Distributed target tracking is a significant technique and is widely used in many applications. Combined with the interval analysis, box particle filtering (BPF) has been proposed to solve the problem of Bayesian filtering when the uncertainties in the measurements are intervals; that is, the measurements are interval-based vectors. This paper is targeted for extending the existing BPF based on a single sensor to a distributed sensor network. We propose a distributed BPF (d-BPF) that each sensor communicates with its direct neighbors to collaboratively estimate the states of the target. The feasibility of the proposed distributed BPF is justified, and some numerical simulations are presented to show its effectiveness in target tracking.

1. Introduction

Target tracking is a prevalent task in many scenarios, such as navigation, military, and transportation. In the literature, various approaches have been proposed for target tracking [1, 2]. One of the most popular approaches is the Bayesian inference [3]. Within the Bayesian framework, the posterior probability density function (pdf) of the states of the target is calculated, conditioned on the available measurements obtained by sensors. Then, based on the obtained posterior pdf, the states of the target can be estimated, correspondingly.

Reviewing the related works of target tracking based on Bayesian inference, the Kalman filter (KF) [4–6] is a classical approach. However, in the presence of the nonlinearities and non-Gaussian uncertainties, the KF becomes inefficient. To tackle this problem, extended trackers such as the extended Kalman filter (EKF) [7] and the unscented Kalman filter (UKF) [8] have been developed. Although these extended trackers can handle some nonlinear cases, they become ineffective when the high-variance measurements and state noise combined with nonlinearities exist [9]. The particle filter (PF), which represents the posterior pdf of the states by sampling, shows outstanding performance, especially for nonlinear and/or non-Gaussian cases [3]. However, the efficiency and accuracy of the PF largely depend on the number of the particles. In the case where a large number of particles are used, the computational complexity is quite high.

It is worth pointing out that all of the above-mentioned approaches can only deal with the point measurements represented by fixed vectors. However, for a large amount of target tracking problems, the uncertainties may lie in a certain region, and thus the measurements are intervals. To deal with such interval-based uncertainties, the box particle filtering (BPF) [10–12] based on the interval framework is proposed, which propagates a set of weighted boxes in a sequential way. Instead of using a large amount of point samples, much less weighted boxes are used to approximate the desired moments of the posterior pdfs.

On the other hand, distributed in-network target tracking, where a set of distributed sensors collaboratively estimate the states of target, has received much attention recently. Distributed tracking can achieve low energy consumption as well as high robustness and scalability. Owing to these merits, various distributed target tracking algorithms have been proposed, such as the distributed KF [13–15] and the distributed PF [16–19].

However, the above-distributed algorithms are only used for the cases where the system measurements are point-based vectors. Decentralized BPF that can handle interval-based uncertainties has not aroused much attention. Although
a decentralized BPF algorithm using belief propagation (BP) like message-passing algorithm is proposed in [20], it is only focused on the context of target localization but not applied to target tracking. Considering this, a distributed BPF (d-BPF) algorithm is proposed in this paper, where only the one-hop data exchange and communication between sensors and their neighbors are used to collaboratively track the states of the target. Theoretical studies and numerical simulations are then performed to show the effectiveness of the proposed algorithm.

The rest of this paper is organized as follows. In Section 2, we review some basic knowledge related to BPF. In Section 3, we derive the d-BPF and analyze the feasibility of the d-BPF. The selection of the combination weights for cooperation is also introduced in the same section. Section 4 presents some simulation examples. Finally, some conclusion is drawn in Section 5.

2. Preliminaries

The traditional Bayesian estimation deals with the point measurements, while the BPF can cope with interval-based measurements. The key idea of the BPF is to replace a particle by a multidimensional interval in the state space [11]. Therefore, the derivation of BPF depends on the theory of interval framework. So, before introducing the BPF, we overview some basic knowledge of interval analysis, including interval arithmetic and inclusion function, as well as constraint satisfaction problems and contraction.

2.1. Interval arithmetic. A real interval [x] is a nonempty set of real numbers:

$$[x] = [\underline{x}, \overline{x}] = \{ x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x} \}, \quad (1)$$

where \( \underline{x} \) and \( \overline{x} \) denote the lower and upper bounds of \([x]\), respectively. From this definition, it is natural to define set-theoretic operations, such as intersection and union, and the results of these must be intervals also. For instance, the interval union of two intervals \([x] \cup [y]\) is defined as

$$[x] \cup [y] = [[x] \cup [y]], \quad (2)$$

where the symbol “∪” denotes the union on sets and the symbol “[:]” denotes the interval hull operator returning the smallest interval enclosing \(S\), for any set \(S\) in real set \(\mathbb{R}\) [25].

The interval intersection of two intervals \([x] \cap [y]\) is defined as

$$[x] \cap [y] = [x] \cap [y], \quad (3)$$

where the symbol “∩” denotes the intersection on sets.

Interval arithmetic operations are also defined on \(\mathbb{IR}\) such that the interval result encloses all possible real results, where \(\mathbb{IR}\) denotes a set of all real-valued intervals. Given \([x] = [\underline{x}, \overline{x}]\) and \([y] = [\underline{y}, \overline{y}]\), the binary operations \{+, −, ×, \} can be defined by

$$[z] = [x] \otimes [y] = [[x] \otimes [y] | x \in [x], \ y \in [y]]], \quad (4)$$

where \(\otimes\) denotes the binary operations.

### Table 1: Interval arithmetic operations.

| Binary operator | Notation |
|-----------------|----------|
| \([x] + [y]\) | \([x + y, \overline{x} + \overline{y}]\) |
| \([x] - [y]\) | \([x - \overline{y}, \overline{x} - \overline{y}]\) |
| \([x] \times [y]\) | \([\min(x+y, \overline{x}+\overline{y}), \max(x+y, \overline{x}+\overline{y})]\) |

Some basic interval arithmetic operations are summarized in Table 1 [25]. An interval vector \([x]\) is defined as a vector with interval components and the space of all \(n_x\) dimension interval vectors is denoted by

$$[x] = [x_1] \times [x_2] \times \cdots \times [x_{n_x}]. \quad (5)$$

Similarly, interval arithmetic to boxes can be extended straightforwardly [25].

