Available online at www.sciencedirect.com

SCIENCE DIRECT®

Mathematical and Computer Modelling 43 (2006) 1114–1135

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

Data association based on optimization in graphical models with application to sensor networks

Lei Chen[a], Martin J. Wainwright[b], Müjdat Çetin[c,*], Alan S. Willsky[a]

[a] *Laboratory for Information and Decision Systems, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, United States*

[b] *Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, United States*

[c] *Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey*

Abstract

We propose techniques based on graphical models for efficiently solving data association problems arising in multiple target tracking with distributed sensor networks. Graphical models provide a powerful framework for representing the statistical dependencies among a collection of random variables, and are widely used in many applications (e.g., computer vision, error-correcting codes). We consider two different types of data association problems, corresponding to whether or not it is known a priori which targets are within the surveillance range of each sensor. We first demonstrate how to transform these two problems to inference problems on graphical models. With this transformation, both problems can be solved efficiently by local message-passing algorithms for graphical models, which solve optimization problems in a distributed manner by exchange of information among neighboring nodes on the graph. Moreover, a suitably reweighted version of the max–product algorithm yields provably optimal data associations. These approaches scale well with the number of sensors in the network, and moreover are well suited to being realized in a distributed fashion. So as to address trade-offs between performance and communication costs, we propose a communication-sensitive form of message-passing that is capable of achieving near-optimal performance using far less communication. We demonstrate the effectiveness of our approach with experiments on simulated data.

1. Introduction

Recent years have witnessed the emergence of a new approach to sensing applications, involving the deployment of a large number of small and relatively inexpensive sensors. Such ad hoc sensor networks have the potential to provide enhanced spatio-temporal sensing coverage in ways that are either prohibitively expensive or even impossible using more conventional approaches to sensing [1,2]. However, realizing the potential of these large and distributed sensor networks requires the development of techniques for *distributed data association and estimation* using sensing and wireless communication nodes with constrained capacities for both computation and communication. This paper

* Corresponding author. Tel.: +90 216 483 9594.
E-mail addresses: lchen@mit.edu (L. Chen), wainwrig@eecs.berkeley.edu (M.J. Wainwright), mcetin@sabanciuniv.edu (M. Çetin), willsky@mit.edu (A.S. Willsky).

0895-7177/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2005.12.002

is devoted to distributed techniques for solving data association problems, using the framework of graphical models. Our primary motivation comes from problems that arise in multiple target tracking with distributed sensor networks.

One central problem in multiple target tracking is data association—i.e., the problem of determining the correct correspondence between measurements and tracks [3,4]. One of the most widely used approaches to data association is the multiple hypothesis tracking (MHT) [5] algorithm. A distributed version of the MHT algorithm was proposed by Chong et al. [6]. However, for a multi-sensor surveillance system, the number of association hypotheses at each time frame increases exponentially with the number of sensors in use, so that the cost of using either centralized or distributed MHT becomes prohibitive for large-scale sensor network applications. Traditional approaches for data association usually exploit a hierarchical architecture by introducing fusion nodes above the level of sensor nodes. Each fusion node forms the local hypotheses based on the information from a few local sensors [7], and then global hypotheses are constructed at an even higher level based on the pruned hypotheses at each fusion node. However, since the total number of fusion nodes and the number of hypotheses kept at each fusion node must be limited, only suboptimal solutions can be obtained.

In this paper, we propose a new approach to solving data association problems in a distributed fashion. Our approach scales well with respect to the number of sensors, thereby rendering feasible optimal data association in applications involving large-scale sensor networks. Moreover, even if applied to centralized data association, our approach is still more efficient than the traditional ones. Our work makes use of the framework of *graphical models* [8,9]. This class of models are well suited to represent the structure of statistical dependencies in a collection of random variables. Accordingly, they are widely used in many applications, including computer vision [10], speech recognition [11], and error-correcting codes [12]. However, graphical models have never been applied in solving data association problems. The complexity of solving estimation problems in graphical models depends critically on the underlying graph structure. On one hand, for graphs without cycles (i.e., trees), it is well known that there exist extremely efficient algorithms for performing optimal estimation [13,14]. However, once there are loops in a graph (as is most often the case in sensor networks), optimal estimation — or even suboptimal estimation with guaranteed levels of performance — poses a significant challenge.

Several reasons underlie our choice of graphical models for addressing the data association problem in sensor networks. First, a graphical model is well suited to capture the structure of a sensor network, which consists of nodes (for sensing, communication, and computation), as well as connections between the nodes (for modeling statistical dependencies, and/or communication links). Second, there has recently been significant progress in the development and analysis of efficient algorithms in graphs with cycles. On one hand, there has been recent work on analyzing and understanding the behavior of the well-known class of *local message-passing* algorithms on loopy graphs [15–18]; on the other hand, there are new, more powerful algorithms, that can provide provably optimal solutions to problems such as maximum a posteriori estimation in loopy graphical models [19]. Third, the algorithms commonly used for performing estimation in graphical models involve *parallel message-passing* operations that are not only efficient, but also well suited to realization via physically distributed processors in parallel, which is a crucial requirement in many sensor network applications. Fourth, graphical models provide a suitable framework in which we can develop and analyze communication-constrained versions of message-passing algorithms, which are necessary given the severe power and energy limitations of sensor networks.

A key issue addressed by our work is how to transform the data association and estimation problems arising in distributed sensing scenarios into the formalism of graphical models. We describe our approach to this issue in the context of two specific problems. In the first problem, we assume that the sensor network is already *organized*, meaning that each sensor knows the set of targets it can observe, and each sensor produces position/bearing/range measurements for the targets in its local surveillance region. With this set-up, the goal is to determine an association between measurements and targets. One can envisage two approaches to constructing a graphical model for this data association problem: either by mapping the sensors to nodes, or by mapping the targets to nodes. These two approaches are roughly analogous to existing sensor-centric or target-centric approaches, respectively, to the data association problem. In centralized data association, target-centric approaches have been favored [20], whereas a sensor-centric approach appears more appealing for distributed processing in sensor networks [7]. Interestingly, our analysis of sensor networks leads naturally to mixed sensor–target representations. In the second problem, we consider a scenario where the sensor network must perform self-organization, meaning that which sensor can see which target must be determined. For simplicity, we assume that sensors behave as proximity indicators. The key issue is to estimate the distribution of targets over the surveillance area. A method for solving this target distribution estimation problem can be viewed as a preprocessing step for the first problem. We propose a region-based modeling approach, in which elementary variables to be estimated are the numbers of targets in each of a set of disjoint subregions covering the surveillance area. In particular, each subregion corresponds to the area surveyed by a distinct subset of sensors. The measurements at each sensor correspond to the number of targets detected within the overall range of that sensor, with some uncertainty due to noise. The various modeling approaches for these two specific problems illustrate how the role of nodes in the graphical model changes according to the underlying estimation problem of interest. For example, the nodes corresponds to sensors or targets in the first problem, whereas they correspond to subregions or sensors in the second problem.

We proceed by transforming each problem described above to a suitable optimization problem in the graphical model. We then show how such problems can be solved efficiently, by using local message-passing algorithms that exploit sparsity inherent in the problem due to the local sensing structure. We use both standard message-passing algorithms, and also the more recently developed *tree-reweighted max–product* (TRMP) algorithm [19]. Whereas standard message-passing algorithms provide no correctness guarantees on loopy graphs, the TRMP algorithm renders it possible to obtain optimal data association (in the maximum a posteriori sense) even on graphs with cycles [19]. In these algorithms, distributed inference is achieved iteratively through the exchange of information among neighboring nodes on the graph in parallel, where each iteration requires a certain amount of communication between the nodes. This parallel message-passing structure makes these algorithms especially suitable for a distributed implementation, in which each node performs local processing and then transmits the results to its neighbors on the graph. We demonstrate the effectiveness and the computational efficiency of this approach for a number of simulated scenarios.

