Math.Comput.Sci. (2014) 8:455–478
DOI 10.1007/s11786-014-0200-2

Mathematics in Computer Science

Non Parametric Distributed Inference in Sensor Networks Using Box Particles Messages

Hiba Haj Chhadé · Amadou Gning · Fahed Abdallah · Imad Mougharbel · Simon Julier

Received: 14 December 2013 / Revised: 26 April 2014 / Accepted: 31 May 2014 / Published online: 7 August 2014

Abstract This paper deals with the problem of inference in distributed systems where the probability model is stored in a distributed fashion. Graphical models provide powerful tools for modeling this kind of problems. Inspired by the box particle filter which combines interval analysis with particle filtering to solve temporal inference problems, this paper introduces a belief propagation-like message-passing algorithm that uses bounded error methods to solve the inference problem defined on an arbitrary graphical model. We show the theoretic derivation of the novel algorithm and we test its performance on the problem of calibration in wireless sensor networks. That is the positioning of a number of randomly deployed sensors, according to some reference defined by a set of anchor nodes for which the positions are known a priori. The new algorithm, while achieving a better or similar performance, offers impressive reduction of the information circulating in the network and the needed computation times.

Keywords Interval analysis · Distributed systems · Graphical models · Bayesian inference · Belief propagation · Wireless sensor networks · Calibration

H. Haj Chhadé · F. Abdallah
Université de Technologie de Compiègne, Centre de Recherches de Royallieu, BP 20529,
Rue Personne de Roberval,
60205 Compiègne Cedex, France
e-mail: hiba.hajchhade@hds.utc.fr

F. Abdallah
e-mail: fahed.abdallah@hds.utc.fr

A. Gning (✉) · S. Julier
Department of Computer Science, University College London, Gower Street,
London WC1E 6BT, UK
e-mail: a.gning@cs.ucl.ac.uk

S. Julier
e-mail: s.julier@cs.ucl.ac.uk

I. Mougharbel
Université Libanaise, Hadath-Beyrouth, Lebanon
e-mail: imadmoug@yahoo.com

1 Introduction

Usually, intelligent systems have to reason constrained by noisy information acquired from their environment, i.e. they have to reason under uncertainty. Since noises are typically modeled as stochastic, probabilistic methods can be used for addressing the problem. In these approaches, variables of interest are modeled using a collection of random variables.

The main application of interest in this paper is to solve the problem of inference in a distributed system such as sensor networks. In sensor networks applications, each node (e.g. sensor) receives information about its local environment and has a local set of attributes (described in terms of random variables) for which it needs to compute a posterior density. Thus each node only stores a relevant portion of the model and has to collaborate with other nodes in the network in order to compute its marginal posterior of interest, given *all* the observations available to the system [22].

Graphical models are considered as a formalism to represent the way a joint distribution, defined over a set of random variables, may be expressed as a product of local factors where each factor depends on a subset of variables. They have important characteristics [5] making them of a wide applicability in statistics, statistic physics and machine learning, as well as in computer vision: they represent the structure of a probabilistic model; they provide a way to visualize the properties of the model (specifically conditional independence properties); visible graphical manipulations implicitly carry along complex computations required to perform inference.

The popular problem of auto-localization in wireless sensor networks is preeminently a distributed inference problem and can be formulated as a problem of inference on a graphical model [14]. In fact, many industrial, scientific and even domestic applications make use of sensor networks when there is a need to monitor, and possibly control, physical phenomena, such as brightness, temperature or pressure. Sensors acquire information from their environment and often communicate the data collected to a processing center. However, a vast majority of applications in sensor networks deploys a large number of sensors *randomly*, usually due to the hostility of the area to be monitored, or its immensity. The localization of the sensors is thus necessary to make the data collected informative.

All sensors cannot be equipped with a positioning module, e.g. GPS module, due to cost and energy constraints. Alternatively, each sensor is equipped with a transmitter–receiver module and communicates with neighboring sensors. Some approaches assume that sensors have capabilities to estimate distances with their neighbors, using technologies such as received signal strength indicator, time of arrival, time difference of arrival [14]. Nevertheless, approaches in [8,25] are based on measures of connectivity rather than distances. Other approaches are based on the fact that sensors are able to calculate angles with their neighbors, using technologies such as angle of arrival (AoA) [24]. The goal is to calculate the coordinates of each sensor based on proximity measures. The computed coordinates can be global, which requires the position of a number of anchors to be known a priori. These approaches are known as "anchor-based" [3]. The placement of the anchors can often have a significant impact on the solution. It was found that the location accuracy improves if the anchors form a convex polygon around the network [21]. Additional anchors placed at the center of the network are nonetheless useful. Other methods create a relative map without the use of anchors and are called "anchor-free" [1]. In all these cases, nodes must themselves determine their respective positions through cooperation techniques. It is, though, useful to note that for some algorithms, the aim is for each node to locate its neighbors *qualitatively*. [12] presents a localized algorithm whose purpose is, for each node, to classify its neighbors into one of three categories: very close, near and far.

Each sensor has limited resources (e.g. bandwidth, battery energy, memory capacity, emission power). It can detect and communicate with other nodes in the network only within some maximum span. Early approaches to solving the problem of localization in wireless sensor networks were proposed in a centralized environment. However, this strategy does not comply the principal energy constraint and is not appropriate for large-scale networks. Indeed, local data processing has low energy cost (see [7] for details about energetic cost of RF communication in function of distance). In addition, the reliability of the centralized approach is low because a failure in the main processing unit causes the entire system to fail. Therefrom distributed approaches [8,14] were proposed. A third approach is to assist the nodes of the network by a mobile anchor enabling them to locate themselves. This approach [3] has many advantages in terms of energy and location accuracy but its implementation can be expensive and is not always feasible as some areas can be hostile to moving robots.

One can also distinguish optimization methods [8,25] and probabilistic methods [23]. While optimization approaches provide a single optimal solution that minimizes or maximizes some criterion, the solution in probabilistic methods takes the form of a probability distribution. In this paper, we adopt a Bayesian probabilistic approach because it therefore allows to quantify the uncertainty on the estimated state. The work in [23] presents a probabilistic method for locating sensors based on AoA technology. The error on the measured angles is modeled using a Gaussian law. In [14], the localization problem is formulated as a problem of inference on a graphical model. The nonparametric belief propagation (NBP) algorithm is used to combine the information obtained from a global positioning system, with measures of relative distances between neighboring sensors. The algorithm of belief propagation (BP) is based on exchanging information iteratively between neighboring nodes in a graphical model. The NBP algorithm is itself a variant of BP, where a set of particles is used to represent probability quantities in a nonparametric way.

Furthermore, in some problems, uncertainties or noise characteristics are unknown or complex. Instead, only minimum and maximum values of the noise are available, e.g quantized measurements. The interval analysis framework offers promising methodologies for reasoning in the presence of unknown or complex statistical but bounded noises [10]. We propose a variant of BP algorithm where information is represented using a collection of boxes (intervals in the case of real variables). The use of this approach involve simplicity in modeling the information and memory optimization. In fact, the use of interval representations in our approach offered many advantages, basically:

1. a reduction of the memory space required to store a pdf (thousands of particles are needed to efficiently represent a pdf using Monte Carlo methods while only a few box-particles are required to approximate a probability distribution).
2. energy saving since less information is exchanged between communicating sensors, this also implies a reduction of the required bandwidth.
3. using interval techniques result in simpler and faster computations and thus more time saving.
4. a set of boxes constitute a direct approximation of the pdf while a set of samples constitute a "representation" of a pdf. This fact allows to save the energy and time needed to estimate the pdf using kernel density estimation (KDE) techniques in the case of NBP.

The rest of the article is organized in the following way. We begin in Sect. 2 by introducing the concept of graphical models. Section 3 presents belief propagation algorithm as a method for solving the problem of inference defined on a graphical model. However, computations required by BP become intractable in the case of complex graphical structures and in the presence of uncertainties or nonlinear observation processes. An approach, based on the use of sample-based representations for uncertainties, as in particle filtering, is described in Sect. 4. The so formulated algorithm is referred to as *NBP*. Section 5 presents a novel algorithm that generalizes message-passing information-exchanging BP algorithm into the bounded error context. Sections 6 and 7 respectively formulate the localization problem and report some simulation results while Sect. 8 concludes the paper.

2 Graphical Models

A probabilistic graphical model is a graph that combines graph theory with probability theory in order to represent conditional independence properties of a probability distribution [22]. Formally, a probabilistic graphical model is a particular graph G defined by $G = (V, E)$ where V is a set of nodes or vertices, and E is a set of edges. Each node $i \in V$ is associated with a random variable X_i and each edge (i, j) represents a probabilistic relation between the random variables X_i and X_j, respectively associated with the nodes $i, j \in V$. Two broad classes of graphical models exist: directed graphical models and undirected graphical models. Directed graphical models known as Bayesian networks (BNs) express causal relationships between random variables. Undirected graphical models, also called

Markov random fields (MRFs), encode constraints and correlations between variables [26]. A third class, referred to as *factor graphs*, generalizes the two classes of directed and undirected graphical models [20,29]. For the problem presented in this paper, MRFs type of graphical model is used and presented in detail in the next section.

2.1 Markov Random Fields

MRFs are a class of graphical models that uses an undirected graph to encode conditional independence relationships between random variables. They are popular in the fields of statistical physics and computer vision [4].

Consider an undirected graph G = (V, E) and let A, B and D refer to three disjoint subsets of V. If every path between set A and set B passes through some node in set D, D is said to separate A and B [29]. This *graph separation* criterion [22] implies conditional independence between X_A and X_B given X_D:

$$p(x_A, x_B|x_D) = p(x_A|x_D)p(x_B|x_D). \tag{2.1}$$

For a node $i \in V$, associated with a random variable X_i, let $\Gamma_i \subset V$ denote the set of neighbors of i. Γ_i comprises all the nodes of the graph connected to i by an edge. The set of variables associated with Γ_i is given by $X_{\Gamma_i} = \{X_j | j \in \Gamma_i\}$. The conditional independence constraints implied by the graph can be formulated straightforwardly as: any random variable X_i is conditionally independent, given X_{Γ_i}, of all other variables in the model:

$$p(x_i|x_{V\setminus i}) = p(x_i|x_{\Gamma_i}). \tag{2.2}$$

As for the parametrization of a MRF, we refer to the Hammersley and Clifford theorem (see [29] for its formulation). According to this theorem, the joint distribution can be parametrized by a product of factors defined over the cliques of the graph (where a clique is a set of fully connected nodes in a graph).

$$p(\mathbf{X}) = \frac{1}{Z}\prod_{c\in C} \psi_c(\mathbf{X}_c), \tag{2.3}$$

where C denotes the set of cliques in the graph, the factors $\psi_c(\mathbf{X}_c)$ denote some joint probability of the random variables $\mathbf{X}_c$ associated with the clique c, and are referred to as *potential functions*, and Z is a normalization constant. Obviously, the parametrization using the cliques is not unique. Figure 1 illustrates an example of a MRF. One possible parametrization of the graph is also given.