2.2. Inclusion Function. As mentioned above, in the BPF, each particle is replaced by a multidimensional interval in the state space. In order to keep consistency, it is necessary to guarantee that the result of interval operation is still an interval or an interval vector. Fortunately, an inclusion function can meet our need. To make it simple, assume \(f\) is a function from \(\mathbb{R}^n\) to \(\mathbb{R}^m\). In the BPF, when propagating box particles via the continuous nonlinear function \(f\), the mapping of the box particle is not necessary a box. But it can be approximated by an inclusion function. An interval function \([f]\) from \(\mathbb{IR}^n\) to \(IR^m\) is said to be an inclusion function for \(f\) if

$$\forall [x] \in IR^n, \quad f([x]) \subseteq [f([x])]. \quad (6)$$

An inclusion function may be very pessimistic. But the minimal inclusion function for \(f\) denoted by \([f]^*\) is unique [11].

2.3. Constraint Satisfaction Problems and Contraction Methods. Consider variables \(x_i\) in \(\mathbb{R}, i \in \{1, \ldots, n_x\}\), linked by \(m\) relations (or constraints) of the form

$$f_j(x_1, \ldots, x_{n_x}) = 0, \quad j = 1, \ldots, m. \quad (7)$$

Define the vector \(x\) as \(x = (x_1, \ldots, x_{n_x})^T\), where \(T\) denotes transposition, and let \(f\) is a vector function whose components are \(f_j\), \(j = 1, \ldots, m\). Then, the \(m\) constraints can be rewritten as

$$f(x) = 0. \quad (8)$$
This corresponds to a constraint satisfaction problem (CSP), which is denoted as $\mathcal{R}$; that is,

$$\mathcal{R} : (f(x) = 0, \ x \in [x]),$$

and the solution set of $\mathcal{R}$ is defined as

$$\mathcal{S} = \{x \in [x] | f(x) = 0\}.$$

Note that the set $\mathcal{S}$ is not necessary a box. Within the interval framework, the term contracting refers to solving a CSP, that is, finding the smallest box $[x]^T \subset [x]$ that constitutes an outer approximation of $\mathcal{S}$ such that $\mathcal{S} \subset [x]^T \subset [x]$. In the BPF, a general and well-known contraction method, the constraints propagation (CP) technique is applied [12]. Illustrative examples of CP method can be found in [11].

3. Target Tracking with Distributed Box Particle Filtering

Assume that there are $N$ static sensors and a moving target. These sensors are randomly distributed over a geometrical field with known positions and can periodically obtain measurements from the target. Our goal is to track the target in a distributed way so as to improve the tracking accuracy. We firstly derive the d-BPF algorithm and then give the Bayesian justification for it. Finally, some choices for the combination weight are given.

3.1. Distributed Box Particle Filtering. Consider the following state-space model:

$$x_{t+1} = f(x_t, v_{t+1}),$$

$$y_{k,t+1} = g_k(x_{t+1}) + w_{k,t+1}, \quad k = 1, \ldots, N,$$

where $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x}$ is a nonlinear transition function from time $t$ to $t+1$ with $n_x$ being the dimension of the states. $v_{t+1}$ denotes an independent identically distributed (i.i.d.) process noise. Function $g_k : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ defines the relationship between the states and the measurements in node $k$ at time $t+1$, where $n_y$ is the dimension of the measurements. $w_{k,t+1}$ is an i.i.d. measurement noise of node $k$. Here, we consider interval-based uncertainties and thus interval measurements. That is, each sensor $k$ provides interval measurements $\{y_{k,t+1}\}$, which can be expressed as $\{y_{k,t+1}\} = [y_{k,t+1} + \Delta_1, y_{k,t+1} + \Delta_2]$, where $\Delta_1$ and $\Delta_2$ denote the lower bound and the higher bound uncertainties, respectively. It is assumed that each node knows the locations of its neighbors at each time $t$.

In [10, 11], target tracking with a single sensor using BPF algorithm has been studied. Different from the PF, in the BPF, each particle is a box, which takes into account the imprecision caused by the model errors [11]. In this manner, the contraction step is similar to the variance matrix measurement update step of the Kalman filtering. Therefore, we follow a similar guideline of distributed Kalman filtering [26] to derive a distributed BPF algorithm for target tracking in a sensor network.

The d-BPF algorithm can be divided into five steps: initialization, time update and measurements broadcast, measurement update, estimation fusion, and resampling, where the second to the fourth steps are shown schematically in Figure 1.

(i) Initialization. At the beginning, the box particles in each node have to be initialized. Taking node $k$ as an example, it can be initialized by $M$ equally weighted and mutually disjoint boxes represented by $\{[x_{k,0}^{(m)}, \omega_{k,0}^{(m)}] \}_{m=1}^M$, where $[x_{k,0}^{(m)}]$ denotes the box particles and $\omega_{k,0}^{(m)}$ denotes the corresponding weights. The BPF is superior to the PF in that it can cover larger prior uncertainty region with much fewer box particles.

(ii) Time Update and Measurements Broadcast. Referring to Figure 1(a), each node obtains a set of predicted box particles according to the transition function locally and broadcasts measurements to its close neighbors simultaneously. Knowing a set of weighted box particles $\{[x_{k,t+1}^{(m)}, \omega_{k,t+1}^{(m)}] \}_{m=1}^M$ at time step $t$ and assuming that the system noise is known to be enclosed in the interval $[v_{t+1}]$, the predicted boxes at step $t+1$ can be calculated by

$$[x_{k,t+1}^{(m)}] = [f] ([x_{k,t}^{(m)}], [v_{t+1}]), \quad m = 1, \ldots, M.$$  \hspace{1cm} (12)

It is worth mentioning that the inclusion function $[f]$ needs to be used for ensuring that the predicted box particle is still an interval when propagating the box particles.

(iii) Measurement Update. As shown in Figure 1(b), after performing time update locally and broadcasting measurements to the neighbors, the weights of the predicted box particles in each node are then updated using the new available measurements from its neighbors (including itself) at time step $t+1$. This is an important step in the d-BPF, which is different from the BPF. For all box particles $(m = 1 \cdots M)$ in node $k$, we have the predicted box measurements expressed as

$$[y_{l,t+1}^{(m)}] = [g_k] ([x_{k,t+1}^{(m)}]), \quad \forall l \in \mathcal{N}_k,$$  \hspace{1cm} (13)

where $[g_k]$ is an inclusion function for $g_k$ and $\mathcal{N}_k$ represents the neighbors of node $k$.

