice.

We denote by G_t the set of the nodes that have an observation available at time t, and by G_n the set of all neighbors of node n (irrespective of whether or not they have an observation). The observation noise $v_{n,t}$ is distributed according to $p_v(\cdot)$, which is not necessarily Gaussian, and typically depends on the particular measurement technique (e.g., acoustic [22], RSS [23], RF tomography [24]) and the environment. Observations at any time t are assumed to be conditionally independent.

We assume that the network is connected (i.e., there is at least a single path between any two nodes), and undirected (if node n can receive a packet from node u, then node u can receive a packet from node n). For simplicity, we assume ideal probability of detection for both sensing and communication range, but more complex models can easily be incorporated [25]. In other words, a sensor can detect the target if the distance between them is less than a certain value r, and two sensors can communicate with each other if the distance between them is less than R. Taking into account that radio of a node is usually more powerful than its sensing devices [9], [26], we assume $R \geq r$.

B. Centralized Particle Filtering

We apply the Bayesian approach for this tracking problem and recursively determine the posterior distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ given the prior $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, dynamic model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ defined by (1), and the likelihood function $p(\mathbf{y}_t|\mathbf{x}_t)$ defined by (2). We assume that $p(\mathbf{x}_0|\mathbf{y}_0) = p(\mathbf{x}_0)$ is initially available. The posterior can be found using the standard prediction and correction equations [3]:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \mathrm{d}\mathbf{x}_{t-1} \tag{3}$$

Algorithm 1 Centralized PF (CPF) (at time t)

for all particles $m = 1 : N_p$ **do**
  Draw particle: $\mathbf{x}_t^{(m)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(m)})$
  Compute weight: $w_t^{(m)} = w_{t-1}^{(m)} \cdot p(\mathbf{y}_t|\mathbf{x}_t^{(m)})$
end for
Normalize: $w_t^{(m)} = w_t^{(m)} / \sum_{m'} w_t^{(m')}$ (for $m = 1 : N_p$)
Compute estimates: $\hat{\mathbf{x}}_t = \sum_m w_t^{(m)} \mathbf{x}_t^{(m)}$
Resample with replacement from $\{w_t^{(m)}, \mathbf{x}_t^{(m)}\}_{m=1}^{N_p}$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}). \tag{4}$$

Due to the conditional independence among observations at time t, the global likelihood function $p(\mathbf{y}_t|\mathbf{x}_t)$ can be written as the product of the local likelihoods:

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto \prod_{n \in G_t} p(y_{n,t}|\mathbf{x}_t), \tag{5}$$

where local likelihood $p(y_{n,t}|\mathbf{x}_t)$ is a function of state $\mathbf{x}_t$ for a given observation $y_{n,t}$. For notational convenience we will still write $p(y_{n,t}|\mathbf{x}_t)$ for $n \notin G_t$, with the tacit assumption that this function is equal to 1.

Since the observation noise is generally not Gaussian, and the observation is not a linear function of the state, a traditional Kalman filtering [2] approach cannot be used. Instead, we apply the particle filter [3], in which the posterior distribution is represented by a set of samples (particles) with associated weights. A well-known solution is the sample-importance-resampling (SIR) method, in which N_p particles are drawn from $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, then weighted by the likelihood function, $p(\mathbf{y}_t|\mathbf{x}_t)$, and finally resampled in order to avoid degeneracy problems (i.e., the situation in which all but one particle have negligible weights). More advanced versions of PF also exist [27]–[29], but we focus on SIR since the distributed implementation of most PF-based methods is similar. We will refer to PF with SIR as centralized PF (CPF). The CPF method is summarized in Alg. 1. The CPF must be initialized with a set of particles $\{w_0^{(m)}, \mathbf{x}_0^{(m)}\}$ drawn from $p(\mathbf{x}_0)$.

This algorithm is run on one of the nodes in the WSN, which serves as fusion center. The main drawbacks of the CPF are [14], [30]: i) large energy consumption of the nodes which are in proximity of the fusion center, ii) high communication cost in large-scale networks; iii) the posterior distribution cannot be accessed from any node in the network; and iv) the fusion center has to know the locations, observations, and observation models of all the nodes. In the following section we will focus on distributed implementations of PF method, which alleviate these problems.

III. Distributed Target Tracking

Our goal is to track the target in a distributed way, such that all the nodes have a common view of the state of the target. We will first describe a flooding approach (the NCPF), followed by a general description of the DPF, which relies on a belief consensus algorithm that is run as an inner loop. We then describe 5 belief consensus methods, and finally quantify the communication cost of DPF and NCPF.

Algorithm 2 Distributed PF (DPF) (at node n, at time t)

for all particles $m = 1 : N_p$ **do**
  Draw particle: $\mathbf{x}_t^{(m)} \sim p(\mathbf{x}_{n,t}|\mathbf{x}_{t-1}^{(m)})$
  Compute weight: $w_{n,t}^{(m)} = w_{t-1}^{(m)} \cdot \mathrm{BC}\left(p(y_{1,t}|\mathbf{x}_t^{(m)}), ..., p(y_{N_s,t}|\mathbf{x}_t^{(m)})\right)$
end for
Normalize: $w_{n,t}^{(m)} = w_{n,t}^{(m)} / \sum_{m'} w_{n,t}^{(m')}$ (for $m = 1 : N_p$)
$\hat{w}_t^{(m)} = \mathrm{MC}\left(w_{1,t}^{(m)}, ..., w_{N_s,t}^{(m)}\right)$ (for $m = 1 : N_p$)
Normalize: $\hat{w}_t^{(m)} = \hat{w}_t^{(m)} / \sum_{m'} \hat{w}_t^{(m')}$ (for $m = 1 : N_p$)
Compute estimates: $\hat{\mathbf{x}}_t = \sum_m \hat{w}_t^{(m)} \mathbf{x}_t^{(m)}$
Resample with replacement from $\{\hat{w}_t^{(m)}, \mathbf{x}_t^{(m)}\}_{m=1}^{N_p}$

A. Non-Centralized Particle Filter

In NCPF [10], every sensor broadcasts its observation, observation model and an identifier, along with observations, observation models and identifiers from neighbors. This flooding procedure is repeated until every node has access to all information, when a common posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ can be determined. This approach is not scalable, but can be competitive in some scenarios (see also Section III-D).

B. Distributed Particle Filtering

For a distributed implementation of the PF, we want to avoid exchanging observations, while at the same time maintaining a common set of samples and weights at every time step. If we can guarantee that the samples and weights at time $t-1$ are common, then common samples[1] at time t can be achieved by providing all nodes with the same seed for random number generation (i.e., by ensuring that their pseudo-random generators are in the same state at all times). Ensuring common weights for all nodes can be achieved by means of a belief consensus (BC) algorithm. BC formally aims to compute, in a distributed fashion the product of a number of real-valued functions over the same variable

$$\mathrm{BC}(f_1(x), f_2(x), \ldots, f_{N_s}(x)) = \prod_{n=1}^{N_s} f_n(x). \tag{6}$$

However, most BC algorithms are not capable to achieve exact consensus in a finite number of iterations (except BP consensus in tree-like graphs; see Section III-C). As we require *exact* consensus on the weights, we additionally apply max-consensus[2] [16], [31],

$$\mathrm{MC}(f_1(x), f_2(x), \ldots, f_{N_s}(x)) = \max_n f_n(x), \tag{7}$$

which computes the exact maximum over all arguments in a finite number of iterations (equal to the diameter of the graph). In particular, max-consensus can be implemented as follows:

[1] Although different sample realizations is advantageous in many applications, this is not the case in DPF since one set of particles is sufficient for one target.