In a distributed implementation, the message-passing operations correspond to communication between the computation nodes in the sensor network. In the standard algorithms described above, the nodes continue transmitting messages until the overall algorithm converges, without making any distinction between messages that contain significant new information (compared to previous iterations) and ones that do not. A key limitation of typical sensor networks is the budget for communication, since communication consumes power, and the nodes are usually power-limited. Hence, it is of considerable interest to develop algorithms that treat communication bandwidth as a limited resource that needs to be used wisely and sparingly. To this end, we propose a new algorithm based on the message-passing algorithm, where we provide nodes with the authority to decide whether or not messages should be sent at each iteration, based on statistical rules related to the information content of messages. We show that such communication-sensitive algorithms can provide considerable savings in communication without much sacrifice in overall decision-making performance. Furthermore, this approach provides insight into the trade-offs between the performance achieved and the amount of communication needed by sensors, as well as into the information flow dynamics inside sensor networks. For example, we observe that a node that has decided not to communicate for a while may suddenly start passing messages again as new information finally reaches it.

The remainder of this paper is organized as follows. Section 2 introduces the essential background of graphical models and message-passing algorithms, as well as the tree-reweighted max–product algorithm. Section 3 contains the mathematical formulation of the measurement-to-target association problem in an organized sensor network and our approach to transforming it to an optimization problem on graphical models. Section 4 proceeds with the formulation and transformation of target distribution estimation for sensor network self-organization. An adaptive technique, communication-sensitive message-passing, is proposed in Section 5. Experimental results are presented in Section 6. We summarize our work and discuss directions for future research in Section 7.

2. Background on graphical models

2.1. Graphical models

A graphical model consists of a collection of random variables that are associated with the nodes of a graph. Although various formalisms [8] exist for graphical models, the work in this paper focuses entirely on *Markov random fields* (MRF), which are based on undirected graphs. More formally, let $\mathcal{G} = (V, E)$ be an undirected graph, where V is a set of nodes or vertices, and E is a set of edges. For each vertex $s \in V$, let $\mathbf{x}_s$ be a random variable that takes values in the discrete set $\mathcal{X} := \{0, 1, \ldots, m-1\}$. Concatenating these variables together yields the $N = |V|$ dimensional random vector $\mathbf{x} := \{\mathbf{x}_s \mid s \in V\}$, which takes values in the Cartesian product space $\mathcal{X}^N$.

The graphical model formalism requires that the random vector $\mathbf{x}$ satisfy certain Markov properties associated with the graph $\mathcal{G}$. For any subset $S \subset V$, we define $\mathbf{x}_S := \{x_s \mid s \in S\}$. Let D be a vertex cutset in the graph, so that removing it separates the graph into at least two pieces A and B. We say that $\mathbf{x}$ is *Markov* with respect to the graph if $\mathbf{x}_A$ and $\mathbf{x}_B$ are conditionally independent given $\mathbf{x}_D$. In this case, the Hammersley–Clifford theorem [21] guarantees that the distribution $p(\mathbf{x})$ can be factorized as the product of functions defined on the cliques (i.e., fully connected subsets of the vertex set V) of the graph. In particular, associated with each clique C is a *clique compatibility function* $\psi_C : \mathcal{X}^{|\mathbf{x}_C|} \to \mathbb{R}^+$ that depends only on $\mathbf{x}_C$. With this notation, a MRF consists of a collection of random variables $\mathbf{x}$ and the distribution $p(\mathbf{x})$ that factorize as

$$p(\mathbf{x}) = \frac{1}{\kappa} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C), \tag{1}$$

where $\mathcal{C}$ is the set of all cliques of $\mathcal{G}$, and κ denotes a normalization constant.[1] In many applications, the random vector $\mathbf{x}$ is not observed; given instead are independent noisy observations $\mathbf{y} = \{y_s \mid s \in V\}$ at all (or some) of the nodes, on which basis one would like to draw inferences about $\mathbf{x}$. The effect of including these measurements — i.e., the transformation from the prior distribution $p(\mathbf{x})$ to the conditional distribution $p(\mathbf{x} \mid \mathbf{y})$ — is simply to modify the factors in (1). As a result, we suppress explicit mention of measurements in this section, since the problems of computing marginals for $p(\mathbf{x} \mid \mathbf{y})$ and $p(\mathbf{x})$ are of identical structure and complexity.

The distribution $p(\mathbf{x})$ is central to various inference problems for graphical models. Of interest in this paper are the problems of estimating the marginal distribution $p(\mathbf{x}_s) = \sum_{\mathbf{x}' \text{ s.t } \mathbf{x}'_s = \mathbf{x}_s} p(\mathbf{x}')$ for each variable $\mathbf{x}_s$ and finding the maximum a posteriori (MAP) configuration $\hat{\mathbf{x}} = \arg\max_{\mathbf{x} \in \mathcal{X}^N} p(\mathbf{x})$. As we describe in the following subsection, local message-passing algorithms, including the sum–product and max–product algorithms, can be used to solve these problems efficiently, either in an exact or in an approximate manner.

Throughout this paper, we focus on MRFs with *pairwise* compatibility functions, for which compatibility functions are defined only for singleton cliques (individual nodes) and pairwise cliques (pairs of nodes joined by edges). For a MRF with pairwise compatibility functions, the factorization of $p(\mathbf{x})$ in (1) takes the simpler form

$$p(\mathbf{x}) = \frac{1}{\kappa} \prod_{s \in V} \psi_s(\mathbf{x}_s) \prod_{(s,t) \in E} \psi_{st}(\mathbf{x}_s, \mathbf{x}_t), \tag{2}$$

where $\psi_s(\mathbf{x}_s)$ is the *node compatibility function* that depends only on the individual variable $\mathbf{x}_s$, and $\psi_{st}(\mathbf{x}_s, \mathbf{x}_t)$ is the *edge compatibility function* that depends only on the variables $\mathbf{x}_s$ and $\mathbf{x}_t$ joined by edge (s, t).

In principle, the pairwise assumption entails no loss of generality, since any MRF with higher order clique compatibility functions can be converted to an equivalent MRF in such a pairwise form. Moreover, it simplifies our exposition, since the most straightforward form of message-passing was originally developed on tree-structured graphs, for which any MRF takes this pairwise form. Although there are generalized forms of message-passing that entail operating on higher order cliques, these generalizations usually involve clustering the nodes within a higher order clique to one node [22]. Since the complexity of message-passing algorithms grows exponentially in the cluster size, concerns of computational tractability limit the appeal of such approaches. For these reasons, it is appropriate to limit our discussion to MRFs with pairwise compatibility functions. The standard message-passing algorithms as well as the tree-reweighted max–product algorithm are introduced in the next two subsections under this assumption.

2.2. Message-passing algorithms

Message-passing techniques [13,23,24], including the sum–product algorithm and the max–product algorithm, are generalizations of the widely used forward–backward algorithm and Viterbi algorithm on Markov chains to arbitrary tree-structured graphs. There are different ways to schedule the passing order of the messages [14]. Here we consider a parallel scheme, in which at each iteration, each node t passes a message to each of its neighbors $s \in \mathcal{N}(t)$ simultaneously and in parallel. The message in the nth iteration, which we denote by $M^n_{ts}(\mathbf{x}_s)$, is a function of the possible states $\mathbf{x}_s \in \mathcal{X}_s$. This parallel message-passing operation is inherently a distributed algorithm, so that it can be realized on physically distributed processors.

[1] We use this notation throughout the paper, where the value of κ may change from line to line.