Likelihood factors are calculated using innovation quantitites, which reflects the proximity between the measured box and the predicted box measurement [11, 20]. The innovation is represented by the intersection between these two boxes, expressed as

$$[r_{l,t+1}^{(m)}] = [y_{l,t+1}^{(m)}] \cap [y_{l,t+1}], \quad \forall l \in \mathcal{N}_k.$$  \hspace{1cm} (14)

The likelihood factor is calculated based on the following idea: if predicted measurement of a box particle is included in the new measurement box, its updated weight should be larger. In contrast, if its predicted measurement does not have any intersection with the new measurement box, its updated weight should be equal to zero [11]. Furthermore, the contraction step is used to remove the intervals in the box particles that are not consistent with the measurements.
Based on these concepts, for each node $k$, the likelihood of a box particle at time $t+1$ can be expressed in the form

$$L_{k,t+1}^{(m)} = \prod_{j=1}^{n_k} L_{k,t+1}^{(m),j}$$

(15)

where

$$L_{k,t+1}^{(m),j} = \left| \frac{x_{k,t+1}^{(m)}(j)}{x_{k,t+1}^{(m)}} \right|$$

(16)

Note that the term $[x_{k,t+1}^{(m)}(j)]$ in (16) represents the $j$th dimension of updated $m$th box particle after contraction using all the new available measurements that node $k$ receives, $| \cdot |$ denotes the length of the interval. The implementation of contraction step is given in Algorithm 1.

**Algorithm 1** (contraction in measurement update). Consider the following.

**Input**

(i) $\{x_{k,t+1}^{(m)}, \omega_{k,t+1}^{(m)}\}_{m=1}^{M}$: a set of weighted boxes,
(ii) $\mathcal{Y}_{k,t+1}$: new available measurements at node $k$.

**(Nested loops)**

For $l \in \mathcal{N}_k$

(1) Predicted measurement:

$$[y_{l,t+1}^{(m)}] = [g] \left( [x_{k,t+1}^{(m)}(j)] \right), \text{ for } m = 1, \ldots, M.$$  

(17)

(2) Innovation:

$$[r_{l,t+1}^{(m)}] = [y_{l,t+1}^{(m)}] \cap [y_{l,t+1}], \text{ for } m = 1, \ldots, M.$$  

(18)

(3) Contraction:

If $[r_{l,t+1}^{(m)}] \neq 0$, then contract $[x_{k,t+1}^{(m)}]$ using $[r_{l,t+1}^{(m)}]$ and CP algorithm to obtain $[\tilde{x}_{k,t+1}^{(m)}]$, else $[\tilde{x}_{k,t+1}^{(m)}] = 0$, for $m = 1, \ldots, M$.

(4) Return to step 1, replace $[x_{k,t+1}^{(m)}]$ by $[\tilde{x}_{k,t+1}^{(m)}]$.

**Endfor**

**Output**

(i) $\{\tilde{x}_{k,t+1}^{(m)}, \omega_{k,t+1}^{(m)}\}_{m=1}^{M}$: a set of weighted boxes after contraction.
Based on the likelihood in (15), the update weights for each node $k$ at time $t + 1$ can be calculated by
\[
\omega_{k,t+1}^{(m)} = \omega_{k,t+1}^{(m)} \cdot \omega_{k}^{(m)},
\]
and then the weights are normalized by the following:
\[
\bar{\omega}_{k,t+1}^{(m)} = \frac{\omega_{k,t+1}^{(m)}}{\sum_{m=1}^{M} \omega_{k,t+1}^{(m)}} .
\]
Hence, a set of updated weighted box particles \{\[\bar{x}_{k,t+1}^{(m)}, \bar{\omega}_{k,t+1}^{(m)}\]\}$_{m=1}^{M}$ are obtained. Meanwhile, local estimation in each node can be approximated by the weighted particles as
\[
\Phi_{k,t+1} = \sum_{m=1}^{M} \frac{\omega_{k,t+1}^{(m)}}{\omega_{k,t+1}^{(m)}} \bar{x}_{k,t+1}^{(m)},
\]
where $\bar{x}_{k,t+1}^{(m)}$ is the center of the $m$th box particle $\bar{x}_{k,t+1}^{(m)}$ at time step $t + 1$ in node $k$.

It is remarked that the box particles in each node are different after contraction, since the new available measurements are different in each node. So, it is improper to transmit weights to the neighbors as that performed in the distributed PF. Instead, here we let nodes only broadcast measurements to their close neighbors in a distributed way and the box particles at each node are obtained based on the information in its neighborhood.

(iv) Estimation Fusion. In Figures I(c) and I(d), a combination step is applied and it attempts to achieve the new estimate $\bar{x}_{k,t+1}$ that approximates the global performance via local node interactions; that is,
\[
\bar{x}_{k,t+1} = \sum_{l \in J_k} a_{k,l} \Phi_{l,t+1},
\]
where $a_{k,l}$ is the $(k,l)$ element of the combination matrix $A \in \mathbb{R}^{N \times N}$ denoting the weight between $k$ and $l$, which satisfies
\[
\begin{align*}
1^T A &= 1^T, \\
a_{k,l} &= 0, \quad \text{if } l \notin J_k, \\
a_{k,l} \geq 0, \quad \forall k,l,
\end{align*}
\]
where $1$ is an $N \times 1$ column vector with unity entries. There are several ways to select $a_{k,l}$, and this will be described elaborately later.

(v) Resampling. The resampling is performed to introduce variety while preventing the degeneration of the box particles. Similar to the BPF algorithm with a single sensor [10, 11], the multinomial resampling combined with a subdivision step is applied, in which a box particle with high weight is divided by the corresponding number of smaller boxes, instead of obtaining the duplicated particles after the resampling step.

The d-BPF algorithm implements these steps in each time slot and a series of weighted box particles and the state estimation can be calculated sequentially in each node. To summarize, its implementation procedure is given in Algorithm 2.

Algorithm 2 (distributed box particle filtering). Consider the following.

(1) Initialization. At each node $k$, set $t = 0$ and generate $M$ box particles \{\[x_{k,0}^{(m)}, \omega_{k,0}^{(m)}\]\}$_{m=1}^{M}$ with empty intersection and equal weights.