Pair-wise MRFs are a restricted class of the MRF family. In a pair-wise MRF, the cliques are exclusively pairs of nodes of the graph connected by edges. In typical scenarios, the set of nodes is partitioned into two disjoint sets

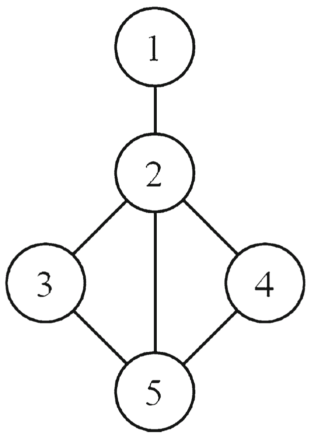

Fig. 1 Example of a Markov network. The joint distribution factors into the product $p(\mathbf{X}) = \frac{1}{Z}\psi_{235}(X_2, X_3, X_5)\psi_{245}(X_2, X_4, X_5)$ $\psi_{12}(X_1, X_2)$

V_x and V_y associated respectively with the set of hidden variables $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$ and the set of observable variables $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_M\}$ of the model, where $N = |V_x|$ and $M = |V_y|$. The set V_x is called the set of *hidden* or *latent* nodes. The nodes in set V_y are referred to as *evidence nodes*. Thereafter, the potential functions may be partitioned into two groups: the first group corresponds to the edges existing between the hidden variables and the observations. The second group corresponds to the edges interconnecting hidden variables. The first group of potential functions will be denoted $\psi_i(\mathbf{X}_i, \mathbf{Y})$ and the second set will be referred to as $\psi_{ij}(\mathbf{X}_i, \mathbf{X}_j)$. The joint probability distribution may then be put into the following form:

$$p(\mathbf{X}, \mathbf{Y}) = \frac{1}{Z} \prod_{(i,j)\in E, i\in V_x, j\in V_x} \psi_{ij}(\mathbf{X}_i, \mathbf{X}_j) \prod_{i\in V_x} \psi_i(\mathbf{X}_i, \mathbf{Y}). \tag{2.4}$$

2.2 Inference in Graphical Models

To define the problem of inference in graphical models, the typical scenario, as mentioned above, is to separate the set of nodes V into two disjoint sets $V = \{V_x, V_y\}$ associated with the hidden and the observable variables respectively. The inference problem generally refers to finding the distribution of all, or of a subset of, hidden variables given the observations [26]. This distribution denoted $p(\mathbf{x}|\mathbf{y})$ is called the posterior distribution. In this paper, it is assumed that the undirected graph modeling the distribution is a pair-wise MRF. It is also assumed that the set $\mathbf{y}$ of observations is actually a collection of observations, y_s, of individual node variables, x_s, each corrupted by a statistical noise that is independent of other components of $\mathbf{x}$ and $\mathbf{y}$. The observations y_s are said to be local to their associated hidden variables x_s [13]. The interpretation of this assumption is that $\psi_i(\mathbf{X}_i, \mathbf{Y}) = \psi_i(\mathbf{X}_i, \mathbf{Y}_i)$ (*local likelihood*). Referring to the subsection above, the posterior of interest can be expressed as follows:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \propto \prod_{(i,j)\in E} \psi_{ij}(\mathbf{x}_i, \mathbf{x}_j) \prod_{i} \psi_i(\mathbf{x}_i, \mathbf{y}_i). \tag{2.5}$$

Figure 2 illustrates the problem of inference on a graphical model. In this figure, the state vector is $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6, X_7)$ and the vector of local observations is $\mathbf{Y} = (Y_1, Y_3, Y_5)$.

3 Belief Propagation Algorithm

Belief propagation, also known as the sum-product algorithm, is a tool for solving the problem of inference on graphical models. It is a statistical estimation algorithm that applies specifically to BNs and MRFs. BP exploits the conditional independence relationships represented by the graphical models in order to calculate, in an exact or an approximated way, the posterior distribution at a node of the model. Although, the focus in our exposition is on pair-wise MRFs, which are a restricted class of *undirected* graphical models, the methods underlying BP as described in this section are also adapted to message-passing in Bayesian networks as well as in factor graphs [6,29].

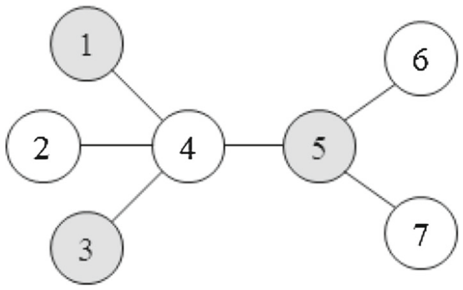

Fig. 2 The problem of inference: the *shaded nodes* represent the variables for which an observation is available

The inference goal is to compute the posterior *marginal* distribution associated with a node t given the set of observations **Y**:

$$p(x_t|\mathbf{y}) = \int_{x\backslash x_t} p(\mathbf{x}|\mathbf{y})dx. \tag{3.1}$$

This marginalization task is captured by belief propagation in an iterative fashion. Noting that

$$p(\mathbf{x}|\mathbf{y}) \propto \prod_{(t,s)\in E} \psi_{ts}(\mathbf{x}_t, \mathbf{x}_s) \prod_t \psi_t(\mathbf{x}_t, \mathbf{y}_t),$$

in the most common form of BP, at each iteration, each node calculates outgoing messages to all of its neighbors simultaneously. The outgoing message from a node t to one of its neighbors s at an iteration i of BP, is updated in terms of the messages incoming to t, at iteration $i-1$, from the set of t's neighbors (noted Γ_t) except for that incoming from s itself (see Fig. 3 for an illustration of BP's message update operation). The message outgoing from t to s at iteration i is denoted $m^i_{ts}(x_s)$. For the simplicity of the notation we will refer to the local likelihood $\psi_t(x_t, y_t)$ as $\psi_t(x_t)$.

$$m^i_{ts}(x_s) \propto \int \psi_{ts}(x_t, x_s)\psi_t(x_t) \prod_{u\in\Gamma_t\backslash s} m^{i-1}_{ut}(x_t)dx_t. \tag{3.2}$$

Throughout this article, $R^{i-1}_{ts}(x_t)$ denotes the message product:

$$R^{i-1}_{ts}(x_t) \propto \prod_{u\in\Gamma_t\backslash s} m^{i-1}_{ut}(x_t), \tag{3.3}$$

while

$$M^i_{ts}(x_t) \propto \psi_t(x_t) \prod_{u\in\Gamma_t\backslash s} m^i_{ut}(x_t) \tag{3.4}$$

is referred to as partial belief. It combines all the available information about t, from t itself and t's neighboring nodes (except for the destination node s).

Next to message-passing, the belief at each node t, defined at iteration i, is computed using the following expression:

$$q^i_t(x_t) \propto \psi_t(x_t) \prod_{u\in\Gamma_t} m^i_{ut}(x_t). \tag{3.5}$$

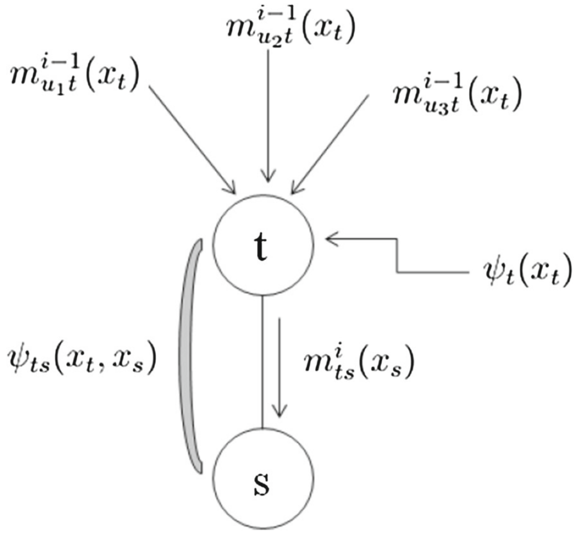

Fig. 3 Message update

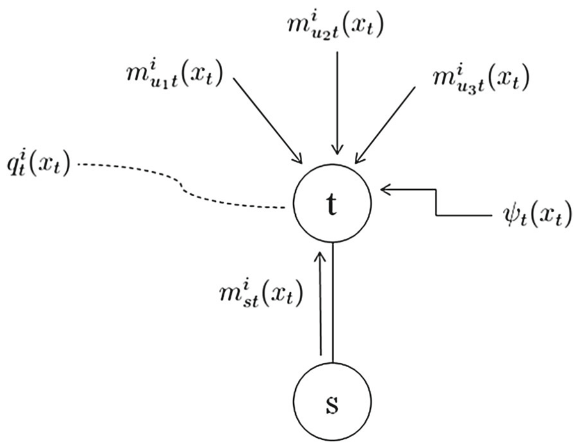

Fig. 4 Belief computation

Figure 4 illustrates BP's belief computation. Beliefs are normalized in order to integrate to unity [13]. The interpretation of the belief at a node t is that it approximates the posterior marginal of interest $p(x_t|\mathbf{y})$. In fact, under some conditions the belief converges to the exact posterior marginal distribution.