The sum–product algorithm (also known as belief propagation) is used to compute the node marginal distribution $p(\mathbf{x}_s)$. In this particular instance, the message M_{ts}^n is computed recursively using

$$M_{ts}^n(\mathbf{x}_s) = \kappa \sum_{\mathbf{x}_t'} \left\{ \psi_{st}(\mathbf{x}_s, \mathbf{x}_t') \psi_t(\mathbf{x}_t') \prod_{u \in \mathcal{N}(t) \setminus s} M_{ut}^{n-1}(\mathbf{x}_t') \right\}, \tag{3}$$

where $\mathcal{N}(t) \setminus s$ is the set of neighbors of node t in the graph $\mathcal{G}$ excluding node s, and κ is a normalization constant (typically chosen to ensure that $M_{ts}^n(\cdot)$ sums to one). For any tree-structured graph, the message update Eq. (3) converges to a unique fixed point $\mathbf{M}^* = \{M_{ts}^*\}$ after a finite number of iterations. The converged values of the messages $\mathbf{M}^*$ can be used to compute the marginal distribution at node s via

$$p(\mathbf{x}_s) = \kappa\, \psi_s(\mathbf{x}_s) \prod_{u \in \mathcal{N}(s)} M_{us}^*(\mathbf{x}_s). \tag{4}$$

The max–product algorithm is a similar form of message-passing that is used to find the MAP configuration $\hat{\mathbf{x}} = \{\hat{\mathbf{x}}_s \mid s \in V\}$. One interpretation [17] of the max–product algorithm is as computing the so-called max-marginals $P_s(\mathbf{x}_s) = \kappa \max_{\{\mathbf{x}' | \mathbf{x}_s' = \mathbf{x}_s\}} p(\mathbf{x}')$ at each node. If, for each node, the max-marginal P_s over $\mathbf{x}_s' \in \mathcal{X}_s$ is attained at a unique value, then it can be shown [17] that the MAP configuration $\hat{\mathbf{x}}$ is unique, with elements given by $\hat{\mathbf{x}}_s = \arg\max_{\mathbf{x}_s' \in \mathcal{X}_s} P_s(\mathbf{x}_s')$. In the max–product algorithm, the messages are updated according to the recursion

$$M_{ts}^n(\mathbf{x}_s) = \kappa \max_{\mathbf{x}_t'} \left\{ \psi_{st}(\mathbf{x}_s, \mathbf{x}_t') \psi_t(\mathbf{x}_t') \prod_{u \in \mathcal{N}(t) \setminus s} M_{ut}^{n-1}(\mathbf{x}_t') \right\}. \tag{5}$$

As with the sum–product algorithm, when applied to a tree-structured graph, the message update Eq. (5) converges to a unique fixed point $\mathbf{M}^* = \{M_{ts}^*\}$ after a finite number of iterations.

For tree-structured problems, both the sum–product and max–product algorithms produce exact solutions with complexity $\mathcal{O}(m^2 d)$, where m is the number of states per node, and d is the number of nodes on the longest path in the graph. The same message-passing algorithms are also applied frequently to graphs with cycles, where they serve as approximate methods. In the presence of cycles, the updates may not converge, and need not compute the exact marginal distribution (for sum–product) or the MAP configuration (for max–product). However, there are some results on the quality of the approximate MAP solutions [15–17] and the approximate marginal distributions [18].

2.3. *Tree-reweighted max–product algorithm*

In this section, we describe a modified version of the max–product algorithm that is guaranteed to output a provably optimal MAP configuration, or to acknowledge failure. One way to describe this tree-reweighted max–product (TRMP) algorithm [19] is as a sequence of updates on trees of the graph, using the ordinary max–product algorithm as a subroutine. The basic idea is to represent the original problem on the graph with cycles as a convex combination of tree-structured problems. It can be shown [19] that whenever the tree problems all share an optimal configuration in common, this configuration must be the MAP configuration for the original problem. Based on this idea, the goal of the TRMP algorithm is to find a convex combination of tree-structured problems that share a common optimum.

Let $\vec{\mu} = \{\mu(T)\}$ be a probability distribution over a set of spanning trees $\{T^i \mid i = 1, \dots, L\}$ of the graph. For each edge $(s,t) \in E$, let $\mu_{st} = \Pr_{\vec{\mu}}[(s,t) \in T]$ be the probability that edge (s,t) appears in a tree T chosen randomly under $\vec{\mu}$. We require a choice of $\vec{\mu}$ such that $\mu_{st} > 0$ for all edges of the graph. With this notation, the updates take the following form:

(1) For each spanning tree T^i, $i = 1, \dots, L$, specify a set $\psi^i = \{\psi_s^i, \psi_{st}^i\}$ of compatibility functions for the tree via:

$$\psi_s^i = \psi_s \quad \forall s \in V, \qquad \psi_{st}^i = [\psi_{st}]^{\frac{1}{\mu_{st}}} \quad \forall (s,t) \in E(T^i).$$

(2) For each tree T^i, $i = 1, \dots, L$, run the max–product algorithm until convergence occurs to obtain a set of tree max-marginals $\{P_s^i \mid s \in V\}$ and messages $\{M_{st}^i \mid (s,t) \in E(T^i)\}$.

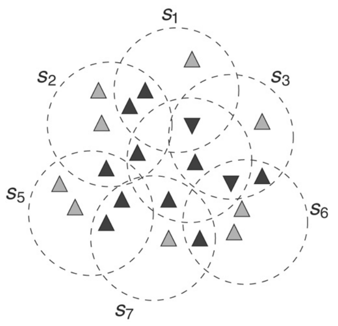

Fig. 1. A coverage configuration of multi-sensor multi-target data association. There are seven sensors (s_4 is the inner sensor, unlabeled) with detection ranges shown as dashed circles. Targets are shown as $\triangle$ and $\triangledown$. Each of the gray $\triangle$ targets falls into the range of exactly one sensor. Black $\triangle$ targets are shared by two sensors and the two black $\triangledown$ targets are shared by three sensors. We assume this sensor–target coverage configuration is known.

(3) Check whether all trees agree on the assignments produced by the max–product algorithm. If yes, output the assignment and stop. If not, update the compatibility functions using

$$\psi_s^{\text{new}} = \exp\left[\sum_{T^i} \mu(T^i) \log P_s^i\right] \quad \forall s \in V,$$

$$\psi_{st}^{\text{new}} = \exp\left[\sum_{T^i} \mu(T^i) \log \frac{\psi_{st}^i}{M_{st}^i M_{ts}^i}\right], \quad \forall (s,t) \in E,$$

and return to step (1) with $\psi \equiv \psi^{\text{new}}$.

It can be shown that this algorithm always has a fixed point for positive compatibilities. The algorithm outputs the correct MAP assignment as long as the max-marginal for each node has a unique optimum, which can be assumed safely in most cases. More details on TRMP and its link to a tree-based linear programming relaxation can be found in [19].

3. Data association in an organized network

Data association is recognized as the central problem of multi-sensor multi-target tracking. This section is devoted to a core problem in data association, with the goal of exploring how graphical models can be used effectively so as to exploit the intrinsic sparsity.

3.1. Problem formulation

The set-up of the core problem that we consider is as follows. The existing tracks of M targets $\{o_1, o_2, \ldots, o_M\}$ with current probability density functions on location for each target are assumed to be known. Moreover, the targets are taken to be independent from one another. We have a surveillance system consisting of N sensors $\{s_1, s_2, \ldots, s_N\}$ with limited but in general overlapping surveillance regions. An example of such a scenario is shown in Fig. 1. Moreover, we assume the organization of the sensor network has been accomplished (i.e., it is known from the sensor surveillance regions which targets can be detected by a given sensor, although false alarms and missed detections are permitted). We divide all targets into N small sets $O_1, O_2, \ldots, O_N$, where the set O_i contains the targets covered by the sensor s_i. A target can be in several sets due to the overlapping coverage. Each sensor measures the state of targets in its surveillance area and generates a set of position/range/bearing measurements $\mathbf{y}_i = \{y_{i\alpha}, \alpha = 1, 2, \ldots, q_i\}$, with q_i being the number of measurements generated by sensor s_i. With this set-up, the goal is to determine the most probable assignment of measurements to the targets. Any assignment must be consistent, meaning that each measurement in $\mathbf{y}_i$ is either assigned to at most one of the targets in O_i, or to none of the targets (hence declared to be a false alarm), and each target in O_i is associated with at most one measurement in $\mathbf{y}_i$, or otherwise corresponds to a missed detection for sensor s_i.

The measurement association for each sensor is coupled with the other sensors due to overlap in sensor coverage regions and the unknown nature of the target states. Therefore, we have to consider the data association at all the sensors jointly in order to obtain the optimal data association. Let $\mathbf{X}$ denote a random variable taking values in the whole association configuration space, representing the possible associations. The essence of the data association problem is to obtain the MAP estimate

$$\hat{\mathbf{X}} = \arg\max_{X} p(\mathbf{X} = X \mid \mathbf{y}). \tag{6}$$

Notice that with N sensors and each sensor covering n targets, $\mathbf{X}$ has $(n!)^N$ different configurations. Although for each particular association configuration X, $p(X \mid \mathbf{y})$ could be computed, it is intractable to enumerate each configuration of $\mathbf{X}$ and compute $\hat{\mathbf{X}}$ for any nontrivial problems.