[2] Min-consensus or average over min- and max-consensus can be also applied [17].

denoting the function at iteration i at node n by $f_n^{(i)}(x)$ (with $f_n^{(0)}(x) = f_n(x)$) every node executes the following rule:

$$f_n^{(i)}(x) = \max\left(f_n^{(i-1)}(x),\ \max_{u\in G_n} f_u^{(i-1)}(x)\right). \quad (8)$$

Note that max-consensus is applied before computing the estimates in order to avoid disagreement of the estimates over network. This approach, already used in [16], [17], can be considered as a consensus-based approach to synchronize the particles for the next time instant. In addition, we assume that time-slot synchronization is performed in distributed way using any appropriate standard technique, see [32] and references therein.

The final algorithm is shown in Alg. 2. Observe that DPFs operate on two distinct time scales: a slow time scale (related to time slots) in which the target moves, and the observations are taken, and a fast time scale (related to iterations) in which BC/MC is executed for a specific time slot. Note also that in comparison with CPF, DPF has the following advantages: i) the energy consumption is balanced across the network; ii) reduced communication cost in large-scale networks; iii) every node has access to the posterior distribution; and iv) no knowledge required of the locations, observations, or observation models of any other node.

C. Belief Consensus Algorithms

Motivated by their scalability and robustness to failures [33]–[36], we consider five variants of BC: standard BC (SBC) [33], randomized gossip (RG) [35], [37], Metropolis BC (MBC) [38], broadcast gossip (BG) [34], [35], and belief propagation (BP) [36], [39] consensus. While BP consensus has not been applied within DPF, the other four BCs have already been applied [14], [16], [17], [19].

We will now describe these five distinct BC algorithms corresponding to line 3 in Alg. 2. Although we use BC to perform the consensus on particle weights, the algorithms will be presented in general form (with continuous variables) in order to relate them with the state-of-the-art algorithms.

1) Standard BC: Standard BC (SBC) [33] is defined in following iterative form:

$$M_n^{(i)}(\mathbf{x}_t) = M_n^{(i-1)}(\mathbf{x}_t) \prod_{u\in G_n} \left(\frac{M_u^{(i-1)}(\mathbf{x}_t)}{M_n^{(i-1)}(\mathbf{x}_t)}\right)^{\xi}, \quad (9)$$

where $M_n^{(i)}(\mathbf{x}_t)$ represents the approximation at iteration i of the global likelihood of the variable $\mathbf{x}_t$, and ξ is update rate, which depends on maximum node degree in the network ($\eta_{\max} = \max_n |G_n|$).[3] The update rate $\xi \approx 1/\eta_{\max}$ provides sufficiently fast convergence for the constant weight model [31], [38]. Optimized constant weights [40] can be found using distributed convex optimization [41], [42], but the optimization is generally complex since the Laplacian matrix has to be estimated at each node. We initialize (9) by

$$M_n^{(1)}(\mathbf{x}_t) = p(y_{n,t}|\mathbf{x}_t). \quad (10)$$

[3]Note that the logarithm of (9) corresponds to standard average consensus algorithm [33, eq. (4)]. All computations are done in the log-domain to avoid numerical problems.

Convergence is guaranteed for all connected graphs in a sense that [33], [38]

$$\lim_{i\to\infty} M_n^{(i)}(\mathbf{x}_t) = \left(\prod_{n'\in G_t} p(y_{n',t}|\mathbf{x}_t)\right)^{1/N_s}, \quad (11)$$

from which the desired quantity, $\prod_{n\in G_t} p(y_{n,t}|\mathbf{x}_t)$, can easily be found, for any value of $\mathbf{x}_t \in \{\mathbf{x}_t^{(1)}, \ldots, \mathbf{x}_t^{(N_p)}\}$. However, in practical circumstances, we run SBC a finite number of iterations ($i = 1, 2, \ldots, N_i^{\mathrm{SBC}}$), so the result will be an approximation of the real likelihood.

If the maximum node degree ($\eta_{\max}$) and number of nodes (N_s) are not known a priori, we need to estimate them in distributed way.[4] The estimation of maximum node degree can be done using max-consensus, while N_s can be determined [43] by setting an initial state of one node to 1, and all others to 0. By using average consensus [31], all nodes can obtain the result $1/N_s$, i.e., the inverse of the number of nodes in the network.

SBC was used for consensus on weights in [17]. Moreover, there are a number of specific instances of SBC (e.g., [11]–[13]) that represent the beliefs in parametric form (e.g., Gaussian, or Gaussian mixture), and make consensus on their parameters.

2) BC based on Randomized Gossip: Gossip-based algorithms [35] can be also used to achieve consensus in a scalable and robust way. We consider randomized gossip (RG) [37]. In RG, it is assumed that all the nodes have internal clocks that tick independently according to a rate of e.g., a Poisson process [35]. When the clock of the n-th node ticks, node n and one of its neighbors (randomly chosen) exchange their current estimates, and make the update. In case of BC based on RG, we need to achieve convergence to the geometrical average, so at the i-th clock tick of node n, the nodes n and u make the following operation:

$$M_u^{(i)}(\mathbf{x}_t) = M_n^{(i)}(\mathbf{x}_t) = \left(M_u^{(i-1)}(\mathbf{x}_t) M_n^{(i-1)}(\mathbf{x}_t)\right)^{1/2} \quad (12)$$

where $u \in G_n$, and all other nodes r in the network ($r \notin \{n, u\}$) do not make any update (i.e., $M_r^{(i)}(\mathbf{x}_t) = M_r^{(i-1)}(\mathbf{x}_t)$). Initialization is done using (10). In order to have the same communication cost as SBC, RG should run approximately $N_i^{\mathrm{RG}} = \lceil N_i^{\mathrm{SBC}} N_s/2 \rceil$ iterations. Finally, we again need to estimate N_s using the same approach as for SBC.

RG has been used in [16] for consensus on weights, and a specific instance (with Gaussian approximation) in [15].

3) BC based on Broadcast Gossip: The main problem of RG is that once a node broadcasts data, only one of its neighbors performs an update. It is natural to expect that if all the neighbors perform an update, the convergence will be faster. To address this problem, broadcast gossip (BG) has been proposed [34], in which a randomly chosen node broadcasts its current estimate, and all of its neighbors (within

[4]While any upper bound on $\eta_{\max}$ guarantees convergence, estimation of $\eta_{\max}$ is preferable to increase convergence speed of SBC.

communication radius R) perform an update. It has been shown [34] that, due to its asymmetric nature, BG converges only in expectation to the real average value.[5]

In our case, we need to achieve convergence to the geometrical average (11), so at the i-th clock tick of node n all the nodes make the following operation:

$$M_u^{(i)}(\mathbf{x}_t) = \begin{cases} M_u^{(i-1)}(\mathbf{x}_t)^{\gamma} M_n^{(i-1)}(\mathbf{x}_t)^{1-\gamma}, & u \in G_n \\ M_u^{(i-1)}(\mathbf{x}_t), & \text{otherwise.} \end{cases} \tag{13}$$

where $0 < \gamma < 1$ is the mixing parameter. Again, initialization is done using (10). To synchronize communication cost with previous BC methods, we run BG $N_i^{\mathrm{BG}} = N_i^{\mathrm{SBC}} N_s$ iterations. It is again necessary to apply average consensus to estimate N_s.

A variant of this method, with particle compression based on support vector machine, has been already applied in [19].