(2) Time Update and Measurements Broadcast. Assume we have a set of weighted box particles \{\[x_{k,t}^{(m)}, \omega_{k,t}^{(m)}\]\}$_{m=1}^{M}$ in node $k$ at time $t$.

Calculate the predicted weighted box particles through the transition function in each node locally
\[
[x_{k,t+1}^{(m)}] = f([x_{k,t}^{(m)}, [v_{t+1}]]), \quad \text{for } m = 1, \ldots, M.
\]

Meanwhile, each node broadcasts measurements to its close neighbors.

(3) Measurement Update

Contraction. Use Algorithm 1 to obtain a set of weighted box particles \{\[x_{k,t+1}^{(m)}, \omega_{k,t+1}^{(m)}\]\}$_{m=1}^{M}$.

Likelihood. Consider
\[
L_{k,t+1}^{(m)} = \prod_{j=1}^{n_t} L_{k,t+1}^{(m,j)}, \quad \text{for } m = 1, \ldots, M,
\]
where
\[
L_{k,t+1}^{(m,j)} = \frac{\left[\bar{x}_{k,t+1}^{(m)}(j)\right]}{\left[x_{k,t+1}^{(m)}(j)\right]}.
\]

Weight Update. Consider
\[
\omega_{k,t+1}^{(m)} = L_{k,t+1}^{(m)} \omega_{k,t}^{(m)}, \quad \text{for } m = 1, \ldots, M.
\]

Weight Normalization. Consider
\[
\bar{\omega}_{k,t+1}^{(m)} = \frac{\omega_{k,t+1}^{(m)}}{\sum_{m=1}^{M} \omega_{k,t+1}^{(m)}}, \quad \text{for } m = 1, \ldots, M.
\]

Local State Estimate. Consider
\[
\Phi_{k,t+1} = \sum_{m=1}^{M} \frac{\omega_{k,t+1}^{(m)}}{\omega_{k,t+1}^{(m)}} x_{k,t+1}^{(m)}.
\]

(4) Estimation Fusion. Consider
\[
\bar{x}_{k,t+1} = \sum_{l \in J_k} \frac{a_{k,l}}{a_{k,l}} \Phi_{l,t+1}.
\]

(5) Resampling. Combine subdivision method for resampling to create $M$ new box particles with the same weights.

(6) Set $t = t + 1$, and go to step 2 until a predefined number $t_{end}$.
3.2. Distributed Box Particle Filtering Analysis within Bayesian Framework. In this section, similar to the proof of BPF [10, 11], we present the Bayesian justification for d-BPF algorithm. To simplify the representation, we denote the set of all the states and measurements obtained until time $t$ by $X_t = \{x_i, i = 1, \ldots, t\}$ and $Y_t = \{y_i, i = 1, \ldots, t\}$, respectively. Our objective is to estimate the predicted distribution $p(x_{t+1} \mid Y_t)$ and the posterior distribution $p(x_{t+1} \mid Y_{t+1})$, recursively. Once the posterior pdf $p(x_{t+1} \mid Y_{t+1})$ is estimated, the system states can be calculated, correspondingly.

Within the framework of Bayesian inference, the posterior pdf of the state vector $p(x_{t+1} \mid Y_{t+1})$ is given by

$$p (x_{t+1} \mid Y_{t+1}) = \frac{1}{\alpha_{t+1}} p (x_{t+1} \mid Y_t) p (y_{t+1} \mid x_{t+1}),$$  \hspace{1cm} (31)

$$p (x_{t+1} \mid Y_t) = \int_{\mathbb{R}^n} p (x_{t+1} \mid x_t) p (x_t \mid Y_t) d x_t,$$  \hspace{1cm} (32)

where $\alpha_{t+1} = \int_{\mathbb{R}^n} p (x_{t+1} \mid Y_t) p (y_{t+1} \mid x_{t+1}) d x_{t+1}$ is a normalization factor. The recursion can be updated with an initial prior pdf $p(x_0)$, for example, a uniform distribution over some region of the state space. Equations (31) and (32) are known as the measurement update and the time update, respectively.

Let a set of box particles be represented by a mixture of uniform pdfs. Comparing with the Gaussian mixture pdf, the uniform pdf is not only simple, but also owns a remarkable adaptation ability to strong nonlinear cases. Based on the interval analysis and by choosing box supports, d-BPF can propagate pdfs through linear, nonlinear, and even differential functions. More specifically, since the particles in the d-BPF algorithm are boxes and with the help of the interval analysis, the box particles can propagate sequentially with the uniform pdf.

Referring to [11], the uniform pdf sum representation of $x$ can be given as

$$p (x) = \sum_{m=1}^{M} \omega^{(m)} U_{[x^{(m)}]} (x),$$  \hspace{1cm} (33)

where $U_{[x^{(m)}]}$ denotes the uniform pdf of the box particle $[x^{(m)}]$ and $M$ denotes the number of mixture components. According to (33), the relative pdfs in the Bayesian inference can be represented by a set of weighted box particles.

Now we can deduce the Bayesian expression of the d-BPF algorithm. Assume that a set of weighted box particles $\{U_{[x^{(m)}]} (x_k), \omega^{(m)}_k\}_{k=1}^{M}$ of node $k$ is obtained at time $t$. Consequently, the pdf of the state at time $t$ can be represented by

$$p (x_t \mid Y_{t \in \mathcal{S}_k,t}) = \sum_{m=1}^{M} \omega^{(m)}_k U_{[x^{(m)}]} (x_k).$$  \hspace{1cm} (34)

Then, the time update step at each node is carried out locally. Taking node $k$ as an example, according to (32), the pdf of the states can be predicted by

$$p (x_{k,t+1} \mid Y_{le,k,t+1}) = \int_{\mathbb{R}^n} p (x_{k,t+1} \mid x_{k,t}) p (y_{le,k,t+1} \mid x_{k,t+1}) d x_{k,t}$$

$$= \int_{\mathbb{R}^n} \sum_{m=1}^{M} \omega^{(m)}_k U_{[x^{(m)}]} (x_k) p (x_{k,t+1} \mid x_{k,t}) d x_{k,t}$$

$$= \sum_{m=1}^{M} \omega^{(m)}_k U_{[x^{(m)}]} (x_k) p (x_{k,t+1} \mid x_{k,t}).$$  \hspace{1cm} (35)

Referring to [11], we can derive

$$\int_{[x^{(m)}]} p (x_{k,t+1} \mid x_{k,t}) U_{[x^{(m)}]} (x_k) d x_{k,t}$$

$$= U_{[f(|x^{(m)}|, y_{le,k,t+1})]} (x_{k,t+1}).$$

Based on (36), (35) can be rewritten as

$$p (x_{k,t+1} \mid Y_{le,k,t+1}) = \sum_{m=1}^{M} \omega^{(m)}_k U_{[f(|x^{(m)}|, y_{le,k,t+1})]} (x_{k,t+1})$$

$$= \sum_{m=1}^{M} \omega^{(m)}_k U_{[y^{(m)}]} (x_{k,t+1}),$$  \hspace{1cm} (37)

where the second equality holds according to (12). So far, we derive that the predicted pdf after time update in each node can still be represented by a set of weighted box particles, where the box particles change according to the transition function, while the weights keep invariant.