However, the sparsity inherent in the problem structure can be exploited so as to reduce the computational complexity. Instead of attacking the data association problem as one huge problem modeled by a single variable $\mathbf{X}$ with a huge state space, we can view $\mathbf{X}$ as the concatenation of a collection of local random association variables, one defined for each sensor or for each target. With this view, it is natural to construct a graphical model for the data association problem, which then allows us to make use of the existing inference algorithms to efficiently compute MAP estimates of the association variables. Constructing a graphical model requires identification of the graphical structure, and factorization of the posterior probability $p(\mathbf{X} \mid \mathbf{y})$ into products of local compatibility functions defined on graph cliques. As we will show in subsequent subsections, there are a variety of different approaches to modeling the data association problem, each of which leads to different types of graphical models. For instance, defining an association variable for each sensor leads to a sensor-oriented modeling approach. On the other hand, defining an association variable for each target corresponds to a target-oriented modeling approach. Interestingly, we find that a hybrid modeling approach, combining elements of both the sensor-centric and target-centric approaches, is best suited for solving the data association problems in sensor networks, using the inference algorithms for graphical models. The following subsection is devoted to a detailed description of these modeling approaches.

3.2. Graphical models for data association

In this section, we explain in detail how we construct the graph and the compatibility functions for the data association problem described in Section 3.1. To simplify our discussion and notation, we assume throughout Sections 3.2.1–3.2.3 that there are no false alarms and no missed detections and postpone the discussion on false alarms and missed detections to Section 3.2.4. Without false alarms and missed detections, the measurements generated and the targets covered by each sensor will be in one-to-one correspondence. For simplicity of explanation only, we further assume that the same number of targets, n, are covered by every sensor. With these assumptions, the number of measurements in each sensor equals n. We also assume that each association configuration is equally likely a priori, thus factorizing $p(\mathbf{X} \mid \mathbf{y})$ is equivalent to factorizing the measurement likelihood $p(\mathbf{y} \mid \mathbf{X})$. The task of data association is equivalent to finding the association configuration that maximizes the measurement likelihood.

We begin by describing how to construct a graphical model in which the nodes are in one-to-one correspondence either with sensors or with targets. As we will see, for the *sensor-oriented models*, when more than two sensors see any particular target, higher order clique compatibility functions appear and the structure of the resulting graph becomes more complex. On the other hand, for the *target-oriented models*, higher order clique compatibility functions can always be avoided, but the graph is quite densely connected. These advantages and disadvantages of both models motivate the introduction of additional nodes corresponding to small groups of targets into sensor-oriented models, thereby leading to a *hybrid* but more tractable and sparse graphical structure.

3.2.1. Sensor-oriented modeling

The sensor-oriented modeling approach entails defining a set of association variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, one for each sensor. Each association variable $\mathbf{x}_i$ takes values in the permutation space of the measurement set $\mathbf{y}_i$, and thus has $(n!)$ states. Each state of $\mathbf{x}_i$ corresponds to a valid association configuration of the measurements generated by sensor s_i. We use $x_i(o_r) = y_{i\alpha}$ to indicate that target o_r is assigned the measurement $y_{i\alpha}$ in the particular association configuration x_i for sensor s_i. Then the measurement likelihood in a certain association configuration $X = \{x_1, x_2, \ldots, x_N\}$ can be

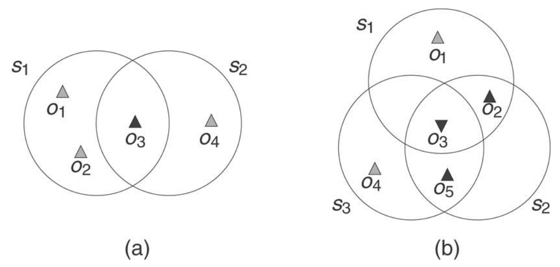

Fig. 2. Two data association scenarios. (a) Two sensors with four targets. (b) Three sensors with five targets. Target o_3 is covered by three sensors.

written as

$$\begin{aligned}
&p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N \mid \mathbf{x}_1 = x_1, \ldots, \mathbf{x}_N = x_N) \\
&= \prod_{i=1}^{N} \left(\prod_{\substack{r \text{ s.t} \\ o_r \in C_1(i)}} p(y_{i\alpha} \mid x_i(o_r) = y_{i\alpha}) \right) \\
&\quad \prod_{\substack{i,j \\ i<j}} \left(\prod_{\substack{r \text{ s.t} \\ o_r \in C_2(i,j)}} p(y_{i\alpha}, y_{j\beta} \mid x_i(o_r) = y_{i\alpha}, x_j(o_r) = y_{j\beta}) \right) \\
&\quad \prod_{\substack{i,j,k \\ i<j<k}} \left(\prod_{\substack{r \text{ s.t} \\ o_r \in C_3(i,j,k)}} p(y_{i\alpha}, y_{j\beta}, y_{k\gamma} \mid x_i(o_r) = y_{i\alpha}, x_j(o_r) = y_{j\beta}, x_k(o_r) = y_{k\gamma}) \right) \\
&\quad \ldots
\end{aligned} \tag{7}$$

where

$$\begin{aligned}
&C_1(i) = \{o_r \mid o_r \in O_i, \text{and } o_r \notin O_l, \forall l \neq i\}, \\
&C_2(i,j) = \{o_r \mid o_r \in O_i \cap O_j, \text{and } o_r \notin O_l, \forall l \neq i, j\}, \\
&C_3(i,j,k) = \{o_r \mid o_r \in O_i \cap O_j \cap O_k, \text{and } o_r \notin O_l, \forall l \neq i, j, k\},
\end{aligned}$$

and O_i is the set of targets covered by sensor s_i. The above factorization defines both the graph structure and the compatibility functions. Since the variables we need to estimate correspond to the sensors, we place a node in the graph for each sensor, and define the node compatibility functions as

$$\psi_i(x_i) = \prod_{\substack{r \text{ s.t} \\ o_r \in C_1(i)}} p(y_{i\alpha} \mid x_i(o_r) = y_{i\alpha}). \tag{8}$$

For two sensors that share a set of targets covered by none of the other sensors, as in the scenario shown in Fig. 2(a), we connect the nodes for these two sensors with an edge because their association variables are coupled directly by the targets they both cover, and define the edge compatibility function as

$$\psi_{ij}(x_i, x_j) = \prod_{\substack{r \text{ s.t} \\ o_r \in C_2(i,j)}} p(y_{i\alpha}, y_{j\beta} \mid x_i(o_r) = y_{i\alpha}, x_j(o_r) = y_{j\beta}). \tag{9}$$

For example, Fig. 3(a) shows the sensor-oriented model for the scenario shown in Fig. 2(a).

This same idea can be applied more generally if we insist on using a graphical structure with nodes in one-to-one mapping with sensors. In particular, if there are targets covered by k sensors with $k \geq 3$, as in the scenario shown in Fig. 2(b), we connect these sensor nodes with each other to form a k-clique (a clique containing k nodes), since they are coupled by the targets they all cover. We use the likelihood of the measurements assigned to the targets covered by all sensors in the clique to define the clique compatibility function. For example, if s_i, s_j and s_k form a 3-clique,

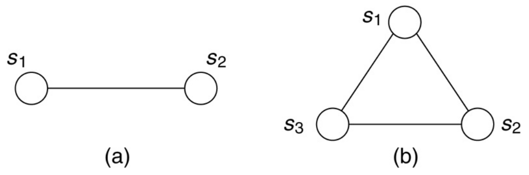

Fig. 3. Sensor-oriented models. (a) The model for the scenario shown in Fig. 2(a). (b) The model for the scenario shown in Fig. 2(b).

the clique compatibility function is defined as

$$\psi_{C_3}(x_i, x_j, x_k) = \prod_{\substack{r \text{ s.t} \\ o_r \in C_3(i,j,k)}} p(y_{i\alpha}, y_{j\beta}, y_{k\gamma} \mid x_i(o_r) = y_{i\alpha}, x_i(o_r) = y_{j\beta}, x_i(o_r) = y_{k\gamma}). \tag{10}$$

Fig. 3(b) shows the sensor-oriented model for the scenario shown in Fig. 2(b).