Comment: Regarding the choice of γ, it has been shown in [34] that its optimal value depends on the algebraic connectivity of the graph, which is the second smallest eigenvalue of the Laplacian matrix [31], [34]. However, this parameter is not available in the distributed scenario, so an empirical study has been used [34] to find the optimal value of γ. Therefore, we will model γ as a function of average node degree $\bar{\eta}$ in the network, since $\bar{\eta}$ can be easily estimated using average consensus. We found that an optimal γ can be modeled as

$$\gamma(\bar{\eta}) = 1 - ae^{-b\bar{\eta}} \in (0, 1), \tag{14}$$

with parameters a and b, which can be estimated by calibration.

4) Metropolis BC: An important problem of the SBC method is that it uses a constant weight model, i.e., $\xi_{nu} = \xi$ for each link (n, u) in the graph, which will not provide good performance in asymmetrical graphs. Instead of this model, we can use so-called Metropolis weights, which should provide faster convergence [38]. For our problem, this leads to Metropolis BC (MBC), with the following update rule:

$$M_n^{(i)}(\mathbf{x}_t) = M_n^{(i-1)}(\mathbf{x}_t)^{\xi_{nn}} \prod_{u \in G_n} M_u^{(i-1)}(\mathbf{x}_t)^{\xi_{nu}}, \tag{15}$$

where the weight on an link $\{n, u\}$ is given by:

$$\xi_{nu} = \xi_{un} = \begin{cases} 1/\max(\eta_n, \eta_u), & \text{for } u \neq n \\ 1 - \sum_{u' \in G_n} \xi_{u'n}, & \text{for } u = n. \end{cases} \tag{16}$$

The initialization is the same as for other BC methods, and the number of iterations is the same as for SBC ($N_i^{\mathrm{MBC}} = N_i^{\mathrm{SBC}}$). This approach also guarantees convergence [38] to (11).[6] As we can see, this method is more suitable than SBC for distributed implementation, since a node needs to know only the local degrees of its neighbors. However, N_s still has to be estimated.

[5]Note that the underlying communication graph is still undirected, since all the nodes have to be capable to broadcast their estimates (even if they don't do so at each iteration).

[6]Provided that the graph is not bipartite. Otherwise, $\max(\eta_n, \eta_u)$ should be replaced with $\max(\eta_n, \eta_u) + 1$.

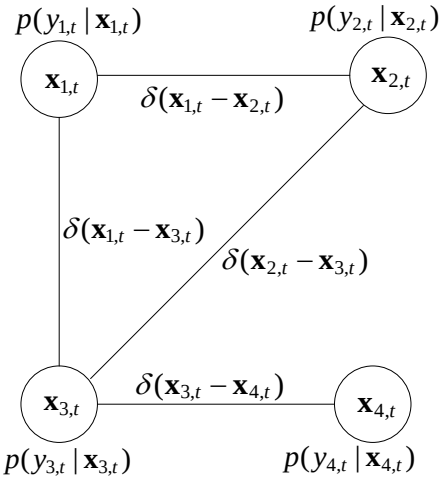


Fig. 2. Example of a graphical model for BP consensus, for a network with 4 sensors.

A specific instance of MBC, in which the beliefs belong to the exponential family, is used in [14].

5) BC based on Belief Propagation: Belief propagation (BP) [36], [39] is a way of organizing the global computation of marginal beliefs in terms of smaller local computations within the graph. To adapt it for BC, we define the following function:

$$f(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, \ldots, \mathbf{x}_{N_s,t}) = \prod_n p(y_{n,t}|\mathbf{x}_{n,t}) \prod_{u \in G_n} \delta(\mathbf{x}_{n,t} - \mathbf{x}_{u,t}). \tag{17}$$

Running BP on the corresponding graphical model (see example in Figure 2) yields the marginals $M_n(\mathbf{x}_{n,t})$ of the function $f(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, \ldots, \mathbf{x}_{N_s,t})$. It is easily verified that for every n

$$M_n(\mathbf{x}_{n,t}) = \sum_{\mathbf{x}_{u \neq n,t}} f(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, \ldots, \mathbf{x}_{N_s,t}) \tag{18}$$

$$= \prod_{n'} p(y_{n',t}|\mathbf{x}_{n,t}). \tag{19}$$

The BP message passing equations are now as follows: the belief at iteration i is given by [25, eq. (8)]

$$M_n^{(i)}(\mathbf{x}_{n,t}) \propto p(y_{n,t}|\mathbf{x}_{n,t}) \prod_{u \in G_n} m_{un}^{(i)}(\mathbf{x}_{n,t}), \tag{20}$$

while the message from node $u \in G_n$ to node n is given by [25, eq. (9)]

$$m_{un}^{(i)}(\mathbf{x}_{n,t}) \propto \int_{\mathbf{x}_{u,t}} \delta(\mathbf{x}_{n,t} - \mathbf{x}_{u,t}) \frac{M_u^{(i-1)}(\mathbf{x}_{u,t})}{m_{nu}^{(i-1)}(\mathbf{x}_{u,t})} d\mathbf{x}_{u,t} \tag{21}$$

$$= \frac{M_u^{(i-1)}(\mathbf{x}_{n,t})}{m_{nu}^{(i-1)}(\mathbf{x}_{n,t})}. \tag{22}$$

We note that since all variables are the same, we can write

$\mathbf{x}_{n,t} = \mathbf{x}_{u,t} = \mathbf{x}_t$. Some manipulation yields (see A)

$$M_n^{(i)}(\mathbf{x}_t) \propto M_n^{(i-2)}(\mathbf{x}_t) \prod_{u \in G_n} \left(\frac{M_u^{(i-1)}(\mathbf{x}_t)}{M_n^{(i-2)}(\mathbf{x}_t)} \right), \tag{23}$$

which represents the consensus algorithm based on BP. Since BP consensus uses the same protocol as SBC and MBC, we should run it $N_i^{\mathrm{BP}} = N_i^{\mathrm{SBC}}$ iterations. This method is initialized by $M_n^{(1)}(\mathbf{x}_t) = p(y_{n,t}|\mathbf{x}_t)$. We also need to set $M_n^{(2)}(\mathbf{x}_t)$ in order to run the algorithm defined by (23). Using (20) and (21), and assuming that $m_{nu}^{(1)}(\mathbf{x}_t) = 1$, we find

$$M_n^{(2)}(\mathbf{x}_t) = p(y_{n,t}|\mathbf{x}_t) \prod_{u \in G_n} p(y_{u,t}|\mathbf{x}_t). \tag{24}$$

BP consensus (as a specific instance of BP) guarantees convergence to $C \prod_n p(y_{n,t}|\mathbf{x}_t)$ for cycle-free network graphs [39], [44], [45], where C is an irrelevant normalization constant.[7] When the network has cycles, the beliefs are only approximations of the true marginals given by (19) (more details in A). Comparing BP consensus with previous consensus methods, we can see that BP-consensus agrees on product of all local evidences (not the N_s-th root of the product), and does not rely on knowledge of any other parameters. Therefore, it is more robust to the changes in the network.

D. Communication Cost Analysis

In this section, we analyze the communication cost of the DPF methods, and compare with the cost of NCPF. We denote by N_{pack} the number of packets that any node n broadcasts at any time t. We assume that one packet can contain P scalar values (mapping from scalars to bits is not considered). In most hardware platforms, $P \gg 1$, and the energy required to transmit one packet does not significantly depend on the amount of data within it. In this analysis, we neglect the cost of determining $\eta_{\max}$, $\bar{\eta}$, and N_s and the cost of time-slot synchronization. Consequently, all DPF methods will have the same communication cost.