If the graphical model is tree-structured (i.e. is loop free), belief Propagation is guaranteed to converge after a finite number of iterations (at most equal to the length of the longest path in the graph [13]). In this case, the belief at a node t will be exactly equal to the posterior marginal of interest $p(x_t|\mathbf{y})$. Nevertheless, BP may also be applied to arbitrary graphical models. The same local belief propagation equations are iterated overlooking the presence of loops in the graph. This iterative procedure is then referred to as *loopy belief propagation*. In this case, the convergence of the sequence of messages is not guaranteed. Under some conditions however, fixed points will appear and in practice, these fixed points (beliefs) constitute reasonable approximations of the exact posterior marginal distribution associated with the node. Loopy belief propagation is explained in details in [6]. Furthermore, it is worth mentioning that algorithms for exact marginalization on arbitrary graphical models do exist. One popular algorithm known as JLO (after its authors F.V. Jensen, S.L. Lauritzen and K.G. Olesen) is based on junction tree representation. This algorithm starts by grouping nodes into cliques to break the original graph's cycles and is well elaborated in [22].

4 Nonparametric Belief Propagation

Sequential Bayesian filtering consists in estimating the states (at each time step) of a system as a set of observations become available [30]. While standard Bayesian solutions such as the Kalman filter [31] and its variants [19] make an assumption on a known posterior distribution form (e.g. Gaussian) to simplify this recursive Bayesian estimation, sequential Monte Carlo [18] techniques also known as particle filtering make no assumptions on the form of the probability densities of interest. Particle filtering [2] uses sample-based representations in order to construct Monte Carlo approximations of the required integrals in the case of seriously non-linear and complex posterior distributions.

Recall that belief propagation is a tool for performing exact or approximate marginalization on arbitrary graphical models. This algorithm relies on two operations which can represent major complexity when performed for general case potential functions/messages. The first operation is a product of a collection of messages, the second is a convolution operation of this message product with a pairwise potential function [see Eq. (3.2)]. For this reason, BP can be practically implemented only in special cases when these two operations remain tractable, namely in the case of discrete-valued random variables (matrices computation) and in the case of Gaussian distributions [13]. In many applications of graphical models, the hidden variables of interest are described by continuous non-Gaussian distributions. That is especially the case of graphical models applications in computer vision [28]. Furthermore, the model (the observation model or the functions describing the propagation of information between nodes) may be governed by serious non-linearities. NBP is an attractive solution in the presence of such complex situations.

NBP algorithm is considered, on one hand, as a generalization of particle filtering applied to arbitrary graphical models and on the other hand as a stochastic approximation of BP [13,28].

The basic idea behind NBP is to use sample-based representations to approximate the operations of BP. As stated above, the message update task is divided into two operations [28]. The first operation is a message product operation through which the partial belief is computed according to Eq. (3.4). The second operation is a convolution operation. As shown in Eq. (3.2), the available information about t is propagated through the pair-wise potential function $\psi_{ts}(x_t, x_s)$ in order to form a message providing some information about the receiving node's local state. In NBP, messages $m^i_{ut}(x_t)$ are represented by a collection $\{\omega^j_{ut}, x^j_{ut}\}$, $j = 1, \ldots, N$, of weighted samples. An estimation of $m^i_{ut}(x_t)$ is hence given by

$$\hat{m}^i_{ut}(x_t) = \sum_{j=1}^{N} \omega^j_{ut} \delta \left(x_t - x^j_{ut} \right). \tag{4.1}$$

To perform the product operation, this form of the message is smoothed using nonparametric density estimation methods [27]. The idea is to smooth the effect of each sample onto a nearby region using a kernel, e.g. a Gaussian kernel noted K_h. The message estimate is now given by a mixture of N Gaussian distributions:

$$m^i_{ut}(x_t) = \sum_{j=1}^{N} \omega^j_{ut} K_h \left(x_t - x^j_{ut} \right). \tag{4.2}$$

Performing the product of d messages, each represented by a mixture of N Gaussian distributions, the result is a Gaussian mixture containing N^d components. To avoid complex computations, the author in [13] proposes to sample exactly N particles from this product of Gaussian mixtures using a Markov chain Monte Carlo method, namely the Gibbs sampler.

Having a collection $\{\Omega^j_{ts}, X^j_{ts}\}$ of weighted particles representing the partial belief $M^i_{ts}(x_t)$, the next step is to approximate the integral given in Eq. (3.2). Recalling Monte Carlo methods, if N independent samples $x^i_p \sim p(x)$, $i = 1, \ldots, N$ can be drawn from a probability density function $p(x)$, the expectation of any function f of x under the distribution p may be approximated empirically as follows:

$$\mathbb{E}_p(f(x)) = \int f(x)p(x)dx \approx \frac{1}{N} \sum_{i=1}^{N} f\left(x^i_p \right). \tag{4.3}$$

A direct application of Monte Carlo approximation may be adopted here. However, the pair-wise potential function $\psi_{ts}(x_t, x_s)$ might have influence on x_t. In other words, the marginal

$$\zeta_{ts}(x_t) = \int \psi_{ts}(x_t, x_s)dx_s \tag{4.4}$$

is not always equal to 1. NBP algorithm accounts for this marginal influence by incorporating $\zeta_{ts}(x_t)$ into the message product operation [13].

Figure 5 summarizes NBP operations. NBP can provide a solution to the problem of inference in arbitrary graphical models containing high-dimensional variables with continuous non Gaussian distributions or presenting severe non-linearities [28]. Each iteration of NBP uses a sampling procedure to update kernel-based estimates of the true messages. On one hand, the storage of a sufficiently large number of weighted particles, and on the other hand, the overhead assigned to computing KDEs and to sampling from the product of Gaussian mixtures, increase the complexity associated with NBP's computations. This difficulty was the motivation behind developing the novel scheme of message-passing presented in the next section.

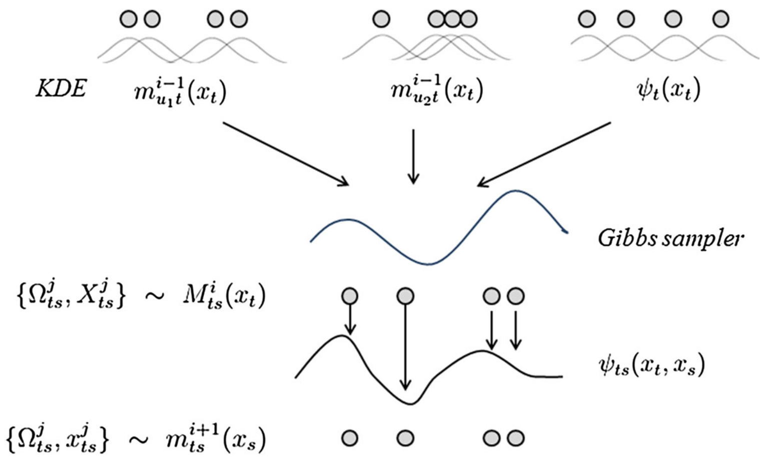
KDE
$m^{i-1}_{u_1 t}(x_t)$
$m^{i-1}_{u_2 t}(x_t)$
$\psi_t(x_t)$
Gibbs sampler
$\{\Omega^j_{ts}, X^j_{ts}\} \sim M^i_{ts}(x_t)$
$\psi_{ts}(x_t, x_s)$
$\{\Omega^j_{ts}, x^j_{ts}\} \sim m^{i+1}_{ts}(x_s)$

Fig. 5 Nonparametric belief propagation

5 Belief Propagation Combined with Interval Analysis

This section presents an algorithm that uses bounded error methods for solving the inference problem in an arbitrary graphical model. In fact, the use of interval representations offers many advantages, basically:

1. a reduction of the memory space required to store the messages and beliefs.
2. energy saving since the exchanged information between communicating nodes is much less bulky (this also implies a reduction of the required bandwidth).
3. using interval-based representations avoids the overhead associated with the computation of nonparametric density estimates and the procedure of sampling from the product of Gaussian mixtures in NBP. This translates into simpler and faster computations and thus more time saving (see Sect. 7).
4. avoiding the need to introduce artificial parameters in order to approximate the true messages, since a set of boxes constitute a direct approximation of the pdf, and getting around the convergence problem of MCMC samplers.

5.1 Interval Analysis

An interval in $\mathbb{R}$ is a closed and connected subset of $\mathbb{R}$ defined as

$$[x] = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}, \tag{5.1}$$

where $\underline{x}$ and $\overline{x}$ refer, respectively, to the minimal and maximal bounds of $[x]$.

A *box* on the other side represents a vector $[\mathbf{x}]$ in $\mathbb{R}^n$ and is defined as a Cartesian product of n intervals:

$$[\mathbf{x}] = [x_1] \times \cdots \times [x_n]. \tag{5.2}$$

Henceforth, $|[x]|$ will denote the length of the interval $[x]$, $\mathbb{IR}$ will refer to the set of intervals in $\mathbb{R}$, and $\mathbb{IR}^n$ the set of boxes in $\mathbb{R}^n$.

Operations on Intervals and Boxes.

Firstly, set-theoretic operations (such as intersection and union) are applicable to intervals. Note that the intersection of two intervals is always an interval, whereas their union is not necessarily an interval. The *interval union* of two intervals $[x]$ and $[y]$ is defined by the following expression:

$$[x] \sqcup [y] = [[x] \cup [y]]\,, \tag{5.3}$$

where interval union operation is denoted by $\sqcup$ whereas $\cup$ refers to the set-theoretic union operation. The symbol [.] denotes the *interval hull* operator returning, for any set S in $\mathbb{R}$, the smallest interval enclosing S [17].

Secondly, the binary operations $\{+, -, \times, /\}$ can be extended to intervals [17]. Let the symbol $\diamond$ refer to any binary operation and $[x]$ and $[y]$ denote any two intervals in $\mathbb{R}$, the resulting interval $[z] = [x] \diamond [y]$ is defined as

$$[z] = [x] \diamond [y] = [\{x \diamond y \mid x \in [x], y \in [y]\}]. \tag{5.4}$$

If the binary operation $\diamond$ is continuous, as in the case of the usual arithmetic operations, the set $\{x \diamond y \in \mathbb{R} | x \in [x], y \in [y]\}$ is an interval. Thus,

$$[x] \diamond [y] = \{x \diamond y \in \mathbb{R} | x \in [x], y \in [y]\}.$$

The extension to intervals of the usual arithmetic operations $\{+, -, \times, /\}$ is given next. Recall that $[x] = [\underline{x}, \overline{x}]$ and $[y] = [\underline{y}, \overline{y}]$.

$$\begin{aligned}
[x] + [y] &= \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right], \\
[x] - [y] &= \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right], \\
[x] \times [y] &= \left[\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})\right].
\end{aligned} \tag{5.5}$$

If the interval $[y]$ does not include the 0 value, one can also define

$$[x]/[y] = [x] \times [1/\overline{y}, 1/\underline{y}].$$

Similarly, elementary functions such as exp, ln, cos and sin, can be simply extended to intervals.