With each node in the graphical model corresponding to a sensor, the sensor-oriented modeling approach is amenable to implementation on sensor networks. However, when there are targets covered by $k \geq 3$ sensors, the sensors form a k-clique and result in a higher order clique compatibility function defined for the whole clique. Unfortunately, the message-passing algorithms require pairwise compatibility functions, as in (2). If the extensions for message-passing such as node aggregation are applied, the new graph will have a huge node with as many as $(n!)^k$ states and could become intractable. It is more convenient here to develop a description in terms of pairwise compatibilities.

3.2.2. *Target-oriented modeling*

Analogous to the centralized data association techniques where the focus is usually on solving the data association for each target [20], we can formulate the data association problem based on association variables defined for each target, i.e., $\mathbf{X} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_M\}$. For target o_r, $\tilde{\mathbf{x}}_r$ takes value in the Cartesian product of the measurement sets generated by sensors covering o_r. We use $\tilde{x}_r(s_i) = y_{i\alpha}$ to denote that in a particular association configuration $\tilde{x}_r$, target o_r is assigned measurement $y_{i\alpha}$ from sensor s_i. Notice that for a globally valid association configuration, different targets cannot be assigned to the same measurement, i.e., if $\tilde{x}_l$ and $\tilde{x}_r$ are the association for o_l and o_r respectively in a globally valid association configuration, then $\tilde{x}_l(s_i) \neq \tilde{x}_r(s_i)$, $\forall i$. Then the measurement likelihood in a particular association configuration $\{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_M\}$ can be factorized as

$$\begin{aligned}
&p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N \mid \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_M) \\
&\quad = \prod_{i=1}^{N} \left(\prod_{\substack{r \text{ s.t} \\ o_r \in C_1(i)}} p(y_{i\alpha} \mid \tilde{x}_r(s_i) = y_{i\alpha}) \right) \\
&\quad \prod_{\substack{i,j \\ i<j}} \left(\prod_{\substack{r \text{ s.t} \\ o_r \in C_2(i,j)}} p(y_{i\alpha}, y_{j\beta} \mid \tilde{x}_r(s_i) = y_{i\alpha}, \tilde{x}_r(s_j) = y_{j\beta}) \right) \\
&\quad \prod_{\substack{i,j,k \\ i<j<k}} \left(\prod_{\substack{r \text{ s.t} \\ o_r \in C_3(i,j,k)}} p(y_{i\alpha}, y_{j\beta}, y_{k\gamma} \mid \tilde{x}_r(s_i) = y_{i\alpha}, \tilde{x}_r(s_j) = y_{j\beta}, \tilde{x}_r(s_k) = y_{k\gamma}) \right) \\
&\quad \cdots \\
&\quad \prod_{i=1}^{N} \left(\prod_{\substack{l,r \text{ s.t} \\ o_l, o_r \in O_i}} (1 - \delta_i(\tilde{x}_l, \tilde{x}_r)) \right)
\end{aligned} \tag{11}$$

where

$$\delta_i(\tilde{x}_l, \tilde{x}_r) = \begin{cases} 1 & \text{if } \tilde{x}_l(s_i) = \tilde{x}_r(s_i) \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

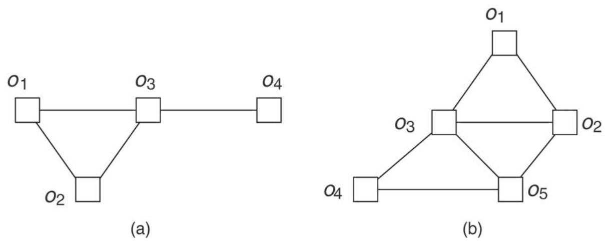

Fig. 4. Target-oriented models. (a) The model for the scenario shown in Fig. 2(a). (b) The model for the scenario shown in Fig. 2(b).

The δ functions here enforce the constraints that a measurement at a sensor can be assigned to only one of the targets covered by the same sensor. These kinds of binary constraints are widely used in graphical models for parity-check codes [25], thereby they are usually referred to as parity-check compatibility functions. The above factorization indicates that with each node in the graph corresponding to a target, the node compatibility function is defined by the likelihood of the measurements associated with the corresponding target. For example, $\psi(\tilde{x}_r) = p(y_{i\alpha} \mid \tilde{x}_r(s_i) = y_{i\alpha})$ if o_r is covered by only s_i; or $\psi(\tilde{x}_r) = p(y_{i\alpha}, y_{j\beta} \mid \tilde{x}_r(s_i) = y_{i\alpha}, \tilde{x}_r(s_j) = y_{j\beta})$ if o_r is covered by only s_i and s_j, etc. Due to interaction defined by the δ functions, we need to connect the target nodes covered by the same sensor with each other, and define the edge compatibility function as

$$\psi(\tilde{x}_l, \tilde{x}_r) = 1 - \delta_i(\tilde{x}_l, \tilde{x}_r). \tag{13}$$

Fig. 4 shows the target-oriented models for the scenarios shown in Fig. 2.

With the target-oriented modeling approach, we can always obtain pairwise compatibility functions even if there are targets covered by more than two sensors. The number of states is n^k for a target covered by k sensors. Compared with sensor-oriented models which have the largest node size of $(n!)^k$ after clustering the clique nodes, this is a significant reduction on node state space. However, the target-oriented modeling approach usually yields a highly connected graph since the nodes for the targets covered by the same sensor form a clique. Besides, the nodes inside the clique are tightly linked by parity-check compatibility functions. The strong interaction between nodes can cause oscillation in the dynamics of local message-passing algorithms, which can slow or prevent convergence.

3.2.3. *Sensor–target hybrid modeling*

We now propose a sensor–target hybrid modeling approach. This hybrid approach allows us to retain the framework of sensor-oriented models (thereby keeping the algorithm close to the network architecture), and simultaneously leads to pairwise compatibility functions with the largest node having n^k states. As in the sensor-oriented models, we build one node for each sensor. If no targets are covered by more than two sensors, we construct the graphical model in exactly same way as the sensor-oriented models. If there are some targets covered by three or more sensors, we incorporate the association variables for these targets into the models as well, and connect each target with every sensor node that covers it. In this way, the clique compatibility function that would occur in sensor-oriented models becomes the node compatibility function for the targets. For example, for the scenario in Fig. 2(b), we can write

$$\begin{aligned} &p(y_{i\alpha}, y_{j\beta}, y_{k\gamma} \mid x_i(o_r) = y_{i\alpha}, x_j(o_r) = y_{j\beta}, x_k(o_r) = y_{k\gamma}) \\ &\quad = p(y_{i\alpha}, y_{j\beta}, y_{k\gamma} \mid \tilde{x}_r(s_i) = y_{i\alpha}, \tilde{x}_r(s_j) = y_{j\beta}, \tilde{x}_r(s_k) = y_{k\gamma})\delta(\tilde{x}_r, x_i)\delta(\tilde{x}_r, x_j)\delta(\tilde{x}_r, x_k) \end{aligned} \tag{14}$$

where

$$\delta(\tilde{x}_r, x_i) = \begin{cases} 1 & \text{if } \tilde{x}_r(s_i) = x_i(o_r) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

is the compatibility function defined on the edge connecting the target node with the corresponding sensor node. This compatibility function ensures that the sensor association variable and target association variable are consistent with each other for forming a valid global association configuration. The hybrid model for this scenario is shown in Fig. 5.

In summary, we construct the sensor–target hybrid models as follows. We first build one node for each sensor to represent the association configurations of this sensor. Two nodes are connected by an edge if they share any target, and no third sensor covers this target. If there is any target shared by more than two sensors, then we build one node

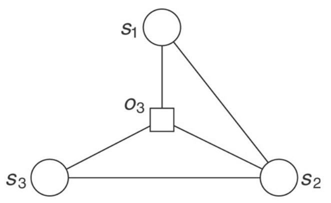

Fig. 5. Sensor–target hybrid model for the scenario shown in Fig. 2(b). The square node corresponds to target o_3 observed by all three sensors.