1) Cost of DPF: At every iteration (except the first), nodes transmit N_p weights. In addition, nodes must perform MC, which also requires transmission of the weights in each iteration. The number of iterations of the MC is equal to the diameter of the graph D_g. Thus, the average cost of DPF per node and per time slot is

$$N_{\mathrm{pack}}^{\mathrm{DPF}} \approx \left\lceil \frac{N_p}{P} \right\rceil (D_g + N_{\mathrm{it}} - 1). \tag{25}$$

We see that the DPF methods are fully scalable, since increasing the number of the nodes in a fixed deployment area will not significantly affect the cost. Although beyond the scope of this paper, we mention that if one prefers to use parametric approximations [11]–[14] instead of consensus on weights, only parameters of the beliefs should be transmitted in each iteration.

[7] It is irrelevant since the weights in Alg. 2 will be normalized later anyway.

[8] We use N_i^{SBC} to count iterations ($N_{\mathrm{it}} = N_i^{\mathrm{SBC}}$). All other DPF methods run the number of iterations which ensures the same communication cost as explained in Section III-C.

2) Cost of NCPF: In contrast to DPF, NCPF does not require transmission of the weights, but only the local data as described in Section III-A. We denote the number of these scalar values as N_{data}. The amount of data will accumulate with iterations since the node has to transmit its own data and all received data. Since the number of iterations is equal to D_g, the cost can be approximated by:

$$N_{\mathrm{pack}}^{\mathrm{NCPF}} \approx \sum_{k=0}^{D_g-1} \left\lceil \frac{\bar{\eta}^k N_{\mathrm{data}}}{P} \right\rceil, \tag{26}$$

where we approximate the degree of the each node with average network degree ($\bar{\eta}$), and $\sum_{i=0}^{k} \bar{\eta}^i$ by $\bar{\eta}^k$.

3) Comparison between DPF and NCPF: Making the reasonable assumption that $N_{\mathrm{it}} = D_g + 1$, we can quantify when DPF is preferred over NCPF, i.e., when $N_{\mathrm{pack}}^{\mathrm{DPF}} < N_{\mathrm{pack}}^{\mathrm{NCPF}}$:

$$\left\lceil \frac{N_p}{P} \right\rceil < \frac{1}{2D_g} \sum_{k=0}^{D_g-1} \left\lceil \frac{\bar{\eta}^k N_{\mathrm{data}}}{P} \right\rceil. \tag{27}$$

This condition is important in order to avoid over-using of consensus-based methods. For example, if the network is fully-connected ($D_g = 1$), or if the packet size is sufficiently large to afford transmission of all accumulated data (i.e., $P > \bar{\eta}^{D_g-1} N_{\mathrm{data}}$), NCPF should be applied. On the other hand, if the communication radius is very small (i.e., if D_g is very large), DPF methods should be applied.

A similar comparison can be done with CPF, but its cost depends on many factors [1], including the routing protocol, and the position of the fusion center. In general, the cost of CPF is much smaller than NCPF since there is only one (instead of N_s) fusion center. However, even with reduced cost, CPF is not a good alternative due to the uneven energy consumption over nodes, and since the posterior cannot be accessed from any node (see also Section II-B).

Regarding computational complexity, DPF methods require $\mathcal{O}(|G_n|\, N_{it} N_p)$ operations per node at time t, while NCPF requires only $\mathcal{O}(|G_t|\, N_p)$ operations per node. However, the energy consumption in typical hardware platforms is mainly caused by communication, while internal computation is very cheap [30]. Therefore, DPF methods are still preferred over NCPF.

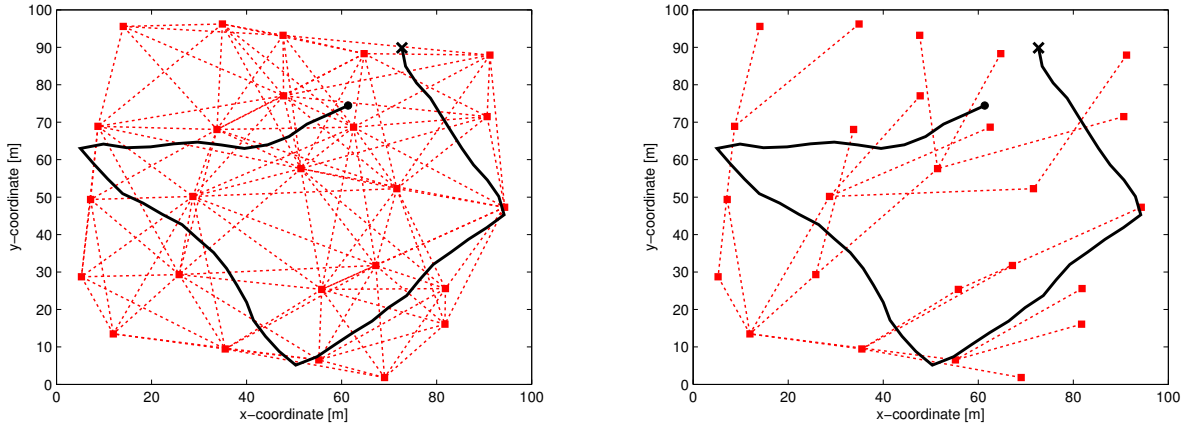
IV. Simulation Results

A. Simulation Setup and Performance Metrics

We assume that there are $N_s = 25$ sensors semi-randomly[9] deployed in a 100m x 100m area. The positions of these sensors are perfectly known. There is also one target in the area, with state $\mathbf{x}_t = [x_{1,t}\ x_{2,t}\ \dot{x}_{1,t}\ \dot{x}_{2,t}]^T$, where $x_{1,t}$ and $x_{2,t}$ represent 2D position of the target, and $\dot{x}_{1,t}$ and $\dot{x}_{2,t}$ the 2D velocity of the target. The target moves with constant amplitude of the speed of 5m/s, according to Gaussian random walk, given by

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{I}_2 & T_s \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix} \mathbf{x}_t + \begin{bmatrix} \frac{T_s^2}{2} \mathbf{I}_2 \\ T_s \mathbf{I}_2 \end{bmatrix} \mathbf{u}_t, \tag{28}$$

[9] Semi-random network is created by adding jitter to a regular 2D grid.



(a) (b)

Fig. 3. Illustrations of the target track in: (a) semi-random, and (b) tree network, for $R = 45$ m. The start point of the track is marked with a small black disc, and the end point with an X. Sensors are marked with the squares, and communication links with dashed lines.

where T_s is the sampling interval (here set to 1 second), and $\mathbf{I}_2$ and $\mathbf{0}_2$ represent the identity and zero 2×2 matrices, respectively. The process noise $\mathbf{u}_t$ is distributed according to zero-mean Gaussian with covariance matrix $0.5\mathbf{I}_2$ $(\mathrm{m/s^2})^2$. The target is tracked for $N_t = 50$ seconds.

We set the sensing radius to $r = 25$m, and consider two values of communication radius: $R = 25$m and $R = 45$m (corresponding to $\bar{\eta} = 3.08$ and $\bar{\eta} = 9.44$, respectively). In addition, we consider a tree configuration created as a spanning tree[10] from semi-random network. An example of a target track in a loopy and tree network is shown in Figure 3. The inclusion of tree topologies is motivated by certain target tracking applications, such as tunnels.

We assume that the observations are distances to the target, i.e., for $n \in G_t$,

$$y_{n,t} = g_n(\mathbf{x}_t) = \left\| \mathbf{l}_n - [x_{1,t}\ x_{2,t}]^T \right\| + v_{n,t}. \qquad (29)$$

The observation noise $v_{n,t}$ is distributed according to Gaussian mixture with two components, a typical model in presence of non-line-of-sight signals. The parameters of this noise are set to following values: means $(1\mathrm{m}, 10\mathrm{m})$, variances $(1\mathrm{m}^2, 1\mathrm{m}^2)$ and mixture weights $(0.9, 0.1)$.