Furthermore, all operations on intervals can be extended to boxes.

Inclusion Functions.

Consider a function $\mathbf{f}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$. We are interested in computing the image $\mathbf{f}([\mathbf{x}])$ of a box $[\mathbf{x}]$ by $\mathbf{f}$. This image is often not a box (see Fig. 6) and its expression might be difficult to obtain. An *inclusion function* approximates $\mathbf{f}([\mathbf{x}])$. Let $[\mathbf{f}]$ denote an interval function from $\mathbb{IR}^n$ to $\mathbb{IR}^m$. By definition, $[\mathbf{f}]$ is said to be an *inclusion function* for $\mathbf{f}$ if

$$\mathbf{f}([\mathbf{x}]) \subseteq [\mathbf{f}]([\mathbf{x}]), \quad \forall [\mathbf{x}] \in \mathbb{IR}^n. \tag{5.6}$$

Inclusion functions may be very pessimistic [16]. An inclusion function $[\mathbf{f}]$ is *minimal* if, for any $\mathbf{x}$, $[\mathbf{f}]([\mathbf{x}])$ is the interval hull of $\mathbf{f}([\mathbf{x}])$. The minimal inclusion function for $\mathbf{f}$ is unique and will be denoted by $[\mathbf{f}]^*$. Refer to Fig. 6 for an example.

Amongst the main purposes of interval analysis [17] are: firstly, finding an inclusion function $[\mathbf{f}]$ such that, for most $\mathbf{x}$, $[\mathbf{f}]([\mathbf{x}])$ is close to $[\mathbf{f}]^*([\mathbf{x}])$; and secondly, finding it with a convenient computational time.

Constraints Satisfaction Problem and Contraction.

Another popular topic in interval analysis are the constraints satisfaction problems (CSPs). Let $\mathbf{x}$ be a vector of n variables $x_i \in \mathbb{R}, \quad i \in \{1, \ldots, n\}$, i.e. $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. $\mathbf{f} = (f_1, f_2, \ldots, f_m)^T$ is a multivalued function such as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{5.7}$$

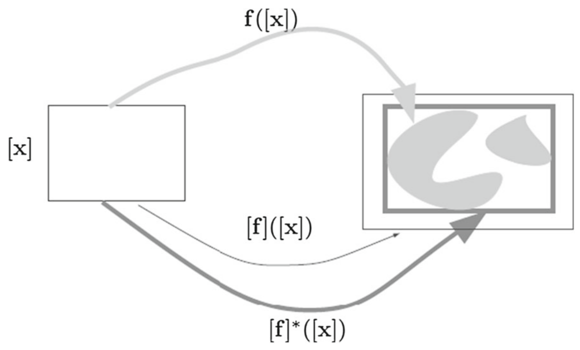

Fig. 6 Inclusion function [10]

The components $f_j, \quad j \in \{1, \ldots, n\}$ of $\mathbf{f}$ represent constraints linking the variables x_i. The vector $\mathbf{x}$ belongs to a given prior domain $[\mathbf{x}]$ of $\mathbb{R}^n$. The purpose is to find the *smallest* box enclosing the set of all $\mathbf{x}$ in the prior domain $[\mathbf{x}]$ which satisfy the constraints $\mathbf{f}$. A CSP is commonly denoted as $\mathcal{H}$ and may be formulated as follows:

$$\mathcal{H} : (\mathbf{f}(\mathbf{x}) = \mathbf{0}, \mathbf{x} \in [\mathbf{x}]). \tag{5.8}$$

The solution set of the CSP $\mathcal{H}$ is given by

$$\mathbf{S} = \{\mathbf{x} \in [\mathbf{x}] \mid \mathbf{f}(\mathbf{x}) = \mathbf{0}\} \tag{5.9}$$

and is not necessarily a box. Hence the domain $\mathcal{H}$ is *contracted* [17]. The term *contracting* refers to replacing $[\mathbf{x}]$ by a *smaller* domain $[\mathbf{x}]'$ such that $\mathbf{S} \subseteq [\mathbf{x}]' \subseteq [\mathbf{x}]$. A well known contraction method is that of *constraints propagation* (CP). This contraction technique is simple, efficient and most importantly independent of nonlinearities [10].

The CP method is based on the use of *primitive constraints*. A constraint is said to be primitive if it involves a single binary operation (such as $+, -, \times, /$) or a single elementary function (such as sin, cos, ln, exp). Constraint Propagation technique proceeds by contracting $\mathcal{H}$ with respect to each primitive constraint until convergence to a minimal domain. This method also referred to as FBP [11] consists of two steps: the forward propagation and the backward propagation. The first step considers the direct forms of the equations. The second uses the inverse of the functions that appear in the equations. Illustrative examples of CP method can be found in [10].

5.2 The Box Particle Filter

The directed graph in Fig. 7 represents the problem of filtering. In this figure, $\mathbf{x}_k$ denotes the state vector at time step k while $\mathbf{y}_k$ refers to the observation available at time k. In many applications, the posterior probability distribution $p(\mathbf{x}_k|\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{y}_1)$ provides sufficient information about the system's state. Filtering problem can thus be seen as a particular inference problem defined on a temporal Markov chain [13]. As stated in Sect. 4, particle filtering (PF) is a Monte Carlo based method for sequentially estimating the posterior marginal distributions $p(\mathbf{x}_k|\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{y}_1)$ [18] and NBP can be considered as a generalization of PF to an arbitrary graphical model [13]. The aim of the Box-PF is to generalize particle filtering into the bounded error context [10]. In the following description of the box-PF, we consider the following model:

$$\begin{cases} x_{k+1} = f(x_k, v_{k+1}), \\ y_{k+1} = g(x_{k+1}, w_{k+1}), \end{cases} \tag{5.10}$$

where f is a nonlinear transition function, g is a function that defines the relation between the state and the measurement vectors, v and w denote noise sequences. Four steps describe the Box-PF: box particle initialization, time update, measurement update and resampling.

- *Box Particle Initialization*. In this stage a prior bounded state space region is split into N equally weighted and mutually disjoint boxes $\{[\mathbf{x}_0^{(\ell)}]\}_{\ell=1}^{N}$. This initialization using boxes allows to explore a large prior uncertainty region using only a few box particles.

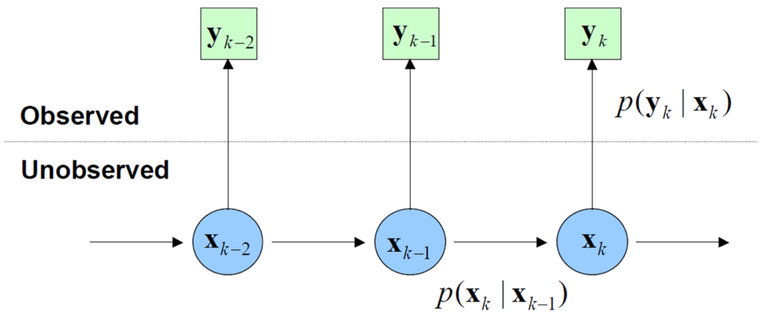

Fig. 7 Filtering problem [30]

- *Time Update Step*. Knowing the cloud of box particles $\{[\mathbf{x}_k^{(\ell)}]\}_{\ell=1}^N$ representing the state at time step k and assuming that the system noise is enclosed in $[\mathbf{v}_{k+1}]$, the boxes at step k are propagated using interval analysis tools through the transition function

  $$\left[\mathbf{x}_{k+1}^{(\ell)}\right] = [\mathbf{f}]\left(\left[\mathbf{x}_k^{(\ell)}\right], [\mathbf{v}_{k+1}]\right) \quad for\ \ell = 1, \dots, N,$$

  where $[\mathbf{f}]$ is an inclusion function for the transition function $\mathbf{f}$.
- *Measurement Update Step*. Similarly to particle filtering, the weights of the predicted box particles are updated using the new measurement at time step $k+1$. For this purpose, likelihood factors are calculated using *innovation quantities* [10]. The innovation for the ℓ-th box particle reflects the proximity between the measured box and the predicted box measurement. Hence, the innovation can be represented using the *intersection* between these two boxes. For each box particle, i.e. for $\ell = 1, \dots, N$, the predicted box measurement has the following expression:

  $$\left[\mathbf{y}_{k+1}^{(\ell)}\right] = [\mathbf{g}]\left(\left[\mathbf{x}_{k+1}^{(\ell)}\right]\right),$$

  where $[\mathbf{g}]$ is an inclusion function for $\mathbf{g}$. The innovation is given by:

  $$\left[\mathbf{r}_{k+1}^{(\ell)}\right] = \left[\mathbf{y}_{k+1}^{(\ell)}\right] \cap [\mathbf{y}_{k+1}].$$

  In the bounded error context, the likelihood is calculated based on the following idea: if the predicted box measurement does not intersect with the corresponding measured box, this box particle has a likelihood factor equal to zero. In contrast, if the predicted box measurement is included in the corresponding measured box, this box particle has a likelihood close to one [10]. Furthermore, a *contraction step* is performed in order to eliminate the inconsistent part of the box particles with respect to the measured boxes, and to preserve an appropriate size of the boxes. The *box likelihood* is thus given by:

  $$L_k^{(\ell)} = \prod_{j=1}^{n_x} L_k^{(\ell),j},$$

  where n_x represents the dimension of the state and the likelihood factor according to a dimension j of x is given by

  $$L_k^{(\ell),j} = \frac{\left|\left[\tilde{\mathbf{x}}_{k+1}^{(\ell)}(j)\right]\right|}{\left|\left[\mathbf{x}_{k+1}^{(\ell)}(j)\right]\right|}.$$

  The term $[\tilde{\mathbf{x}}_{k+1}^{(\ell)}(j)]$ represents the new ℓ-th box particle after the contraction step.
- *Resampling Step*. Similarly to the PF algorithm, a resampling step is also added in order to introduce variety into the box particles. Different resampling algorithms exist. However, in the Box-PF algorithm reported in [10], the multinomial resampling is applied, combined with a new subdivision step; this means that, after resampling, each box is divided by the corresponding number of realizations in order to obtain smaller boxes around the regions with high likelihoods.