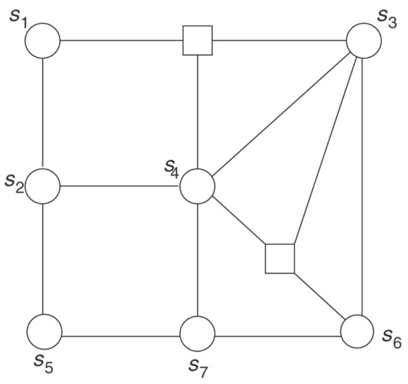

Fig. 6. The sensor–target hybrid model for the example shown in Fig. 1. Circle nodes correspond to the sensors as labeled, square nodes are for the two $\bigtriangledown$ targets observed by more than two sensors.

for each such target and connect it with all the nodes that represent the sensors which cover that particular target. As another example, Fig. 6 shows the graphical model for the scenario in Fig. 1 using the sensor–target hybrid modeling approach.

3.2.4. *False alarms and missed detections*

It is straightforward to extend our approach to deal with data association in an environment where false alarms and missed detections are possible. In this subsection, we discuss how to handle false alarms and missed detections in the sensor–target hybrid modeling approach. Similar techniques can be applied to the sensor-oriented models and target-oriented models.

In the case of false alarms, it is possible that some of the measurements are not assigned to any targets. On the other hand, in the case of missed detections, we need to consider the possibility that a target might not produce any measurement at a sensor covering it. In order to take into account false alarms at sensor s_i, we augment each target set O_i with a virtual target o_0 such that each sensor has its own virtual target, and this virtual target can only be assigned to a measurement generated by the sensor s_i. Similarly, we augment each measurement set $\mathbf{y}_i$ with a virtual measurement y_0 to take into account the targets missed by s_i. We use the notation $x_i(o_0) = y_{i\alpha}$ to denote that measurement $y_{i\alpha}$ is a false alarm in a particular association configuration for sensor s_i. Multiple measurements at a particular sensor could be assigned to the virtual target o_0 due to the fact that multiple false alarms could exist. Similarly, we use the notation $x_i(o_r) = y_0$ to denote that target o_r is missed by sensor s_i in the particular association configuration x_i. Multiple targets covered by s_i could be assigned y_0, corresponding to the fact that multiple targets could be missed. For the sensor nodes, the state space is now the permutation space of the augmented measurement set where y_0 can be used repeatedly. We evaluate the likelihood of false alarm measurements in a particular association configuration through $p(y_{i\alpha} \mid x_i(o_0) = y_{i\alpha})$. We also multiply the sensor node compatibility function with a probability that corresponds to the number of detections and missed detections. For example, if sensor s_i has k missed detections in x_i, and also one measurement $y_{i\alpha}$ is deemed a false alarm, then the node compatibility function is defined as

$$\psi_i(x_i) = \left(\prod_{\substack{r \text{ s.t} \\ o_r \in C_1(i)}} p(y_{i\alpha} \mid x_i(o_r) = y_{i\alpha}) \right) p(y_{i\beta} \mid x_i(o_0) = y_{i\beta}) P_F P_D^{(n-k)} (1 - P_D)^k \tag{16}$$

where P_F and P_D are the false alarm probability and detection probability of the sensor respectively. For the target nodes in the target-oriented models as well as in the sensor–target hybrid models, the state space of the node is the Cartesian product of the augmented measurement sets generated by each sensor covering it. If a particular state for a target node is such that $\tilde{x}_r(s_i) = y_0$, then it means a virtual measurement is assigned to o_r by s_i, i.e., o_r is missed by s_i. The compatibility function between a target node and a sensor node is still the parity-check function in this case, but the value of $\tilde{x}_r(s_i)$ and $x_i(o_r)$ could be the virtual measurement y_0.

3.3. Scalability

One major advantage of using techniques based on graphical models is that they scale well to large-scale sensor networks. With the sensor–target hybrid models, all sensors and some targets have corresponding nodes in the graph. Without false alarms and missed detections, each sensor node has $n!$ states. The target node has n^k states, where k is the number of sensors covering the corresponding target. If the graph is of tree structure, both message-passing algorithms and TRMP have the complexity $\mathcal{O}(\max(n!, n^k)^2(N + M))$ which corresponds to the worst case in which every target is observed by k sensors. For a typical tracking application with $n \approx 5$ and $k \approx 4$, the complexity is about $\mathcal{O}(n^{2k}(N + M))$ (compared to $(n!)^N$ for direct computation), i.e., linear with the number of sensors used in the network. The above analysis can be extended to the scenarios where false alarms and missed detections exist accordingly. Of course, there are loops in the graphical models arising from most sensor networks, so that local message-passing algorithms typically need more iterations (and hence more computation) to converge than in the case of trees. The tree-reweighted max–product (TRMP) algorithm involves running max–product on several spanning trees embedded in the graph. The construction of spanning trees introduces a certain amount of extra computation, but this computation cost is negligible as compared to the inference cost because just a small number of spanning trees are usually enough to cover the original graph. With each tree having the above complexity, TRMP needs to iterate on these spanning trees until agreement is obtained. Although these iterations can incur further computational cost, the benefit is the guarantee of exact MAP estimates. The computation can still be fully distributed over the sensor network, so that the load for each sensor remains reasonable. Consequently, our approach can be used to solve data association problems in large-scale sensor networks. In Section 6, we provide simulated results on problems of moderate size (namely, 25 sensors with up to 54 targets) so as to support this assertion.

4. Target distribution estimation for network self-organization

4.1. Problem formulation

In this section, we consider the problem of *self-organization* in a sensor network, where the goal is to form estimates of how the targets are distributed in the surveillance area. Such estimation results are helpful for determining which targets are seen by which sensors, and thus can be viewed as a preprocessing step that determines the graphical model structure for the data association problem formulated in Section 3. Specifically, we assume that the sensors are proximity indicators that can only detect and indicate the presence of targets within their observation range (hence within their proximity), and we consider a target distribution estimation problem to determine the number of targets covered by each distinct subset of sensors. We consider a planar region where N sensors $\{s_1, s_2, \ldots, s_N\}$ are deployed to monitor the surveillance area. Each sensor s_i has some limited local signal processing capability to generate a noisy observation y_i about the likelihood of the number of targets it detected in its surveillance range. Sensors are randomly distributed in the surveillance area such that part of the coverage area of each sensor overlaps with the coverage area of its neighboring sensors. Thus, there are disjoint *subregions* $\{r_1, r_2, \ldots, r_M\}$, each covered by a distinct set of sensors (see Fig. 7(a) for example). Let R_i denote the set of subregions covered by sensor s_i. We define $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ to be a set of independent random variables, where each $\mathbf{x}_j$ corresponds to the number of targets located in subregion r_j. Let $p(\mathbf{x}_j)$ denote the prior for $\mathbf{x}_j$. Our objective then is to estimate the posterior distribution of the number of targets in each subregion given the sensor observations $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$, i.e., $p(\mathbf{x}_j \mid \mathbf{y})$ for each r_j.

This target distribution estimation problem illustrates a challenge common to many fusion problems using sensor networks. Specifically, the goal is to obtain a global decision based on local, imprecise and redundant information provided by a large number of relatively simple and myopic sensors. Dealing with such problems typically requires a large amount of computation. For example, in the target distribution estimation problem, if the number of targets in

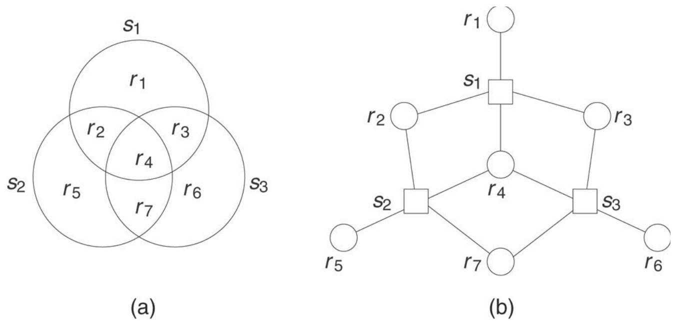

Fig. 7. Sensor–subregion hybrid model for the target distribution estimation problem.

each subregion has n possible values, then with M regions, there are n^M possibilities. For a large sensor network where the surveillance area consists of many subregions, enumerating all possibilities by brute force is usually infeasible. However, graphical models can be used so as to take advantage of the sparsity inherent in the problem structure. For the target distribution estimation problem, the sparsity originates from the fact that each sensor has only limited coverage, and that $\mathbf{x}_j$ is Markov with respect to the graph structures embedded in the sensor coverage topology. In the next subsection, we propose a *sensor–subregion hybrid modeling* approach for constructing the graphical models and use the sum–product algorithm to estimate $p(\mathbf{x}_j \mid y_1, y_2, \ldots, y_N)$.