For mixing parameter in BG, given by (14), we set $a = 0.49$, and $b = 0.17$ found by calibration. We use $N_p = 500$ particles. The results are averaged over $N_R = 500$ Monte Carlo runs. In each run, we generate different network, track, observations and particle seed.

We will compare five DPF methods (DPF-SBC, DPF-MBC, DPF-RG, DPF-BG, and DPF-BP). Moreover, as a benchmark, we show the performance of the exact approach, which corresponds to performance of CPF and NCPF. We consider root-mean-square error (RMSE) in the position error $e_{\mathrm{rms},t}$ and e_{rms} as a performance metrics. Introducing $e_{n,t,s}$ as the target positioning error (i.e., Euclidean distance between the true and estimated position of the target) at node n, at time t in simulation run s, we have

$$e_{\mathrm{rms},t} = \sqrt{\frac{\sum_{n=1}^{N_s}\sum_{s=1}^{N_R} e_{n,t,s}^2}{N_s N_R}}, \quad e_{\mathrm{rms}} = \sqrt{\frac{\sum_{t=1}^{N_t} e_{\mathrm{rms},t}^2}{N_t}} \qquad (30)$$

[10]Tree configurations should not be created using an online algorithm since it would require a routing of data. We simply assumed that the tree configuration is established offline or is an inherent property of the network.

B. Performance Results

We first investigate the RMSE e_{rms} as a function of the number of iterations N_{it}, for different communication radii and network topologies. The results are shown in Figure 4. We can draw a number of conclusions. As expected, exact methods provide the best RMSE performance, as they compute the exact likelihood corresponding to (5). Secondly, all DPF methods (except for DPF-BP) are capable to reach asymptotically the performance of the exact approach. Among the DPF methods in the semi-random topology (Figures 4a and 4c), DPF-MBC provides the fastest convergence, for both considered communication radii, though DPF-BG and DPF-SBC achieve similar performance for $R = 45$ m. DPF-BP achieves a minimal RMSE for $N_{\mathrm{it}} = D_g + 1$, since then all local likelihoods are available at each node (see also A). A further increase of the number of iterations will only increase the amount of over-counting of the local likelihoods, thus leading to biased beliefs. In the tree topology (Figures 4b and 4d), the situation is very different: DPF-BP provides the fastest convergence and *exact* result after $D_g + 1$ iterations. However, note that D_g is typically higher for tree networks than loopy networks. Finally, we also see that: i) increasing R increases the convergence speed of all DPFs, and ii) DPF-RG performs the worst for both considered radii and configurations.

Secondly, in Figure 5, we analyze the RMSE $e_{\mathrm{rms},t}$ as a function of time for the track and the networks from Figure 3. We consider the case in which $R = 45$ m, and N_{it} is set to provide the optimal performance of DPF-BP. Here, in each Monte Carlo run, we generate different observations and particle seeds, but keep the same track and the network. In case of semi-random network (Figure 3a), we see that all DPFs (except DPF-RG) provide the performance close to the