Note that the Bayesian justification for the steps described here for the box-PF was established by interpreting each box as a uniform pdf [10].

5.3 Belief Propagation in the Bounded Error Context

This section presents the main theoretical contribution of this paper. A message passing algorithm is used to infer on graphical model when posterior probabilities are represented using boxes. Inspired by the theoretic derivation of the Box-PF presented in [9,10], we show the theoretic derivation of BP in the bounded error context by interpreting a box as a uniform pdf. Note that an advantage offered by this interpretation is that while, strictly speaking, a set of samples constitute a "representation" of a pdf, a set of boxes constitute a direct approximation of the pdf. Going back to Eq. (3.2) representing BP's message update, recall that the first operation is the messages product:

$$R_{ts}^{i-1}(x_t) \propto \prod_{u \in \Gamma_t \backslash s} m_{ut}^{i-1}(x_t) = \prod_{l=1}^{d} m_{u_l t}^{i-1}(x_t).$$

Γ_t denotes the set of neighbors of t, $u_l \in \Gamma_t \backslash s$ and $d = card(\Gamma_t \backslash s)$. In our method, the messages are represented using N weighted boxes. Thus, a message $m_{u_l t}(x_t)$ received by node t from its neighbor u_l, $l = 1, \ldots, d$, is represented by the collection $\{\omega_{u_l t}^{p_l}, [x_{u_l t}^{p_l}]\}_{p_l=1}^{N}$. Note that, for simplicity, in this representation we skip the index i (referring to the message passing iteration number) on the particles and their weights. Since each box is interpreted as a uniform probability distribution, then:

$$m_{u_l t}^{i-1}(x_t) = \sum_{p_l=1}^{N} \omega_{u_l t}^{p_l} U_{[x_{u_l t}^{p_l}]}(x_t), \quad for\ l = 1, \ldots, d,$$

where $U_{[x]}$ denotes the uniform pdf over the box $[x]$. Let P denote the vector of indexes $(p_0, p_1, \ldots, p_d)$. Replacing the expression above in that of the message product we obtain

$$\begin{aligned} R_{ts}^{i-1}(x_t) &\propto \prod_{l=1}^{d} \left(\sum_{p_l=1}^{N} \omega_{u_l t}^{p_l} U_{\left[x_{u_l t}^{p_l}\right]}(x_t) \right) \\ &\propto \sum_{P \in I^d} \omega_{u_1 t}^{p_1} \ldots \omega_{u_d t}^{p_d} U_{\left[x_{u_1 t}^{p_1}\right]} \ldots U_{\left[x_{u_d t}^{p_d}\right]}(x_t), \end{aligned} \tag{5.11}$$

where $I = \{1, 2, \ldots, N\}$. Recall that a uniform pdf $U_{[x]}$ is actually constant (and equal to $1/|[x]|$) over its support and equal to *zero* everywhere else. The product $U_{[x_{u_1 t}^{p_1}]} \ldots U_{[x_{u_d t}^{p_d}]}(x_t)$ is then different to *zero* if x_t belongs to the intersection of the supports of its terms. This product may hence be modeled using a uniform pdf given by

$$U_{\left[x_{u_1 t}^{p_1}\right]} \ldots U_{\left[x_{u_d t}^{p_d}\right]}(x_t) \ = \ U_{\left[x_{u_1 t}^{p_1}\right] \cap \cdots \cap \left[x_{u_d t}^{p_d}\right]} \times \frac{\left|\left[x_{u_1 t}^{p_1}\right] \cap \cdots \cap \left[x_{u_d t}^{p_d}\right]\right|}{\left|\left[x_{u_1 t}^{p_1}\right]\right| \ldots \left|\left[x_{u_d t}^{p_d}\right]\right|}. \tag{5.12}$$

Then

$$R_{ts}^{i-1}(x_t) \ \propto \ \sum_{P \in I^d} \omega_{u_1 t}^{p_1} \ldots \omega_{u_d t}^{p_d} U_{\left[x_{u_1 t}^{p_1}\right] \cap \cdots \cap \left[x_{u_d t}^{p_d}\right]} \times \frac{\left|\left[x_{u_1 t}^{p_1}\right] \cap \cdots \cap \left[x_{u_d t}^{p_d}\right]\right|}{\left|\left[x_{u_1 t}^{p_1}\right]\right| \ldots \left|\left[x_{u_d t}^{p_d}\right]\right|}. \tag{5.13}$$

The number of possible assignments of P is N^d. Thus, the sum above contains at most N^d terms. In practice, this number is much less given that some combinations result in an empty intersection while others result in coincident boxes. Note that for simplicity, we are considering all messages represented using the same number of boxes, N. This reasoning may however be easily extended to a more general case with varying number of box particles per node/message.

Let us denote by Q the set of assignments of P, for which $[x_{u_1 t}^{p_1}] \cap \cdots \cap [x_{u_d t}^{p_d}] \neq \emptyset$ and let $Z = card(Q)$. We will also adopt the following notations for an assignment $P^k \in Q$, where $k = 1, \ldots, Z$:

$$\omega_{u_1 t}^{p_1} \ldots \omega_{u_d t}^{p_d} \times \frac{\left|\left[x_{u_1 t}^{p_1}\right] \cap \cdots \cap \left[x_{u_d t}^{p_d}\right]\right|}{\left|\left[x_{u_1 t}^{p_1}\right]\right| \ldots \left|\left[x_{u_d t}^{p_d}\right]\right|} = \omega_{ts}^{k},$$

$$\left[x_{u_1 t}^{p_1}\right] \cap \cdots \cap \left[x_{u_d t}^{p_d}\right] = \left[x_{ts}^{k}\right].$$

The weights ω_{ts}^{k} shall be normalized.

Based on Eq. (5.13) we can define an algorithm to perform the messages combination task.

Algorithm 1 describes the method for combining *two* messages according to (5.13); note that both operations, $(\times)$ and $(\cap)$ are associative operations. In this algorithm, each message is represented using a certain number of weighted boxes. M_1 (M_2) denote the array of boxes representing message 1 (message 2), W_1 (W_2) represents the corresponding weights. X_{ts} and W_{ts} denote respectively the resulting array of boxes and the corresponding weights.

Algorithm 1. Messages combination.

```
1. Initialize $X_{ts}$ and $W_{ts}$ to empty arrays, set k=0.
2. Set $L_1$ equal to the number of boxes representing message 1, and $L_2$ equal to the number of boxes representing message 2.
3. (Nested loops)
   $i = 1, \dots, L_1$
   $j = 1, \dots, L_2$
   if $(M_1(i) \cap M_2(j) \neq \emptyset)$
     • $k = k + 1$,
     • $X_{ts}(k) = M_1(i) \cap M_2(j)$,
     • $W_{ts}(k) = W_1(i) \times W_2(j) \times \frac{|M_1(i) \cap M_2(j)|}{|M_1(i)|.|M_2(j)|}$.
4. Check for the existence of coincident boxes in $X_{ts}$, keep one occurrence and sum up the corresponding weights.
5. Normalize the weights:
   $W_{ts}(k) \longleftarrow W_{ts}(k)/sum(W_{ts})$.
```

The partial belief is now given by the following expression:

$$M_{ts}^{i-1}(x_t) \propto \psi_t(x_t) \sum_{k=1}^{Z} \omega_{ts}^k U_{[x_{ts}^k]}(x_t), \tag{5.14}$$

where $\psi_t(x_t)$ is actually $\psi_t(x_t, y_t)$ [see Eq. (2.5)]. In the general case, the potential function $\psi_t(x_t, y_t)$ can be represented as a function g linking the local observation y_t at node t to the local state x_t (observation model), $y_t = g(x_t, v_g)$ where v_g is a *bounded* measurement noise. Using contraction techniques (see Sect. 5.1), the resulting weighted boxes $\{\omega_{ts}^k, [x_{ts}^k]\}_{k=1}^Z$ can be contracted and re-weighted [15]. *Algorithm 2* summarizes the procedure of contracting the boxes $\{\omega_{ts}^k, [x_{ts}^k]\}_{k=1}^Z$ using the measurement $[y_t]$. In this procedure $[g]$ denotes an inclusion function for g.

Algorithm 2. Contraction using the local likelihood.

```
input:
$\{w_{ts}^k, [x_{ts}^k]\}_{k=1}^Z$: set of boxes and corresponding set of weights,
$[y_t]$: observation at node $t$.

1. Predicted measurement:
   $[y^k] = [g]([x_{ts}^k], [v_g]), \; for \; k = 1, \dots Z$.
2. Innovation: $[r^k] = [y^k] \cap [y_t], \; for \; k = 1, \dots Z$.
3. Box particle contraction: if $[r^k] \neq \emptyset$, then contract $[x_{ts}^k]$ using $[r^k]$ and CP algorithm to obtain $[x_{ts}^k]_{new}$, else
   $[x_{ts}^k]_{new} = \emptyset$, for $k = 1, \dots Z$.
4. Re-weighting: $w_{ts}^k \longleftarrow w_{ts}^k \times \frac{|[x_{ts}^k]_{new}|}{|[x_{ts}^k]|}$.
5. Weights normalization:
   $w_{ts}^k \longleftarrow w_{ts}^k/sum(w_{ts})$.
```

Note that the belief at node t as given in Eq. (3.5) can be computed in the same manner.

The next step in the message update iteration is the convolution operation of the partial belief with the pair-wise potential function:

$$\begin{aligned} m_{ts}^{i}(x_s) &= \int \psi_{ts}(x_t, x_s) M_{ts}^{i-1}(x_t) dx_t \\ &= \int \psi_{ts}(x_t, x_s) \sum_{k=1}^{V} \omega_{ts}^{k} U_{[x_{ts}^{k}]}(x_t) \\ &= \sum_{k=1}^{V} \omega_{ts}^{k} \int\limits_{[x_{ts}^{k}]} \psi_{ts}(x_t, x_s) U_{[x_{ts}^{k}]} dx_t \\ &= \sum_{k=1}^{V} \omega_{ts}^{k} \frac{1}{|[x_{ts}^{k}]|} \int\limits_{[x_{ts}^{k}]} \psi_{ts}(x_t, x_s) dx_t. \end{aligned} \tag{5.15}$$

To develop further Eq. (5.15) consider general practical case where the potential $\psi_{ts}(x_t, x_s)$ can be represented as a transition function by which we can pass from x_t to x_s, i.e. $x_s = f(x_t, \mathbf{e}, v_f)$ where v_f denotes a noise and $\mathbf{e}$ refers to a vector of constants (e.g. some known parameters of the model), and let $[f]$ be an inclusion function for f. We assume that the noise v_f is bounded in the box $[v_f]$.