In the measurement update step, each node spreads its observations to its neighbors. When nodes receive information from their neighbors, the measurement update step can be implemented. Consequently, according to (31), the posterior pdf of the states can be computed by

$$p (x_{k,t+1} \mid Y_{le,k,t+1}) = \frac{1}{\alpha_{t+1}} p (y_{le,k,t+1} \mid x_{k,t+1}) \cdot p (x_{k,t+1} \mid Y_{le,k,t+1})$$

$$= \frac{1}{\alpha_{t+1}} \prod_{l \in \mathcal{S}_k} \{ p (y_{l,t+1} \mid x_{k,t+1}) \} \cdot \prod_{m=1}^{M} \omega^{(m)}_k U_{[y^{(m)}]} (x_{k,t+1}),$$  \hspace{1cm} (38)

where $\alpha_{t+1} = \int_{\mathbb{R}^n} p (x_{k,t+1} \mid Y_{le,k,t+1}) p (y_{le,k,t+1} \mid x_{k,t+1}) d x_{k,t+1}$ is a normalized factor.

Referring to Algorithm 2, the measurement update step consists of two main steps, that is, contraction and weight update. The former is used to contract a set of new box particles and the latter is used to calculate updated weights.
Once a set of new weighted box particles is obtained, the estimation can be calculated naturally. We first focus on the contraction step. Since the measurements are intervals, the measurement likelihood function \(p(y_{t+1} | x_{k,t+1})\) is taken to be one component with uniform distribution, which is expressed as
\[
p(y_{t+1} | x_{k,t+1}) = U\left(y_{t+1} \left| \mathbf{g} (x_{k,t+1}) \right. \right).
\] (39)

Consequently, (38) can be rewritten as
\[
p\left(x_k,t+1 \mid Y_{i \in \mathcal{E}_k} \right) = \frac{1}{\alpha_{t+1}} \prod_{l \in \mathcal{E}_k} \left\{ U\left[ y_{l} \mid \mathbf{g} (x_{l,t+1}) \right] \right\} \sum_{m=1}^{M} \omega_k^{(m)} U\left[ x_{k,t+1} | \mathbf{g} (x_{k,t+1}) \right] U\left[ y_{k} | \mathbf{g} (x_{k,t+1}) \right] \prod_{l \in \mathcal{E}_k} U\left[ y_{l} | \mathbf{g} (x_{l,t+1}) \right] U\left[ y_{k} | \mathbf{g} (x_{k,t+1}) \right].
\] (40)

We use \(U[y]\) to denote \(U[y_{t+1}] \left| \mathbf{g} (x_{k,t+1}) \right. \) for short. Then, (40) can be rewritten as
\[
p\left(x_{k,t+1} \mid Y_{i \in \mathcal{E}_k} \right) = \frac{1}{\alpha_{t+1}} \sum_{m=1}^{M} \omega_k^{(m)} \prod_{l \in \mathcal{E}_k} \left\{ U[y_{l}] \right\} U\left[ x_{k,t+1} | \mathbf{g} (x_{k,t+1}) \right] U\left[ y_{k} | \mathbf{g} (x_{k,t+1}) \right] \prod_{l \in \mathcal{E}_k} U\left[ y_{l} | \mathbf{g} (x_{l,t+1}) \right] U\left[ y_{k} | \mathbf{g} (x_{k,t+1}) \right]
\] (41)

where \(l_i \in \mathcal{N}_k, i = 1 \cdots n_k\) with \(n_k\) being the degree of node \(k\). The term \(U[y_{l}] \left| U[\mathbf{g} (x_{l,t+1})] \right. \) in (41) is a function with a support being the following set:
\[
x_{k,t+1} \in U\left[ x_{k,t+1} | \mathbf{g} (x_{k,t+1}) \right] = \left\{ x_{k,t+1} \mid \exists w_{l_i} \right\}
\] (42)

Note that the result of \(U[y_{l}] \left| U[\mathbf{g} (x_{l,t+1})] \right. \) is also a support, which has the similar expression as \(U[\mathbf{g} (x)]\). Repeating this recursively, the result of \(U[y_{l_i}] U[y_{l_{i-1}}] \cdots U[y_{l_1}]\) is also a support or a box particle. In other words, from (41), it can be seen that the predicted boxes \([x_{k,t+1}]_{m=1}^{M}\) of node \(k\) have to be contracted with respect to the new available measurements in node \(k\), including the measurements of its neighbors and itself. By using the contraction step, we obtain a set of new boxes \([x_{k,t+1}]_{m=1}^{M}\) for the posterior pdf \(p(x_{k,t+1} | Y_{i \in \mathcal{E}_k}; t+1)\) at time \(t+1\).

After contraction, the corresponding weights can be calculated according to (16). Finally, the posterior pdf at time \(t+1\) can be expressed as
\[
p\left(x_{k,t+1} \mid Y_{i \in \mathcal{E}_k} \right) = \sum_{m=1}^{M} \omega_k^{(m)} U\left[ x_{k,t+1} | \mathbf{g} (x_{k,t+1}) \right] U\left[ y_{k} | \mathbf{g} (x_{k,t+1}) \right] \prod_{l \in \mathcal{E}_k} U\left[ y_{l} | \mathbf{g} (x_{l,t+1}) \right] U\left[ y_{k} | \mathbf{g} (x_{k,t+1}) \right],
\] (43)

which has the same form as \(p(x_{k,t} | Y_{i \in \mathcal{E}_k})\) at time \(t\). That is to say, the posterior distribution \(p(x_{k,t+1} | Y_{i \in \mathcal{E}_k})\) can be estimated recursively following the framework of Bayesian inference, and thus the feasibility of the d-BPF algorithm is justified.