4.2. Graphical models for target distribution estimation

Let $\mathbf{z} = \{\mathbf{z}_{ij} \mid i = 1, 2, \ldots, N,\ j \text{ s.t. } r_j \in R_i\}$ be a set of auxiliary random variables, with $\mathbf{z}_{ij}$ being the number of detections generated by sensor s_i for subregion r_j, where $r_j \in R_i$. We note that $\mathbf{z}_{ij}$ is not necessarily equal to $\mathbf{x}_j$ since some targets in the subregion might not be detected and some detections might be false alarms. We use $\mathbf{Z}_i$ to denote the set $\{\mathbf{z}_{ij} \mid j \text{ s.t. } r_j \in R_i\}$, and we assume we know $p(y_i \mid \mathbf{Z}_i)$, then the marginal distribution $p(\mathbf{x}_j \mid y_1, y_2, \ldots, y_N)$ can be written as

$$\begin{aligned} p(\mathbf{x}_j \mid y_1, y_2, \ldots, y_N) &= \kappa \sum_{\substack{\mathbf{y},\mathbf{z} \text{ s.t} \\ \mathbf{x}_j = x_j}} p(\mathbf{y} \mid \mathbf{z}, \mathbf{x}) p(\mathbf{z} \mid \mathbf{x}) p(\mathbf{x}) \\ &= \kappa \sum_{\substack{\mathbf{y},\mathbf{z} \text{ s.t} \\ \mathbf{x}_j = x_j}} p(\mathbf{y} \mid \mathbf{z}) p(\mathbf{z} \mid \mathbf{x}) p(\mathbf{x}) \\ &= \kappa \sum_{\substack{\mathbf{x},\mathbf{z} \text{ s.t} \\ \mathbf{x}_j = x_j}} \left(\prod_{i=1}^{N} p(y_i \mid \mathbf{Z}_i) \right) \left(\prod_{j=1}^{M} p(\mathbf{x}_j) \right) \left(\prod_{\substack{(i,j) \text{ s.t} \\ r_j \in R_i}} p(\mathbf{z}_{ij} \mid \mathbf{x}_j) \right). \end{aligned} \tag{17}$$

The above factorization indicates that our graph contains sensor nodes and subregion nodes. A sensor node and a subregion node are connected if and only if that particular sensor covers the corresponding subregion. For sensor nodes, the node compatibility function is defined as $\psi_{s_i}(\mathbf{Z}_i) = p(y_i \mid \mathbf{Z}_i)$. For the subregion nodes, the node compatibility function is defined as $\psi_{r_j}(\mathbf{x}_j) = p(\mathbf{x}_j)$. For the edge connecting s_i and r_j such that $r_j \in R_i$, the edge compatibility function is defined as

$$\psi_{s_i r_j}(\mathbf{z}_{ij}, \mathbf{x}_j) = p(\mathbf{z}_{ij} \mid \mathbf{x}_j), \tag{18}$$

where $p(\mathbf{z}_{ij} \mid \mathbf{x}_j)$ is a function of the false alarm rate and missed detection rate of sensor s_i.[2] As an example, Fig. 7(b) shows the sensor–subregion hybrid model for the scenario shown in Fig. 7(a).

For a network with N sensors and M subregions, the graphical model we obtain has $N + M$ nodes. Each state of a subregion node corresponds to the number of targets in the subregion, and each state of a sensor node corresponds

[2] When there are no false alarms and missed detections, the number of targets detected by the sensor in each subregion is exactly the true number of targets in that subregion, then the edge compatibility functions are reduced to parity-check functions.

to a q-tuple of the detections in each subregion that the sensor covers. With each subregion having n states and each sensor covering k subregions, the sensor nodes have n^k states if there are no false alarms and missed detections. The sum–product algorithm has the complexity of $\mathcal{O}(n^{2k}(N+M))$ on tree-structured graphs. With its scalability with respect to the number of sensors in use, the approach proposed here can also be applied to other multi-sensor fusion applications.

5. Communication-sensitive message-passing

In a distributed implementation, message-passing operations require communication between the sensor nodes in the network. Although the parallel message-passing operation is inherently a distributed algorithm and hence appealing for sensor network applications, the potentially unbounded amount of communication incurred by this procedure poses a significant challenge. With the parallel message-passing technique, each node is required to send a message to each of its neighbors at every iteration. In current sensor technologies, power is a limited resource, and power consumption is dominated by the cost of communication. Consequently, for local message-passing algorithms to be broadly applicable in sensor networks, it is critical to reduce the amount of communication that they require.

One approach to reducing communication costs is simply to stop the algorithm early, prior to convergence. This naive approach can fail, as it is likely that important information may never be generated or transmitted. Consequently, we propose an adaptive approach that, while reducing the amount of communication, does *not* lead to serious degradation in performance. In this approach, after a new message is formed at a node, the node has the authority to make a decision about whether it needs to transmit this message or not. A message will be sent only when it contains “significant” new information compared to the message sent by the same node on the same edge in the previous iteration; otherwise the message will not be sent. If the message in the current iteration is not sent, the destination node uses the corresponding message from the previous iteration instead. In other words, during every round of message-passing after the message M_{ts}^n is generated, each node t computes $d(M_{ts}^n, M_{ts}^{n-1})$ according to a certain distance measure $d(\cdot,\cdot)$ and compares it with a message tolerance ϵ. If $d(M_{ts}^n, M_{ts}^{n-1}) < \epsilon$, message M_{ts}^n will not be sent, and node s will use M_{ts}^{n-1} that it already received in the previous iteration to do its own local computation. Integrating this technique into standard message-passing algorithms leads to a new class of communication-sensitive message-passing (CSMSG) algorithms. In particular, we obtain the communication-sensitive sum–product (CSSP) algorithm in the sum–product setting and the communication-sensitive max–product (CSMP) algorithm in the max–product setting. With CSMSG, a trade-off arises between the performance the algorithm can achieve and the amount of communication it requires. By using a proper message tolerance ϵ, we can tune the algorithm to achieve a suboptimal solution according to the budget for the communication cost. With smaller ϵ, the algorithm obtains a more accurate approximation to the value computed by standard message-passing. However, with smaller ϵ the amount of communication saving is decreased, since more messages exceed the message tolerance and are transmitted. Yet even with very small ϵ such that the loss of performance is trivial, the communication saving compared with the standard message-passing might still be significant. In Section 6, we show the tremendous communication saving of CSMSG compared with standard message-passing.

One key observation for justifying the effectiveness of CSMSG is that the message M_{ts}^n can be interpreted as a sufficient statistic of $y_{t\setminus s}^n$, the set of data in the subtree rooted at node t and separated from t in graph $\mathcal{G}$ by at most n edges. Thus, M_{ts}^n differs from M_{ts}^{n-1} in that M_{ts}^n contains the information that M_{ts}^{n-1} already contains as well as the information collected by nodes that are exactly n edges away. However, for sensor network applications, the neighboring nodes in graphs usually correspond to sensors that are deployed close to each other and thus collect similar information. Consequently, many messages in two consecutive iterations are similar and the new messages need not be sent. The fact that the messages are statistics also suggests that we can use the Kullback–Leibler (KL) divergence [26] to measure the distance between the information contents of two messages. The KL divergence is widely used to measure the similarity of two probability distributions in the information theory literature. In this case, it is defined as

$$d(M_{ts}^n, M_{ts}^{n-1}) = \sum_{\mathbf{x}_s} M_{ts}^n \log \frac{M_{ts}^n(\mathbf{x}_s)}{M_{ts}^{n-1}(\mathbf{x}_s)}. \tag{19}$$

Besides the communication savings gained by not transmitting messages below tolerance, the CSMSG updates usually converge much faster than standard message-passing and thus reduces the amount of communication further, especially on loopy graphs (where standard message-passing may fail to converge). Note that CSMSG converges at the point when no nodes are active in sending messages. This is equivalent to relaxing the convergence criterion for standard message-passing, allowing early stopping when all nodes are satisfied that early stopping will not lead to loss of crucial information. Furthermore, the convergence behavior of CSMSG suffers less from the effect of *rumor propagation*, which could result in a slow convergence rate as messages travel around cycles and are no longer independent of each other. In particular, a message is sent out in CSMSG only when it contains enough (decided by the message tolerance ϵ) new information compared with the old message. Empirically, we find that this stopping criterion helps to suppress the effect of cycles, and can improve the convergence rate.