Then, by definition of an inclusion function, we have

$$\forall x_t \in \left[x_{ts}^{k}\right], \quad x_s \in [f]\left(\left[x_{ts}^{k}\right], [v_f], [\mathbf{e}]\right) \quad for\ k \in \{1, \ldots, Z\},$$

i.e.

$$\psi_{ts}(x_t, x_s) U_{[x_{ts}^{k}]}(x_t) = 0, \quad \forall x_s \notin [f]\left(\left[x_{ts}^{k}\right], [v_f]\right). \tag{5.16}$$

Equation (5.16) shows that for any transition function f, using interval analysis techniques, the *support* for the pdf terms $\int_{[x_{ts}^{k}]} \psi_{ts}(x_t, x_s) U_{[x_{ts}^{k}]} dx_t$ can be *approximated* by $[f]([x_{ts}^{k}], [v_f], [\mathbf{e}])$ and thus:

$$\int\limits_{[x_{ts}^{k}]} \psi_{ts}(x_t, x_s) U_{[x_{ts}^{k}]} dx_t \ \approx \ U_{[f]([x_{ts}^{k}], [v_f], [\mathbf{e}])}. \tag{5.17}$$

Note that, it is shown in [10] that approximation (5.17) can be done more precisely at a computation cost using a mixture of uniform boxes e.g. more than one box particle. For simplicity and without loss of generality only one box is used in this paper. Based on Eqs. (5.15) and (5.17), we can describe the message update procedure, for a message sent from node t to its neighbor s, as follows: Once the incoming messages to the sending node t are combined as depicted in *Algorithm 1* and contracted using the local potential function at t, the resulting boxes $\{X_{ts}(k)\}_{k=1}^{Z}$, combining all information about x_t, are propagated through the model $x_s = f(x_t, v, \mathbf{e})$ to obtain an information about x_s. The weights must also be corrected (divided by $|[X_{ts}(k)]|$) and then normalized. This procedure is summarized in *Algorithm 3*.

Algorithm 3. Message update.

```
1. Set $MsgCounter = 0$.
2. for ($u \in \Gamma_t$ and $u \neq s$)
   if a message is incoming from u to t, increase $MsgCounter$.
3. Messages product
   • if ($MsgCounter = 0$) and an observation is available for t,
     use it to generate uniformly $N$ boxes mutually disjoint and weighted equally with $\frac{1}{N}$.
   • if ($MsgCounter = 0$) and no observation is available for t,
     no message can be forwarded from t to s.
   • if ($MsgCounter \neq 0$), use Algorithm 1 and the associativity of ($\cap$) to combine the messages. Get the arrays
     $\{X_{ts}(k)\}$ and $\{W_{ts}(k)\}$. Note that if an observation is available for t, it is used to contract the resulting boxes
     using Algorithm 2.
4. Weights correction: $W_{ts}(k) \longleftarrow W_{ts}(k)/|X_{ts}(k)|$.
5. Propagate the boxes through the model $x_s = f(x_t, v, \mathbf{e})$.
6. Weights normalization:
   $W_{ts}(k) \longleftarrow W_{ts}(k)/sum(W_{ts})$.
```

Notes About the Algorithm

1. For the first iteration of message-passing between neighboring nodes, the messages are initialized using the local likelihood (obviously no previous messages exist). This is shown in the first two items of point 3 in *Algorithm 3*. If no observation is available for the sending node t, no message could be sent, at the first iteration, from t to any of its neighbors. If, however, an observation is available for t, we use it to generate uniformly N boxes $[x_t^k]$ mutually disjoint and weighted equally with $\frac{1}{N}$.
2. When propagating the boxes $[x_{ts}^k]$, representing an information about x_t, through the model f, if an observation is available for the receiving node s, i.e. a prior domain for x_s is known, *contraction* techniques (see Sect. 5.1) might be used to reduce the size of the forwarded boxes.
3. To compute the belief at any node t, at a certain iteration, points 1–3 of *Algorithm 3* are used with the difference that we actually combine all incoming messages to t from the set of all its neighbors, Γ_t.

Figure 8 summarizes the operations of BP in the bounded error context.

6 Self Localization in Sensor Networks: Problem Formulation

A number n of sensors is randomly deployed in a planar region. Each sensor has noisy measurements of its distances from neighboring sensors. The position of sensor t is denoted $\mathbf{x}_t,\ t = 1, \ldots, n$. A small number of nodes (called anchors) have significant a priori information, $p_t(\mathbf{x}_t)$, about their positions. These sensors can be placed manually or can be equipped with a GPS module if manual placement is impractical. Two sensors are able to communicate if the distance separating them is less than some maximum range denoted R. Let d_{ts} denote the noisy measurement of the distance between sensors s and t, then:

$$d_{ts} = ||\mathbf{x}_t - \mathbf{x}_s|| + v_{ts}, \quad v_{ts} \sim p_v(\mathbf{x}_t, \mathbf{x}_s), \tag{6.1}$$

where $p_v(\mathbf{x}_t, \mathbf{x}_s)$ refers to the noise probability distribution and $||\ .\ ||$ denotes the Euclidean distance between t and s. The binary random variable o_{ts} indicates whether the distance d_{ts} is observed or not. According to the assumption above:

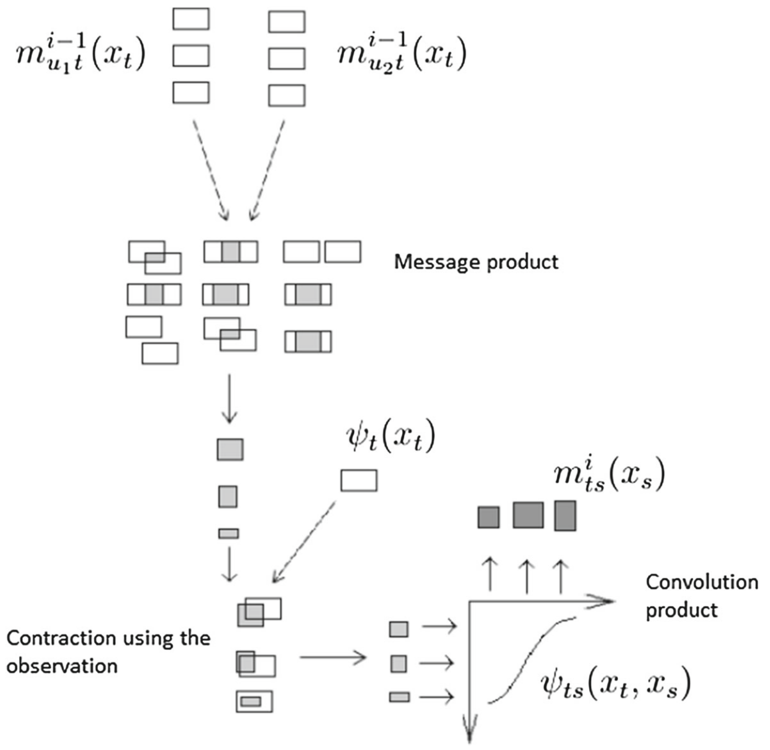

Fig. 8 Belief propagation combined with interval analysis

$$P(o_{ts} = 1) = \mathbf{1}_{||\mathbf{x}_t - \mathbf{x}_s|| \le R}. \tag{6.2}$$

The joint probability distribution $p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \{d_{ts}\}_{(t,s):o_{ts}=1})$ can be factorized as follows:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \{d_{ts}\}_{(t,s):o_{ts}=1}) = \prod_{(t,s):o_{ts}=1} p(d_{ts}|\mathbf{x}_t, \mathbf{x}_s) \prod_t p_t(\mathbf{x}_t). \tag{6.3}$$

Set

$$\begin{aligned} \psi_{ts}(\mathbf{x}_t, \mathbf{x}_s) &= p(d_{ts}|\mathbf{x}_t, \mathbf{x}_s) \\ &= p_v(d_{ts} - ||\mathbf{x}_t - \mathbf{x}_s||) \quad if \quad o_{ts} = 1, \\ \psi_t(\mathbf{x}_t) &= p_t(\mathbf{x}_t), \end{aligned} \tag{6.4}$$

then

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \{d_{ts}\}_{(t,s):o_{ts}=1}) = \prod_t \psi_t(\mathbf{x}_t) \prod_{t,s} \psi_{ts}(\mathbf{x}_t, \mathbf{x}_s). \tag{6.5}$$

The undirected graph describing this joint probability distribution is a graph in which each node represents a sensor and each arc models an established neighborhood between two sensors. This factorization indeed justifies the intuitive representation of this system by a graph wherein each edge represents an actual physical link established between neighboring sensors/nodes allowing them to communicate and to mutually detect each other.

Both algorithms, the novel BP with box representations and NBP, are tested using the scenario described herein. For the novel box-BP algorithm, message passing between two neighboring nodes, namely t and s, is conducted as follows. For the first iteration, messages are initialized using the local observations. Only anchor nodes are thus able to send informative messages to their neighbors, since they are the only nodes with significant a priori information,

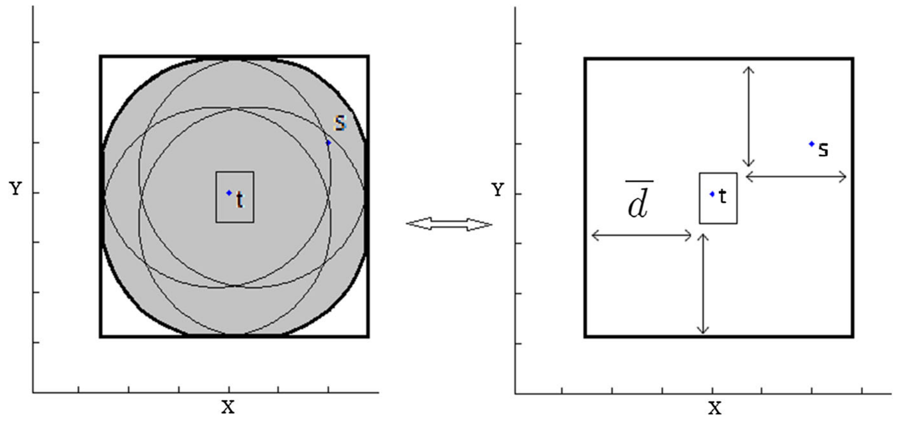

Fig. 9 An inclusion function for the propagation model. The *small box* represents the position of sensor t. The *circles* of radius $\overline{d}$ centered at the four summits of this *box* are shown. The *bigger box* represents the possible position of sensor s. The true positions of t and s are also shown

$p_t(\mathbf{x}_t)$, about their positions. The prior information being represented by a box, N boxes $[\mathbf{x}_t^j]^0$, $j = 1, \ldots, N$, mutually disjoint and weighted equally with $\frac{1}{N}$ are uniformly generated from the available observations at the anchors.