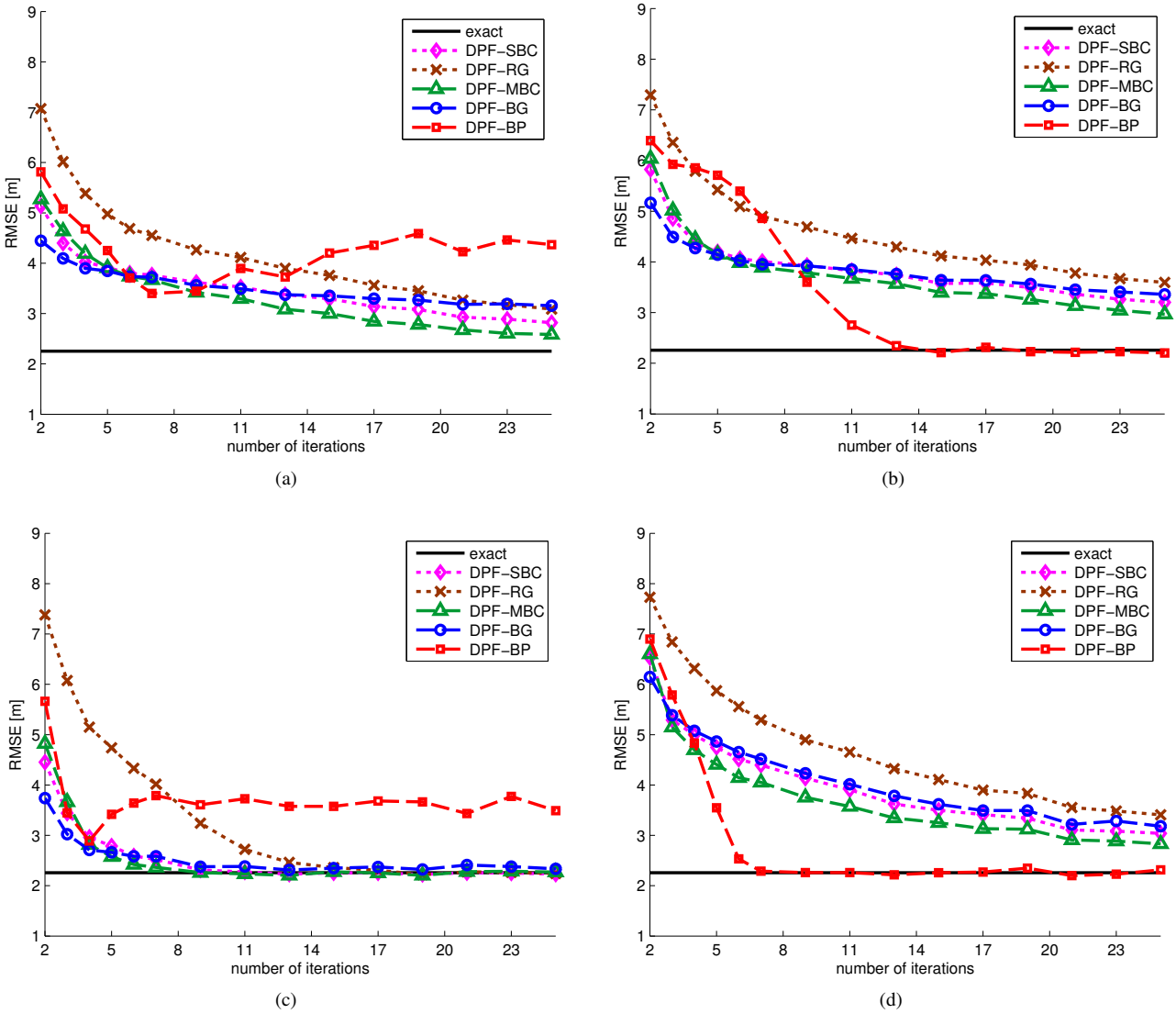


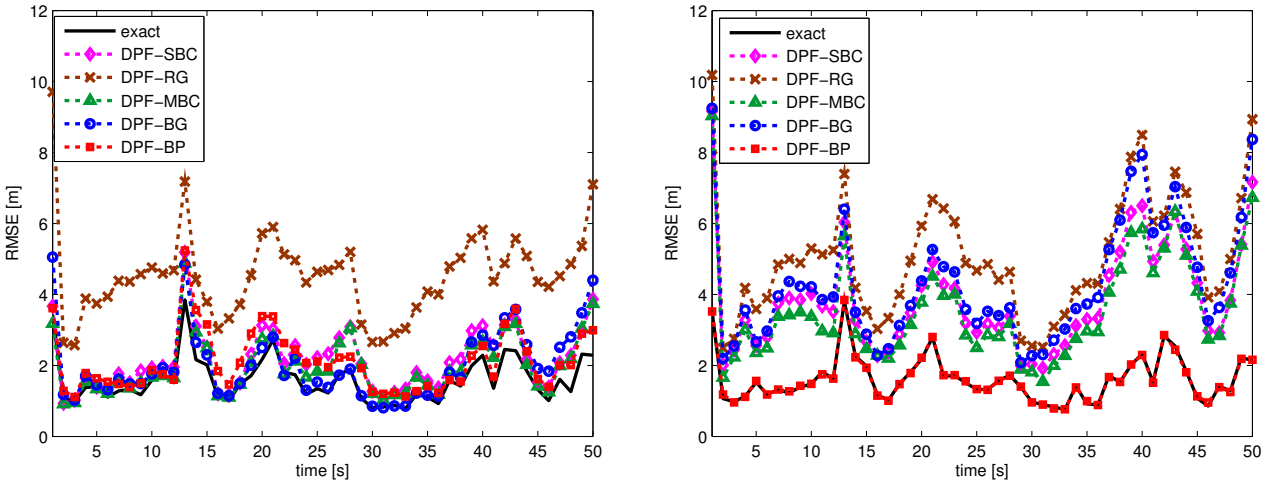
Fig. 4. RMSE of DPF methods as a function of the number of iterations for (a) semi-random ($R = 25$ m), (b) tree ($R = 25$ m), (c) semi-random ($R = 45$m), and (d) tree ($R = 45$m).

exact approach. On the other hand, in the tree network, DPF-BP performance overlaps with the exact approach. In general, DPF-BP can be used in all configurations, if we know (or are able to estimate) D_g. However, in practice, the network may have no knowledge of D_g, so DPF-BP would lead to biased results.

Finally, we evaluate the communication cost per node, by analyzing the average number of packets per node as a function of the communication radius R for $N_{\text{it}} = \lceil L/R \rceil + 1$. Here L is the diameter of the deployment area ($L = 100\sqrt{2}$ m, in our case). We consider networks with 25 and 100 nodes, and packet sizes of $P = 1$, $P = N_p$, and $P = 5N_p$, where $N_p = 500$, and $N_{\text{data}} = 9$ (i.e., it includes 2 scalars for the position, 6 scalars for the observation model, and 1 scalar for the observation). To count the number of packets, we simulated the degree of each node in all networks, and applied equations (25)–(26). As we can see from Figure 6, DPF-based methods provide nearly constant communication cost as a function of R, since (25) only depends linearly on D_g, and it does not depend on N_s. Thus, these methods are scalable. On the other hand, the communication cost of NCPF is highly sensitive to R and N_s: the cost increases as R increases (while $\lceil L/R \rceil$ is fixed), and decreases significantly when $\lceil L/R \rceil$ decrements its value (e.g., for $R = 50\sqrt{2}$). Overall, decreasing $\lceil L/R \rceil$ has the largest effect (see (26)), so the total cost has a decreasing tendency with R. In addition, since the increased N_s affects $\bar{\eta}$, the communication cost will be significantly larger. Regarding the effect of P, we can see that larger values of P will make NCPF cheaper, as more data can be aggregated in one packet. Finally, comparing with NCPF, we can see that DPF methods have a lower communication cost for $R < 70$m, except when P is very large (as in Figure 6c).

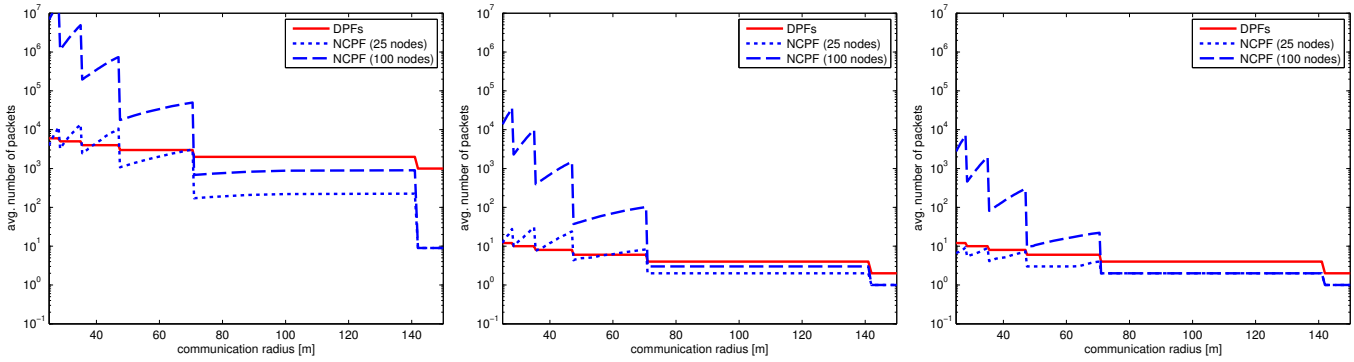
C. Discussion

According to previous results, we can see that BC-based DPF methods should be applied only in sparse networks, i.e.,



(a) (b)

Fig. 5. RMSE of DPF methods as a function of time for: (a) semi-random network from Figure 3a ($N_{\text{it}} = 4$), and (b) tree network from Figure 3b, ($N_{\text{it}} = 7$).



(a) (b) (c)

Fig. 6. Communication cost comparison as a function of the communication radius, for: (a) $P = 1$, (b) $P = N_p$, and (c) $P = 5N_p$.

in networks with relatively high diameter. Otherwise, NCPF may be used since it is cheaper to flood a few scalars instead of iteratively broadcasting weights or parameters of the likelihood function. Moreover, we found that DPF-MBC is the fastest in loopy graphs, closely followed by DPF-SBC, and DPF-BG. Hence, DPF-MBC is preferred in unconstrained areas in which the loops are expected, such as conference halls, airports, and warehouses [21]. On the other hand, DPF-BP is only suitable for confined areas, in which long chains of sensors, globally forming a tree, are deployed. Typical examples are subway, roadway and mine tunnels [46], [47]. Note that the robustness is decreased in tree graphs since a node failure can break the graph into two subgraphs. However, taking into account all other benefits of DPF methods (especially, scalability), they are still preferred over leader-agent methods in tree graphs.

V. Conclusion

We have studied DPF for target tracking and compared five consensus methods in a unified scenario, in terms of RMSE performance and communication cost. The five methods include four from literature (SBC, RG, MBC, and BG), and one novel method based on BP. According to our results, DPF-MBC should be used in loopy networks, while DPF-BP is preferred in tree networks. We also found that BC-based DPF methods have lower communication overhead than NCPF (based on data flooding) in sparse networks. Further research is required to estimate the diameter of the graph in distributed way, to reduce the bias of DPF-BP in loopy networks, and to assess the impact of medium access control on the communication delay.