Remark 3. It is also remarked that the d-BPF is superior to the noncooperation BPF (Nc-BPF). On the one hand, it can be seen from (42) that the contraction uses the measurements to remove the inconsistent part of each box particle. As a result, combined with measurements of neighbors, the box particles after contraction in each node would have less or at least no more inconsistent part than those without cooperation; that is, the uncertainty of box particle is decreased. On the other hand, according to (16), in the d-BPF algorithm, we can deduce that the box particles with large consistent part have larger normalized weights while those with small consistent part acquire smaller normalized weights, comparing with those in a noncooperation algorithm. As a conclusion, a better set of normalized weights of the box particles can be obtained than that without cooperation and thus the estimate outperforms the noncooperation one.

3.3. Selection of Combination Weight. The entities of combination matrix \(A\) represent the weights used to combine the estimates of the neighbors. Some popular choices in the context of graph theory, average consensus, and diffusion adaptive networks are summarized in Table 2, where \(d_k\) denotes the degree of node \(k\). In this paper, we employ the metropolis rule as it determines the weights only according to the degrees of the neighbors. According to [23], weights chosen by the metropolis rule can guarantee convergence provided that the graph is not bipartite.

Moreover, considering that the sensors are usually equipped with mobility, a dynamic network topology is also

| Name and reference | Rule \((l \in \mathcal{N}_k), l \neq k\) |
|-------------------|-------------------|
| Uniform [21]      | \(a_{lk} = 1/d_k\) |
| Laplacian [22, 23]| \(a_{lk} = 1/d_{\text{max}}\) |
| Maximum [24]      | \(a_{lk} = 1/N\)   |
| Metropolis [23]   | \(a_{lk} = 1/\max(d_k,d_l)\) |
considered. To model this, we assume that the links and nodes are random entities for simplicity. That is, at every time instant \( t \), the undirected random link weight \( a_{kl}(t) = a_{lk}(t) \), which connects node \( k \) to node \( l \), is set to either a nominal value \( a_{kl} = a_{lk} \) with probability \( p_{kl} = p_{lk} \) or zero with probability \( q_{kl} = 1 - p_{kl} \); that is, the combination matrix \( A \) is selected by

\[
[A]_{kl} = \begin{cases} 
    a_{kl} & \text{with probability } p_{kl}, \\
    0 & \text{with probability } q_{kl} 
\end{cases} \quad (44)
\]

where \( a_{kl} \) is selected according to the metropolis rule mentioned above.

**Remark 4.** Note that when the sensor network is fully connected, the combination matrix selected by the Metropolis rule becomes \( A = (1/N)I_N \), where \( I_N \) is an identity \( N \times N \) matrix. In this case, the estimate of each node is the same since every node collects the information of all the sensors. That is to say, each sensor can be considered as the fusion center in the centralized processing, and consequently the d-BPF reduces to the centralized BPF (c-BPF).

### 4. Simulation

In this section, we perform two examples of target tracking using the d-BPF algorithm proposed in Section 3. In these two examples, we consider the target tracking in a 2D plane. Assume there are \( N = 50 \) sensors randomly deployed in the area of \( 300 \text{ m} \times 500 \text{ m} \). There is one target in the area and its state information at time \( t \) is denoted by \( x_t = [x_t, \dot{x}_t, y_t, \dot{y}_t]^T \), where \( x_t \) and \( y_t \) represent the position of the target and \( \dot{x}_t \) and \( \dot{y}_t \) denote the velocity of the target, respectively. In the following, we will discuss these two examples, respectively.

#### 4.1. Example 1

In the first example, we consider a linear and Gaussian system that can be expressed as

\[
x_{t+1} = Fx_t + v_{t+1}, \quad (45)
\]

The target is moving according to the nearly constant velocity model with a transition density function \( p(x_{t+1} \mid x_t) = N(x_{t+1}, Fx_t, Q) \):

\[
F = I_2 \otimes \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix},
\]

\[
Q = I_2 \otimes \begin{pmatrix} T^3 & T^2 \\ \frac{3}{2} & 2 \end{pmatrix} \cdot \bar{\theta}, \quad (46)
\]

where \( Q \) is the covariance matrix of the system noise, \( I_2 \) is a \( 2 \times 2 \) identity matrix, \( \otimes \) is the Kronecker product, \( T \) denotes the sampling interval (here set to 1 second), and \( \bar{\theta} \) denotes the process noise intensity (here set to 0.005). Initially, at \( t = 0 \), the target is located at position \( (550, 300) \text{ m} \) and is moving with the velocity \( (-5, -8.5) \text{ m/s} \). The target is tracked for \( t = 50 \text{ s} \).

![The sensor network with 50 nodes.](image)

Figure 2: The sensor network with 50 nodes.

A nonlinear and Gaussian measurement function \( g \) is also used. Each node uses the same measurement model expressed as

\[
\begin{pmatrix} r_{t+1} \\ \dot{r}_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} \sqrt{x_{t+1}^2 + y_{t+1}^2} \\ \frac{x_{t+1} \dot{x}_{t+1} + y_{t+1} \dot{y}_{t+1}}{\sqrt{x_{t+1}^2 + y_{t+1}^2}} \\ \arctan \left( \frac{y_{t+1}}{x_{t+1}} \right) \end{pmatrix}, \quad (47)
\]

where \( r_{t+1}, \dot{r}_{t+1}, \theta_{t+1} \) denote range, range rate, and azimuth, respectively. The measurement noise is ordinary additive zero-mean white Gaussian denoted by \( \omega_{k,t+1} \), with a covariance matrix \( \Sigma = \text{diag}(\sigma_r^2, \sigma_r^2, \sigma_\theta^2) \), where \( \sigma_r = 1 \text{ m} \), \( \sigma_r = 0.01 \text{ m/s} \), and \( \sigma_\theta = 0.025 \text{ rad} \). The sensors provide interval measurements with interval length \( \Delta = |\Delta \dot{r}, \Delta \dot{\theta}, \Delta \theta| \), where \( \Delta \dot{r} = 20 \text{ m/s} \), \( \Delta \dot{\theta} = 0.2 \text{ rad} \), and \( \Delta \theta = 0.4 \text{ rad} \) are the length of intervals in range, range rate, and azimuth, respectively. That is, the interval-based measurement of node \( k \) at time \( t+1 \) lies within

\[
\left[ y_{k,t+1} \right] = \left[ g_k(x_{t+1}) + \omega_{k,t+1} + \Delta \dot{y} \right] = \begin{pmatrix} r_{t+1} \\ \frac{x_{t+1} \dot{x}_{t+1} + y_{t+1} \dot{y}_{t+1}}{\sqrt{x_{t+1}^2 + y_{t+1}^2}} \\ \arctan \left( \frac{y_{t+1}}{x_{t+1}} \right) \end{pmatrix}, \quad (48)
\]

where \( \Delta_1 = -(1/2)\Delta \) and \( \Delta_2 = (1/2)\Delta \) denote the lower bound and the higher bound uncertainties, respectively.