At an iteration i, the message product at node t is represented by a collection $\{\omega_t^{(i)}, [\mathbf{x}_t]^{(i)}\}$ of N weighted box-particles, obtained using *Algorithm 1*. These boxes are contracted and re-weighted using the local evidence at t as described in *Algorithm 2*. Note that in this application, we are in the special case of an observation y_t of the same nature as the state x_t, the contraction reduces into an intersection between $[y_t]$ and $[x_t]^i$. The forwarded message from t to s, separated by noisy distance d_{ts}, is then computed by propagating these boxes $\{[\mathbf{x}_t]^{(i)}\}$ through the following model, as shown in the approximation (5.17):

$$\mathbf{m}_{ts}^{(i)} = \mathbf{x}_t^{(i)} + d_{ts}\begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix}, \tag{6.6}$$

where $\theta \in [0, 2\pi]$. Figure 9 illustrates the result of propagating a box through the model (6.6). Let $[\mathbf{x}_t] = [\underline{x}_t \;\; \overline{x}_t] \times [\underline{y}_t \;\; \overline{y}_t]$ and $[d_{ts}] = [\underline{d} \;\; \overline{d}]$. Then:

$$\begin{aligned} [d_{ts}] \times cos[\theta] &= [\underline{d} \;\; \overline{d}] \times [-1 \;\; 1] \\ &= [-\overline{d} \;\; \overline{d}], \end{aligned}$$

$$\begin{aligned} [x_t] + [d_{ts}] \times cos[\theta] &= [\underline{x}_t \;\; \overline{x}_t] + [-\overline{d} \;\; \overline{d}] \\ &= [\underline{x}_t - \overline{d} \;\; \overline{x}_t + \overline{d}]. \end{aligned}$$

Similarly,

$$[y_t] + [d_{ts}] \times sin[\theta] = [\underline{y}_t - \overline{d} \;\; \overline{y}_t + \overline{d}].$$

The implementation of NBP for self localization in sensor networks is described next. For the first iteration, N particles $\{\omega_t^0, \mathbf{x}_t^0\}$ are sampled from $p_t(\mathbf{x}_t)$. At an iteration i of NBP, the message product at node t is represented by a collection $\{\omega_t^i, \mathbf{x}_t^i\}$ of weighted particles. The weights are corrected using the local evidence at t. The forwarded message from t to s, separated by noisy distance d_{ts}, is then computed by propagating the samples $\{\mathbf{x}_t^i\}$ through the model (6.6) where $\theta \sim U([0, 2\pi])$.

7 Simulation Results

We present next some simulation results. In our simulations, 100 sensors are randomly deployed in a planar region $L \times L$.

7.1 Grid-Like Placement of the Anchors

Basically, two parameters can have impact on the solution provided by a localization algorithm: the range within which the sensors can communicate and the number of anchor nodes. Varying the communication range R leads to a variation of the number of neighborhoods established. Thus, increasing R, more links between neighboring nodes are established and a more complex graph is obtained. In [14], only three anchor nodes are used for calibration. In order to obtain accurate results for NBP, the authors choose to increase the number of neighborhoods/links established between sensors. This resulted in a very dense graphical model and thus complex computations and more energy and time consumption. Drawing on scenarios presented in [8,21], we choose to place nine anchors in a grid-like position. Eight of them are located on the contour of the region, these can be placed manually, and one anchor is at the center of the network. In fact, positioning algorithms perform better when anchors surround the nodes with unknown positions [21]. Intuitively, nodes at the edges of the graph are less likely to be connected making their localization more difficult. The range of communication is set to $R = L/4$. The noise standard deviation is set to $0.005L$. Figure 10 shows the distribution of the sensors. Anchor nodes are marked by circles. Figure 11 shows the corresponding graphical model for $R = L/4$.

Table 1 summarizes the results obtained for both NBP and box-BP algorithms. The simulations were carried out using Matlab on Intel Core i7-3520M processor (2.90 GHz–4 MB Cache, Dual-core) for $L = 100$ m. The error refers to the root mean squared error (RMSE), it shows the mean distance between the true and the estimated position of a sensor and is given as a percentage of L. For the box-BP algorithm, the error is calculated using the distance between the center of the boxes and the true position of the sensors. Note that both algorithms achieve comparable accuracy (see Table 1). However, in order to represent the posterior pdf, NBP stores 200 weighted particles, which corresponds, in our 2D application to a total of 600 floating points values. The box-BP algorithm uses only 9 box particles, that is equivalently 45 floating points values. This reduced storage capability is a great enhancement in terms of energy saving and bandwidth needed for the information exchanged in the network. Furthermore, the box-PF is about ten times faster than NBP (see Table 1).

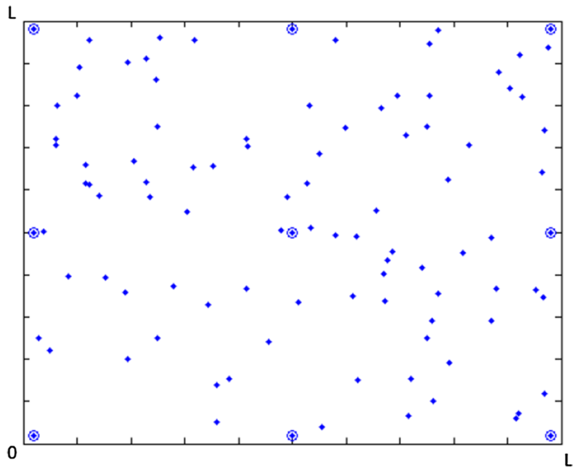

Fig. 10 A scenario with 100 sensors and 9 anchors

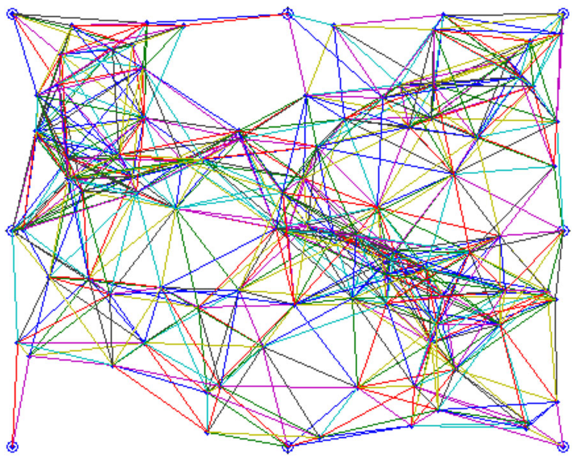

Fig. 11 The corresponding graph for $R = L/4$

Table 1 Simulation results

Algorithm	Error (%)	No of particles	Time (s)
Box-BP	2.11	9	14.54
NBP	2.08	200	159.4

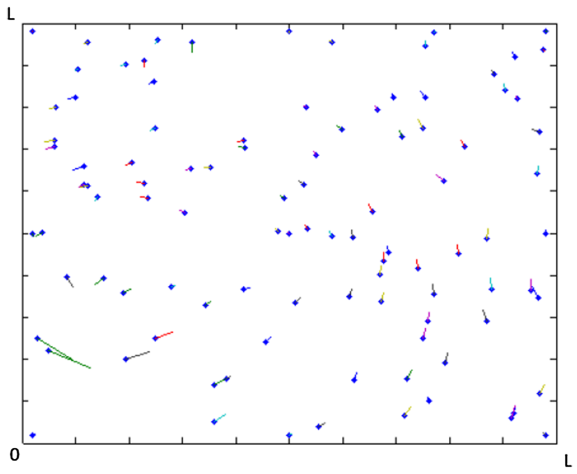

Fig. 12 Results for NBP algorithm

Figures 12 and 13 illustrate the results obtained for NBP and box-BP respectively. The dots denote the true sensors positions whereas the lines indicate the error on these positions.

7.2 Random Placement of the Anchors

Next, we test both message-passing algorithms on different anchor layouts. Rather than placing the anchors in a grid-like position so as to enclose the randomly spread sensors, the anchors are placed randomly as shown in Fig. 14.

We also study the performance of the localization algorithms while varying the number of anchors within the range 6–9. The results are grouped in Tables 2 and 3 for the Box-Bp and the NBP respectively.

Table 2 shows that the novel box-BP algorithm presents comparable results in term of the RMSE for different number of anchors. However, this error rate is achieved within less time, i.e. less iterations, if the number of anchors

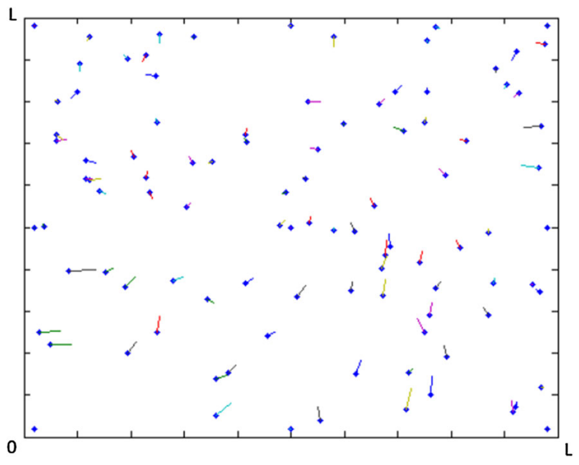

Fig. 13 Results for box-BP algorithm

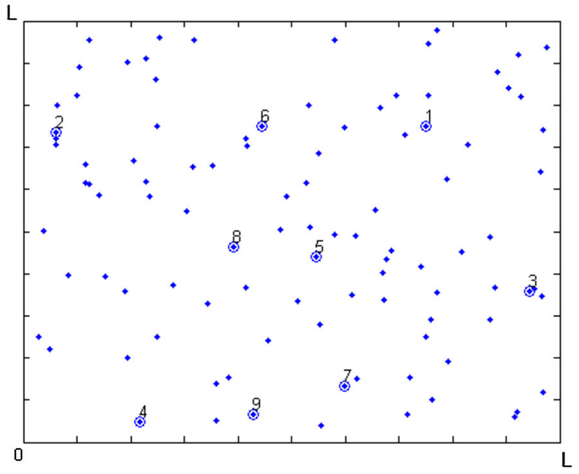

Fig. 14 A scenario with 100 sensors and 9 randomly spread anchors. The anchors are marked as *circles*

Table 2 Results fo box-BP algorithm

No of anchors	6	7	8	9
No of particles	9	9	9	9
Error (%)	2.53	2.08	2.08	2.06
Time (s)	22.80	22.87	15.72	14.27

Table 3 Results for the NBP algorithm

No of anchors	6	7	8	9
No of particles	200	200	200	200
Error (%)	6.89	11.05	12.23	10.72
Time (s)	244.33	299.96	220.41	220.87

is increased. The only condition is to have the anchors uniformly deployed across the region. Figure 15 shows the results obtained using box-BP and a layout with six anchors.