Appendix

From (22), we know that

$$m_{un}^{(i)}(\mathbf{x}_t) \propto \frac{M_u^{(i-1)}(\mathbf{x}_t)}{m_{nu}^{(i-1)}(\mathbf{x}_t)}, \tag{31}$$

where we removed index n since all the nodes have the same variable ($\mathbf{x}_{n,t} = \mathbf{x}_{u,t} = \mathbf{x}_t$). The denominator of (31) is the message from node n to node u in the previous iteration, and can be expressed as

$$m_{nu}^{(i-1)}(\mathbf{x}_t) \propto \frac{M_n^{(i-2)}(\mathbf{x}_t)}{m_{un}^{(i-2)}(\mathbf{x}_t)}. \tag{32}$$

Combining previous two equations, we get the recursive expression for the messages

$$m_{un}^{(i)}(\mathbf{x}_t) \propto \frac{M_u^{(i-1)}(\mathbf{x}_t)}{M_n^{(i-2)}(\mathbf{x}_t)} m_{un}^{(i-2)}(\mathbf{x}_t) \tag{33}$$

Combining (20) and (33), we find a recursive expression for the beliefs:

$$\begin{aligned} M_n^{(i)}(\mathbf{x}_t) &\propto p(y_{n,t}|\mathbf{x}_t) \prod_{u \in G_n} \left(\frac{M_u^{(i-1)}(\mathbf{x}_t)}{M_n^{(i-2)}(\mathbf{x}_t)} m_{un}^{(i-2)}(\mathbf{x}_t) \right) \\ &= M_n^{(i-2)}(\mathbf{x}_t) \prod_{u \in G_n} \left(\frac{M_u^{(i-1)}(\mathbf{x}_t)}{M_n^{(i-2)}(\mathbf{x}_t)} \right). \end{aligned} \tag{34}$$

In this appendix, we provide a deeper analysis of convergence behavior of BP consensus in loopy graphs. It is well-known [39] that BP consensus (as a special case of standard BP) converges to the exact solution after a finite number of iterations in cycle-free graphs. Using an appropriate message schedule, this number of iterations is equal to $D_g + 1$, where D_g is the diameter of the graph (i.e., the maximum hop-distance between any two nodes). However, for general graphs, it is straightforward to show (using equation (23)) that the beliefs of BP consensus after $D_g + 1$ iterations is given by

$$M_n^{(D_g+1)}(\mathbf{x}_t) \propto \prod_{u \in G_t} p(y_{u,t}|\mathbf{x}_t)^{\alpha_{u,n,t}} \tag{35}$$

where $\alpha_{u,n,t} \geq 1$ is an exponent ($\alpha_{u,n,t} \in \mathbb{N}$) of node pair (u, n) at time *t*. In case of cycle-free graphs and some specific symmetric graphs (see later examples), $\alpha_{u,n,t} = 1$, so the estimated belief is equal to desired global likelihood (given by (5)). That means that DPF-BP, after D_g+1 iterations, provides (at each node) an estimate exactly the same as CPF/NCPF. In case of $\alpha_{u,t,n} > 1$, the observation from node *u* at time *t* is *over-counted* at node n. To understand the overcounting behavior, we determine $\alpha_{\max}$, the maximum value (maximized over n and u) of $\alpha_{u,n,t}$ after $D_g + 1$ iterations. Note that running more than $D_g + 1$ iterations is unnecessary, as it will increase the α-values. While for the general case this problem is hard, we limit ourselves to some best- and the worst-case examples. In particular, we consider 4 representative graph configurations, shown in Figure 7:

1) *Fully-connected graph (clique):* For the example in Figure 7a, $D_g = 1$, so the belief at second iterations is given by (24). Since the graph is fully-connected, we know that set G_n includes all nodes in the graph except node *n* (which is locally available). Therefore, $\alpha_{\max} = 1$, so BP consensus is correct.
2) *Single-cycle graph with even number of nodes:* For the example in Figure 7b, $D_g = 2$, so we need to run 3 iterations of BP. In the second iteration, node 1 will obtain likelihood from nodes 2 and 3, but in the third iteration it will obtain likelihood from node 4 twice (through nodes 2 and 3). Therefore, $\alpha_{\max} = 2$.
3) *Single-cycle graph with odd number of nodes:* For the example in Figure 7c, again $D_g = 2$, so we need to run 3 iterations of BP. In the second iteration, node 1 will obtain likelihood from nodes 2 and 3, and in the third iteration it will obtain likelihood from nodes 4 and 5. Therefore, $\alpha_{\max} = 1$, so BP consensus is correct.
4) *Graph with short loops:* For the example in Figure 7d (without node 8), $D_g = 3$, so we need 4 iterations of BP. After 4 iterations, nodes 1 and 6 will have triple-counted their own local likelihoods (since it has its own information, as well as messages received due to the clockwise and counter-clockwise circulation through short loop[11] 1-6-7). Therefore, $\alpha_{\max} = 3$. If we consider the case with node 8 and two dashed links (in Figure 7d), $\alpha_{\max} = 5$. This reasoning can be generalized to a case with $N_{\text{short}-\text{loops}}$ short loops (which all contain the link 1-6), $\alpha_{\max} = 1 + 2N_{\text{short}-\text{loops}}$.

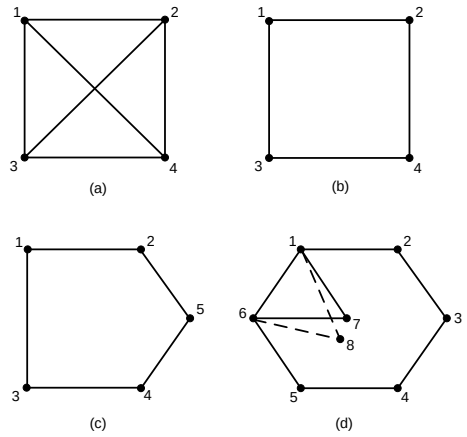


Fig. 7. Example graphs: (a) fully-connected graph ($D_g = 1$), (b) single-cycle graph with even number of nodes ($D_g = 2$), (c) single-cycle graph with odd number of nodes ($D_g = 2$), and (d) single-cycle graph with added short loop(s) ($D_g = 3$).

TABLE I
ESTIMATES OF BP CONSENSUS (FIRST 3 ITERATIONS) FOR THE GRAPH IN FIGURE 7C. THE LOCAL LIKELIHOOD $p(y_{u,t}|x_t)$ IS MARKED BY $\phi_{u,t}$.

	iter. 1	iter. 2	iter. 3
node 1	$\phi_{1,t}$	$\phi_{1,t}\phi_{2,t}\phi_{3,t}$	$\phi_{1,t}\phi_{2,t}\phi_{3,t}\phi_{4,t}\phi_{5,t}$
node 2	$\phi_{2,t}$	$\phi_{1,t}\phi_{2,t}\phi_{5,t}$	$\phi_{1,t}\phi_{2,t}\phi_{3,t}\phi_{4,t}\phi_{5,t}$
node 3	$\phi_{3,t}$	$\phi_{1,t}\phi_{3,t}\phi_{4,t}$	$\phi_{1,t}\phi_{2,t}\phi_{3,t}\phi_{4,t}\phi_{5,t}$
node 4	$\phi_{4,t}$	$\phi_{3,t}\phi_{4,t}\phi_{5,t}$	$\phi_{1,t}\phi_{2,t}\phi_{3,t}\phi_{4,t}\phi_{5,t}$
node 5	$\phi_{5,t}$	$\phi_{2,t}\phi_{4,t}\phi_{5,t}$	$\phi_{1,t}\phi_{2,t}\phi_{3,t}\phi_{4,t}\phi_{5,t}$

All previous claims can be easily proved using (23) and (24). As example, we show in Table I the estimates for the cycle-graph with odd number of nodes.

Taking into account that the fourth case is the worst-case scenario, we can conclude that in the worst-case $\alpha_{\max} = 1 + 2N_{\text{short}-\text{loops}}$. This is not a promising conclusion, since $\alpha_{\max}$ can be unbounded, for fixed D_g, as the number of nodes grows. Therefore, the configurations in which there are many short loops over the same link, are not preferable for BP consensus.

[11]A short loop is defined as a loop that consists of 3 nodes.