In the following, both the simulations on static network topology and dynamic network topology are represented, respectively.

#### 4.1.1. Case 1: Static Network Topology

Firstly, we consider a fixed network topology as shown in Figure 2. The positions of these sensors are perfectly known and each sensor has the knowledge of the positions of its neighbors. The experiment is implemented using MATLAB and the so-called toolbox INTLAB [27]. The results are averaged over 50 experiments. To evaluate the performance of the proposed d-BPF, the root-mean-square error (RMSE) is used, which is defined as

\[
\text{RMSE}_i = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N} \sum_{k=1}^{N} \text{RMSE}^T_{k,i} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N} \sum_{k=1}^{N} \left\| x_t^i - \tilde{x}_{k,t}^i \right\|^2, \quad (49)
\]
where \( \Gamma \) denotes the total number of experiments and the term \( \bar{X}_{k,t} \) represents the estimation of node \( k \) at time \( t \) in the \( \tau \)th experiment.

In this example, we use 32 box particles for the d-BPF algorithm. Figure 3 shows tracking trajectory of the d-BPF of node 1 for a single run based on measurements generated from (47). Circle marks represent the true target trajectory, while the “plus” represents the estimated trajectory. From Figure 3, it is observed that the d-BPF is able to track the trajectory of target accurately. Note that the results for different sensors are similar, and thus they are omitted here.

Figure 4 shows the averaged performance of the entire network over 50 Monte Carlo simulations. For performance comparison, the results of the Nc-BPF, the c-BPF, and distributed PF (d-PF) with 300 particles and 5000 particles are also presented. For clarity, the results are presented by taking the Napierian logarithm. It is observed that the d-BPF has significantly reduced the RMSE comparing to the Nc-BPF. Although it is a little inferior to the c-BPF from the viewpoint of RMSE, less communication cost is taken since the number of links is greatly smaller than that in the c-BPF.

The performance of the d-PF is largely dependent on the number of the particles. As shown in Figure 4, the d-BPF with only 32 box particles outperforms the d-PF with 300 particles. Although the accuracy of d-BPF is worse than the d-PF with 5000 particles, it takes much less computation time than the d-PF by using fewer box particles. It is also worth mentioning that, in order to make a comparison between d-BPF and d-PF algorithms, the noise we used in the simulation is Gaussian. It is a special case that we can use cumulative distribution function (cdf) to calculate weights so that the d-PF can deal with the interval measurements. This does not contradict with the aforementioned statement that the d-PF is in general unsuitable for the interval measurements, since the prior pdf of noise may be non-Gaussian or even unknown. However, the d-BPF can handle these cases very well. Therefore, it is where the d-BPF is expert.

4.1.2. Case 2: Dynamic Network Topology. In the second case, we simulate the d-BPF over a dynamic sensor network. In the simulation, the propagation model is also the constant velocity model defined in (45). The measurement model defined in (47) is used. We also use 32 box particles for d-BPF algorithm. Assume that all the link probabilities between two nodes are equivalent and denoted as \( p_{kl} = p \), and \( p \) keeps invariant in each simulation. Note that when \( p \) is close to 0, the network almost reduces to a noncooperation one, that is, Nc-BPF. When \( p = 1 \), the network is fully connected, and the result corresponds to the c-BPF.

Figures 5(a) and 5(b) show the average position RMSE and average velocity RMSE, respectively, using different link probabilities \( p \) over 50 Monte Carlo simulations, while Table 3 shows the corresponding average RMSE for \( t \in [20, 50] \) s. From Figure 5, we can see that the performance of d-BPF over dynamic network depends on the network topology in each time slot, which is dependent on the value of the link probability \( p \). With the increasing of \( p \), the performance of the d-BPF is increased. Moreover, initially, when \( p \) is relatively small, for example, \( p < 0.2 \), the average RMSE is reduced greatly compared to the Nc-BPF. After that, with the increasing of \( p \), the performance of d-BPF improves slightly. When \( p \) is larger than about 0.6, further increasing the value of \( p \), the performance of d-BPF does not improve much and converges to the c-BPF. It is noted that although the networks with larger link probabilities \( p \) show better estimation performance, their communication cost is also increased. Therefore, in practice, we should make a tradeoff between the estimation performance and the cost.

4.2. Example 2. In the second example, we consider the target tracking of a variable velocity model, expressed as

\[
x_{t+1} = F x_t + Gu + v_{t+1}.
\]

So the target is moving with a transition density function \( p(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1}, F x_t + Gu, Q) \):

\[
G = \begin{pmatrix} 5 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \end{pmatrix}^T,
\]

\[
u = (-0.1 \ 0.1)^T,
\]

where \( F \) and \( Q \) are defined in (46). The measurement function \( g \) is selected the same as (47).

In the simulation, the same network as given in Figure 2 is used. The initialization and other parameters are set the same as those in Example 1. We also use 32 box particles for the d-BPF algorithm in this example.

Figure 6 shows the tracking trajectory of the d-BPF of node 1 for a single run. From Figure 6, it is observed that the d-BPF is also successful in tracking the trajectory of the target even if the velocity of target is time-varying. Figure 7 shows the averaged performance of the entire network over 50 Monte Carlo simulations. The results of the Nc-BPF, the c-BPF, and distributed PF (d-PF) are also presented. Note that the results are also presented by taking the Napierian logarithm. Similar to Example 1, it is observed that the d-BPF
Figure 4: Simulation results for Example 1. (a) Average position RMSE over the entire network as a function of time. (b) Average velocity RMSE over the entire network as a function of time.