The NBP turned out to be more sensitive to the anchors layout. Figure 16 shows the results obtained using the NBP algorithm for a six anchor layout. It can be seen that the sensors on the edge of the region could not be well

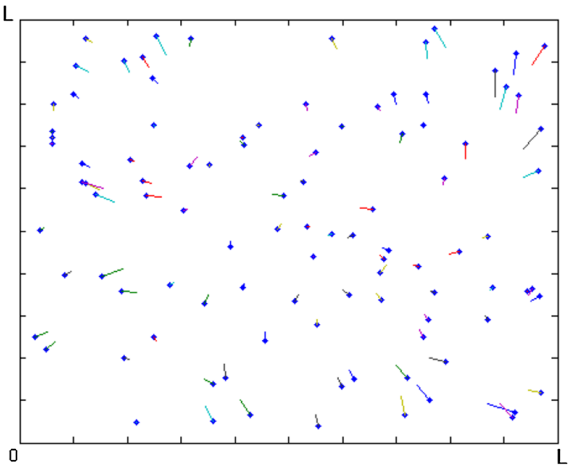

Fig. 15 Results for the box-BP algorithm for six anchors

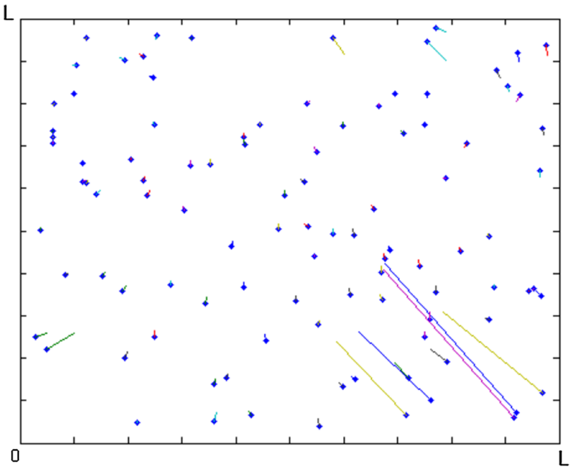

Fig. 16 Results for the NBP algorithm for six anchors

positioned. The error on the position of some of these is around $L/3$. This huge error implies a large value of the RMSE as it can be seen in Table 3. This was observed for all layouts defined by different anchor number.

8 Conclusion

In this paper, we introduced a message-passing algorithm that uses interval representations of probability quantities to infer on arbitrary graphical models. The simulation results showed that for a grid-like placement of the anchors the estimation accuracy provided by NBP algorithm is achieved by the novel box-BP algorithm using much less particles and within less computational time. For a random placement of the anchors, the NBP algorithm failed to accurately locate sensors on the edge of the network. The box-BP provided in this case more accurate results. The advantages offered by the new algorithm are reducing the required storage memory, bandwidth and energy needed to exchange information between nodes of a network. Another advantage is decreasing the complexity of the computations and the computational time.

However, in this paper, a parallel update scheme was adopted. In this sense, at each iteration, each node sends information to all of its neighbors simultaneously. An interesting subject is the problem of finding an optimal message-passing order, or an optimal path for the exchanged information, for which the message-passing algorithm gives a reasonable approximation of the posterior marginal of interest. Furthermore, the application we considered is described by a time independent static model. In fact, dynamic graphical models, namely dynamic Bayesian networks, theory does exist. The structure of this graphical model encodes conditional independence properties between the variables of the model in each time step and also across time steps. This representation is especially useful when reasoning about a system whose state changes over time [22]. The mapping problem is such an example because it involves moving robots whose positions change over time. The application of message-passing procedures to infer in such problems also offers an interesting research subject.

References

1. Arbula, D.: Distributed algorithm for anchor-free network localization using angle of arrival. In: Proceedings of IEEE International Symposium on Industrial Electronics, Cambridge, pp. 792–797 (2008)
2. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non Gaussian Bayesian tracking. IEEE Trans. Signal Process. **50**, 174–188 (2002)
3. Bahi, J.M., Makhoul, A., Mostefaoui, A.: A mobile beacon based approach for sensor network localization. In: Proceedings of Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, White Plains, pp. 44–51 (2007)
4. Ben-Gal, I.: Encyclopedia of Statistics in Quality and Reliability, Chapter Bayesian Networks. Wiley, New York (2007)
5. Bishop, C.M.: Pattern Recognition and Machine Learning, Chapter 8: Graphical Models. Springer, New York (2006)
6. Cheung-Mon-Chan, P.: Réseaux Bayésiens et Filtres Particulaires pour l’égalisation adaptative et le décodage conjoints. PhD thesis, École Nationale Supérieure des Télécommunications (2003)
7. Cook, B.W., Lanzisera, S., Pister, K.S.J.: Soc issues for rf smart dust. In: Proceedings of the IEEE, vol. 94, pp. 1177–1196 (2006)
8. Essoloh, M., Richard, C., Snoussi, H.: Localisation distribuée dans les réseaux de capteurs sans fil par résolution d’un problème quadratique. In: Colloque GRETSI Troyes (2007)
9. Gning, A., Mihaylova, L., Abdallah, F.: Mixture of uniform probability density functions for non linear state estimation using interval analysis. In: Proceedings of the 13th Conference on Information Fusion, Edinburgh, pp. 1–8 (2010)
10. Gning, A., Mihaylova, L., bdallah, F., Ristic, B.: Integrated Tracking, Classification, and Sensor Management: Theory and Applications, Chapter Particle Filtering Combined with Interval Methods for Tracking Applications. Wiley, New York (2012)
11. Gning, A., Ristic, B., Mihaylova, L., Abdallah, F.: An introduction to box particle filtering. IEEE Signal Process. Mag. **30**(4), 166–171 (2013)
12. Heurtefeux, K., Valois,F.: Localisation collaborative pour réseaux de capteurs. In: Colloque Francophone sur l’Ingénierie des protocoles (CFIP) Les Arcs, France (2008)
13. Ihler, A.T.: Inference in sensor networks: graphical models and particle methods. PhD thesis, Massachusetts Institue of Technology (2005)
14. Ihler, A.T., Fisher, J.W., Moses, R.L., Willsky, A.S.: Nonparametric belief propagation for self-localization of sensor networks. IEEE J. Sel. Areas Commun. **23**(4), 809–819 (2005)
15. Jaulin, L.: Nonlinear bounded-error state estimation of continuous-time systems. Automatica **38**(6), 1079–1082 (2002)
16. Jaulin, L., Walter, E.: Set inversion via interval analysis for nonlinear bounded-error estimation. Automatica **29**(4), 1053–1064 (1993)
17. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer, New York (2001)
18. Johansen, A.M., Doucet, A.: A tutorial on particle filtering and smoothing: fifteen years later. In: The Oxford Handbook of Nonlinear Filtering, pp. 4-6 (2009)
19. Julier, S., Uhlmann, J., Durrant-White, H.: A new approach for filtering nonlinear systems. In: Proceedings of the American Control Conference, Washington, DC (1995)
20. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inf. Theory **47**(2), 498–519 (2001)
21. Langendoen, K., Reijers, N.: Distributed localization in wireless sensor networks: a quantitative comparison. Comput. Netw. Int. J. Comput. Telecommun. Netw. Spec. Issue Wirel. Sens. Netw. **43**(4), 499–518 (2003)
22. Paskin, M.A.: Exploiting locality in probabilistic inference. PhD thesis, University of california, Berkeley (2004)
23. Rong, P., Sichitiu, M.L.: Angle of arrival localization for wireless sensor networks. In: Proceedings of the Third annual IEEE Communications society on Sensor and Ad Hoc Communications and Networks, Reston, vol. 1, pp. 374–382 (2006)
24. Saad, C., Benslimane, A., Konig, J.C.: At-angle: a distributed method for localization using angles in sensor networks. In: Proceedings of IEEE Symposium on Computers and Communications, Marrakech, pp. 1190–1195 (2008)

25. Shang, Y., Rumel, W., Zhang, Y., Fromherz, M.: Localization from connectivity in sensor networks. In: Proceedings of IEEE Transactions on Parallel and Distributed Systems, vol. 15. pp. 961–974 (2004)
26. Sigal, L.: Continuous state graphical models for object localization, pose estimation and tracking. PhD thesis, Brown University, Providence (2008)
27. Silverman, B.W.: Density estimation for statistics and data analysis. In: Monograph on Statistics and Applied probability. Chapman and Hall, London (1986)
28. Sudderth, E.B., Ihler, A.T., Freeman, W.T., Willsky, A.S.: Nonparametric belief propagation. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. I, pp. 605–612 (2003)
29. Sudderth, E.H.: Graphical models for visual object recognition and tracking. PhD thesis, Massachusetts Institute of Technology (2006)
30. Wan, E.A., Van der Merwe, R.: The unscented Kalman filter for nonlinear estimation. In: Adaptive Systems for Signal Processing, Communications and Control Symposium (AS-SPCC), pp. 153–158. IEEE, Lake Louise (2000)
31. Welch, G., Bishop, G.: An introduction to the Kalman filter. In: Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles. ACM Press, Addison-Wesley, USA (2001)