REFERENCES

[1] O. Hlinka, F. Hlawatsch, and P. M. Djuric, "Distributed particle filtering in agent networks: A survey, classification, and comparison," *IEEE Signal Processing Magazine*, vol. 30, pp. 61–81, Jan. 2013.

[2] G. Welch and G. Bishop, "An introduction to the Kalman filter," tech. rep., University of North Carolina at Chapel Hill, July 2006.

[3] M. S. Arulampalam, S. Maskell, N. G. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, Feb. 2002.

[4] M. Coates, "Distributed particle filters for sensor networks," in *Proc. of 3rd Workshop on Information Processing in Sensor Networks (IPSN)*, pp. 99–107, April 2004.

[5] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," in *Proc. of Fourth Int. Symp. Information Processing in Sensor Networks (IPSN)*, pp. 181–188, April 2005.

[6] S. H. Lee and M. West, "Markov chain distributed particle filters (MCDPF)," in *Proc. of 48th IEEE Conf. held jointly with the 2009 28th Chinese Control Conf Decision and Control (CDC/CCC)*, pp. 5496–5501, Dec. 2009.

[7] O. Hlinka, P. M. Djuric, and F. Hlawatsch, "Time-space-sequential distributed particle filtering with low-rate communications," in *Proc. of Asilomar Conf.*, pp. 196–200, Nov. 2009.

[8] M. Coates and G. Ing, "Sensor network particle filters: motes as particles," in *in Proc. of IEEE Workshop on Statistical Signal Processing (SSP)*, July 2005.

[9] B. Jiang and B. Ravindran, "Completely distributed particle filters for target tracking in sensor networks," in *Proc. of IEEE Int. Parallel & Distributed Processing Symp.*, pp. 334–344, May 2011.

[10] P. M. Djuric, J. Beaudeau, and M. Bugallo, "Non-centralized target tracking with mobile agents," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5928–5931, May 2011.

[11] D. Gu, "Distributed particle filter for target tracking," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 3856–3861, April 2007.

[12] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in *Proc. of Int. Conf. Information and Automation*, pp. 302–307, June 2008.

[13] O. Hlinka, O. Sluciak, F. Hlawatsch, P. M. Djuric, and M. Rupp, "Distributed gaussian particle filtering using likelihood consensus," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3756–3759, May 2011.

[14] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, and M. Rupp, "Likelihood consensus and its application to distributed particle filtering," *IEEE Transactions on Signal Processing*, vol. 60, pp. 4334–4349, Aug. 2012.

[15] B. N. Oreshkin and M. J. Coates, "Asynchronous distributed particle filter via decentralized evaluation of gaussian products," in *Proc. of 13th Conf. on Information Fusion (FUSION)*, pp. 1–8, July 2010.

[16] D. Ustebay, M. Coates, and M. Rabbat, "Distributed auxiliary particle filters using selective gossip," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3296–3299, May 2011.

[17] S. Farahmand, S. I. Roumeliotis, and G. B. Giannakis, "Set-membership constrained particle filter: Distributed adaptation for sensor networks," *IEEE Trans. on Signal Processing*, vol. 59, pp. 4122–4138, Sept. 2011.

[18] C. Lindberg, L. S. Muppirisetty, K-M. Dahlén, V. Savic, and H. Wymeersch. "MAC Delay in Belief Consensus for Distributed Tracking," in *Proc. of 10th Workshop on Positioning, Navigation and Communication (WPNC)*, March 2013.

[19] H. Q. Liu, H. C. So, F. K. W. Chan, and K. W. K. Lui, "Distributed particle filter for target tracking in sensor networks," *Progress In Electromagnetics Research*, vol. 11, pp. 171–182, 2009.

[20] V. Savic, H. Wymeersch, and S. Zazo, "Distributed target tracking based on belief propagation consensus," in *Proc. of the 20th European Signal Processing Conference (EUSIPCO)*, pp. 544–548, Aug. 2012.

[21] V. Savic, A. Athalye, M. Bolic, and P. M. Djuric, "Particle filtering for indoor RFID tag tracking," in *Proc. of IEEE Statistical Signal Processing Workshop (SSP)*, pp. 193 –196, June 2011.

[22] N. Ahmed, M. Rutten, T. Bessell, S. S. Kanhere, N. Gordon, and S. Jha, "Detection and tracking using particle-filter-based wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 1332–1345, Sept. 2010.

[23] A. Oka and L. Lampe, "Distributed target tracking using signal strength measurements by a wireless sensor network," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 1006–1015, Sept. 2010.

[24] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men, "Sequential Monte Carlo for simultaneous passive device-free tracking and sensor localization using received signal strength measurements," in *Proc. of IEEE/ACM Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pp. 342–353, April 2011.

[25] A. T. Ihler, J. W. I. Fisher, 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, pp. 809–819, April 2005.

[26] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem, "Distributed online localization in sensor networks using a moving target," in *Proc. of 3rd Int. Symp. on Information Processing in Sensor Networks (IPSN)*, pp. 61–70, April 2004.

[27] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, pp. 590–599, June 1999.

[28] R. van der Merwe, A. Doucet, N. D. Freitas, and E. Wan, "The unscented particle filter," in *Proc. of Advances in Neural Information Processing Systems*, Nov. 2001.

[29] J. H. Kotecha and P. M. Djuric, "Gaussian sum particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2602–1612, Oct. 2003.

[30] 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," *IEEE Signal Processing Magazine*, vol. 22, pp. 54–69, July 2005.

[31] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1520 – 1533, Sept. 2004.

[32] M. Leng and Y.-C. Wu, "Distributed clock synchronization for wireless sensor networks using belief propagation," *IEEE Transactions on Signal Processing*, vol. 59, pp. 5404 –5414, Nov. 2011.

[33] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. S. Shamma, "Belief consensus and distributed hypothesis testing in sensor networks," in *Proc. of NESC Worskhop*, pp. 169–182, Springer Verlag, 2006.

[34] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal Processing*, vol. 57, pp. 2748–2761, July 2009.

[35] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. of the IEEE*, vol. 98, pp. 1847–1864, Nov. 2010.

[36] C. Crick and A. Pfeffer, "Loopy belief propagation as a basis for communication in sensor networks," in *Uncertainty in Artificial Intelligence*, pp. 159–166, Aug. 2003.

[37] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, pp. 2508–2530, June 2006.

[38] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004.

[39] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo: Morgan Kaufmann, 1988.

[40] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, pp. 33–46, 2007.

[41] B. Johansson, *On Distributed Optimization in Networked Systems*. PhD thesis, KTH, Stockholm, Sweden, Dec. 2008.

[42] D. Mosk-Aoyama, T. Roughgarden, and D. Shah, "Fully distributed algorithms for convex optimization problems," *SIAM Journal on Optimization*, vol. 20, pp. 3260–3279, Oct. 2010.

[43] T.-D. Pham, H. Q. Ngo, V.-D. Le, S. Lee, and Y.-K. Lee, "Broadcast gossip based distributed hypothesis testing in wireless sensor networks," in *Proc. of Int. Conf. on Advanced Technologies for Communications*, pp. 84–87, Oct. 2009.

[44] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Understanding belief propagation and its generalizations*, pp. 239–269. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[45] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Computation*, vol. 12, pp. 1–41, Jan. 2000.

[46] V. Savic, H. Wymeersch and E. G. Larsson, "Simultaneous sensor localization and target tracking in mine tunnels," *in IEEE Proc. of Intl. Conf. on Information Fusion*, July 2013.

[47] A. Chehri, P. Fortier, and P. M. Tardif, "Characterization of the ultra-wideband channel in confined environments with diffracting rough surfaces," *Wireless Personal Communications (Springer)*, vol. 62, pp. 859–877, Feb. 2012.