Figure 5: Simulation results for Case 2. (a) Average position RMSE over the entire network with different link probabilities $p$ as a function of time. (b) Average velocity RMSE over the entire network with different link probabilities $p$ as a function of time.

Table 3: Average position RMSE and velocity RMSE with different link probability $p$.

|                  | Nc-BPF, $p = 0$ | d-BPF, $p = 0.02$ | d-BPF, $p = 0.05$ | d-BPF, $p = 0.1$ | d-BPF, $p = 0.2$ | d-BPF, $p = 0.6$ | c-BPF, $p = 1$ |
|------------------|-----------------|-------------------|-------------------|------------------|------------------|------------------|----------------|
| Position RMSE [m] | 0.744           | 0.104             | 0.075             | 0.051            | 0.037            | 0.031            | 0.025          |
| Velocity RMSE [m/s] | 0.516       | 0.078             | 0.060             | 0.047            | 0.039            | 0.030            | 0.024          |
still significantly reduces the RMSE comparing with the Nc-BPF in this case.

5. Conclusion

In this paper, based on the traditional box particle filtering and distributed processing, we have proposed a distributed box particle filter for target tracking, namely, d-BPF, to handle the cases where the measurements are intervals-based vectors. The feasibility of the proposed d-BPF algorithm is justified. The simulations for tracking both constant and variable velocity models are carried out. From the simulation results, we find that the tracking performance of our proposed d-BPF is significantly superior to that of the Nc-BPF. Moreover, from the simulation results with a dynamic network topology, we find that the tracking performance is improved with the increasing of the link probability. It is close to the c-BPF when the link probability is much larger, while the communication cost is increased simultaneously. Therefore, in practice, we should make a balance between the tracking performance and its cost.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.
Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grants nos. 6171153 and 61471320).

References

[1] H. Grabner and H. Bischof, "On-line boosting and vision, computer vision and pattern recognition," in Proceedings of the IEEE Conference on Computer Society, vol. 1, pp. 260–267, June 2006.

[2] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 5, pp. 564–577, 2003.

[3] A. Doucet, N. de Freitas, and N. Gordon, Sequential Monte Carlo Methods in Practice, Springer, New York, NY, USA, 2001.

[4] G. Welch and G. Bishop, An Introduction to the Kalman Filter, 1995.

[5] C. K. Chui and G. Chen, Kalman Filtering with Real Time Applications, Springer Science+Business Media, 2008.

[6] D. H. Dini, C. Jahanchahi, and D. P. Mandic, "Kalman filtering for widely linear complex and quaternion valued bearings only tracking," IET Signal Processing, vol. 6, no. 5, pp. 435–445, 2012.

[7] N. Peach, "Bearings-only tracking using a set of range-parameterized extended Kalman filters," IEEE Proceedings: Control Theory and Applications, vol. 142, no. 1, pp. 73–80, 1995.

[8] E. Wan and R. van der Merwe, "The unscented Kalman filter," in Kalman Filtering and Neural Networks, pp. 221–280, John Wiley & Sons, 2001.

[9] Y. Bar-Shalom, X. Li, and T. Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, New York, NY, USA, 2001.

[10] F. Abdallah, A. Gning, and P. Bonnifait, "Box particle filtering for nonlinear state estimation using interval analysis," Automatica, vol. 44, no. 3, pp. 807–815, 2008.

[11] A. Gning, L. Mihaylova, and F. Abdallah, "Particle filtering combined with interval methods for tracking applications," in Integrated Tracking, Classification, and Sensor Management: Theory and Applications, Wiley, 2012.

[12] A. Gning, B. Ristic, L. Mihaylova, and F. Abdallah, "An introduction to box particle filtering," IEEE Signal Processing Magazine, vol. 30, no. 4, pp. 166–171, 2013.

[13] H. R. Hashemipour, S. Roy, and A. J. Laub, "Decentralized structures for parallel Kalman filtering," IEEE Transactions on Automatic Control, vol. 33, no. 1, pp. 88–94, 1988.

[14] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in Proceedings of the 46th IEEE Conference on Decision and Control, pp. 5492–5498, December 2007.

[15] F. S. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," IEEE Transactions on Automatic Control, vol. 55, no. 9, pp. 2069–2084, 2010.

[16] A. Mohammadi and A. Asif, "Distributed consensus + innovation particle filtering for bearing/range tracking with communication constraints," IEEE Transactions on Signal Processing, vol. 63, no. 3, pp. 620–635, 2015.

[17] O. Hlinka, F. Hlawatsch, and P. M. Djuric, "Consensus-based distributed particle filtering with distributed proposal adaptation," IEEE Transactions on Signal Processing, vol. 62, no. 12, pp. 3029–3041, 2014.

[18] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in Proceedings of the IEEE International Conference on Information and Automation (ICIA '08), pp. 302–307, June 2008.

[19] W. Zhou, W. Shi, X. Wang, and K. Wang, "Adaptive sensor activation algorithm for target tracking in wireless sensor networks," International Journal of Distributed Sensor Networks, vol. 2012, Article ID 515906, 10 pages, 2012.

[20] H. H. Chhadé, A. Gning, F. Abdallah, I. Mougharbel, and S. Julier, "Non parametric distributed inference in sensor networks using box particles messages," Mathematics in Computer Science, vol. 8, no. 3–4, pp. 455–478, 2014.

[21] V. D. Blondel, J. M. Hendrickxs, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC '05), pp. 2996–3000, Seville, Spain, December 2005.

[22] D. S. Scherber and H. C. Papadopoulos, "Locally constructed algorithms for distributed computations in ad-hoc networks," in Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '05), pp. 63–70, Los Angeles, Calif, USA, April 2005.

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

[24] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05), pp. 63–70, Los Angeles, Calif, USA, April 2005.

[25] L. Jaulin, M. Kieffer, and O. Didrit, Applied Interval Analysis, Springer, Berlin, Germany, 2001.

[26] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering: formulation and performance analysis," in Proceedings of the Cognitive Information Processing, pp. 36–41, 2008.

[27] S. M. Rump, "INTLAB—INTerval LABoratory," in Developments in Reliable Computing, T. Sendes, Ed., pp. 77–104, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.
Submit your manuscripts at
http://www.hindawi.com