The overhead for implementing CSMSG instead of standard message-passing is insignificant considering the potential savings in communication. In CSMSG, every sensor node requires some additional memory for storing messages from the previous iteration, so that it can compare the new messages (that it generated) with the old messages (that it generated), and use the old messages (that it received) when necessary. In addition, we also require a mechanism for letting the sensor know when a new message has not been sent, so that it needs to use the old message instead. One possibility is to pass one extra bit of information on every link in each iteration to indicate whether the nodes have new information or not. Alternatively, we could synchronize the communication and design the protocol in such a way that the sensor will use the old message after some latency period, whether the new message was not sent or was simply lost on the way.

6. Experimental results

6.1. Data association in an organized network

We have tested our approach to the data association problem formulated in Section 3 on simulated data. Our set-up is a surveillance system consisting of 25 sensors, forming a 5×5 grid in the 2D plane. Each sensor measures the 2D position of targets in its detection range. All sensors have the same detection radius $r = 5$. The targets are uniformly distributed in the surveillance area. Their prior position distributions are assumed available, as for example acquired from the prediction step of a tracking algorithm.

We use a Gaussian distribution around the true position of a target as the predicted position distribution. The covariance is fixed as $\Sigma = \sigma^2 I$ with $\sigma^2 = 5$. The measurements are corrupted by independent Gaussian noise of zero mean and covariance $\Omega = \omega^2 I$. Each sensor has a detection rate of $P_D = 0.8$. The number of false alarms at each sensor is randomized to zero with probability 0.85 and one with probability 0.15. We adopt the sensor–target hybrid modeling approach for this experiment. We have conducted our approach with various target density and measurement noise levels. As the target density increases from two targets, three targets to four targets per sensor, the total number of targets in the whole area increases from 17, to 33, to 54.

Fig. 8(a) shows the association error rate achieved by the tree-reweighted max–product (TRMP) algorithm. The association error rate is calculated as the percentage of measurements that are associated with wrong targets. Each data point is the average result of 50 trials. The results are optimal in the MAP sense. The plots clearly show that with more intensified contention for measurements and with increasing measurement noise, the association error goes up. Fig. 8(b) shows the corresponding amount of communication required by the TRMP algorithm. The amount of communication is evaluated as the total number of messages sent until TRMP converges normalized by the number of sensors. In the high signal-to-noise ratio region (i.e., $\omega^2/r^2 \leq 0.15$), the amount of communication goes up monotonically as the measurement noise increases and the target density increases. However, when the measurement noise becomes too large, the amount of communication dips. This behavior is intuitively reasonable, since the measurements are getting closer and closer to other targets, so that a wrong association is more likely to be achieved with less communication when the measurements become unreliable.

We also investigate the trade-off between the amount of communication and performance by applying the standard max–product (MP) algorithm and communication-sensitive version of max–product (CSMP). We use the message tolerance based on the KL divergence as introduced in Section 5. Table 1 compares the performance achieved and a communication needed by TRMP, the standard MP and CSMP. The data are the average results obtained from 50 trials on the scenarios corresponding to three targets per sensor case and $\omega^2/r^2 = 0.2$ in Fig. 8. The error rate generated by the TRMP algorithm is optimal in the MAP sense. Table 1 shows that the standard MP algorithm and the CSMP algorithm with reasonable message thresholds have slightly higher error rates and can achieve near-optimal

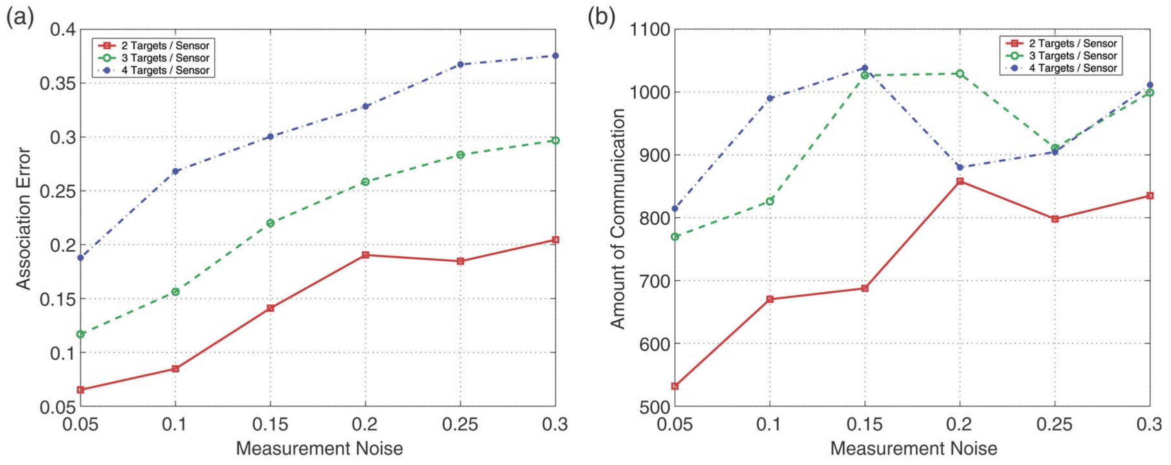

Fig. 8. Data association at various target density and measurement noise levels with TRMP. (a) Association error rate versus the relative measurement noise variance ω^2/r^2. The standard deviation for each point is around 0.075. (b) Amount of communication per sensor versus the relative measurement noise variance. The standard deviation for each point is around 400.

Table 1
Performance achieved and communication needed for data association

Algorithm	TRMP	Standard MP	CSMP, $\epsilon = 0.1$	CSMP, $\epsilon = 1$
Error	25.84%	25.89%	27.88%	28.95%
Communication	1029.3	55.25	10.41	8.79

performance. Although all three algorithms yield similar error rates in this case, there is a huge difference between the communication costs required by the three algorithms. The TRMP algorithm consumes much more communication than the other two, but is guaranteed to output the correct association. In contrast, both the standard max–product algorithm and CSMP produce only approximate estimates on loopy graphs, but entail a lower communication cost than TRMP. The communication cost associated with the CSMP updates is significantly less than that with the standard max–product algorithm. Therefore, when communication is costly, CSMP is a preferable algorithm in that it can achieve a near-optimal accuracy with far less communication (with the appropriate choice of tolerance parameter). The trade-off curves for CSMP at various message tolerances are shown in Fig. 9. It clearly shows that an interesting threshold exists around $\epsilon = 1.7$. With smaller message tolerances than this threshold, we can achieve a similar error rate as the standard max–product but with much less communication. However, when the message tolerance exceeds this threshold, the error rate increases sharply with a further drop in communication. The error rate quickly rises close to the error rate when we treat each node independently and generate the estimates based on local node compatibility functions only. This reflects the fact that for a message tolerance above the threshold, some messages that are crucial for obtaining a reliable estimation are ignored. The existence of the threshold also suggests that the message tolerance corresponding to it might be an ideal parameter when we want to pursue the best performance–communication cost ratio. However, how to identify this message tolerance in advance remains an open question.

With CSMP, we can show the information flow dynamics by displaying the message transmission in each iteration. The information flow for one of the problems ($\epsilon = 0.1$) in Fig. 9 is shown in Fig. 10. In the first iteration, every node sends messages to its neighbors to initialize the iteration. As the iteration goes on, fewer and fewer nodes need to transmit messages, and only one node sends a message in the last iteration. The nodes that have strong local information or strong interaction with their neighbors usually need to send more messages in this whole process, as we see in the case of nodes s_7 and o_4 in Fig. 10. On the other hand, the nodes located in a cluster of nodes that have consistent observations with each other usually stop passing messages earlier, for example, the five nodes in the upper left corner. This observation suggests that the corresponding sensors can be shut off earlier for more power saving. However, while the five nodes at the upper left corner can be shut off permanently after the second iteration, there are other nodes (e.g., node s_{12} and node o_7) which can be shut off temporarily and need to start communication again later.

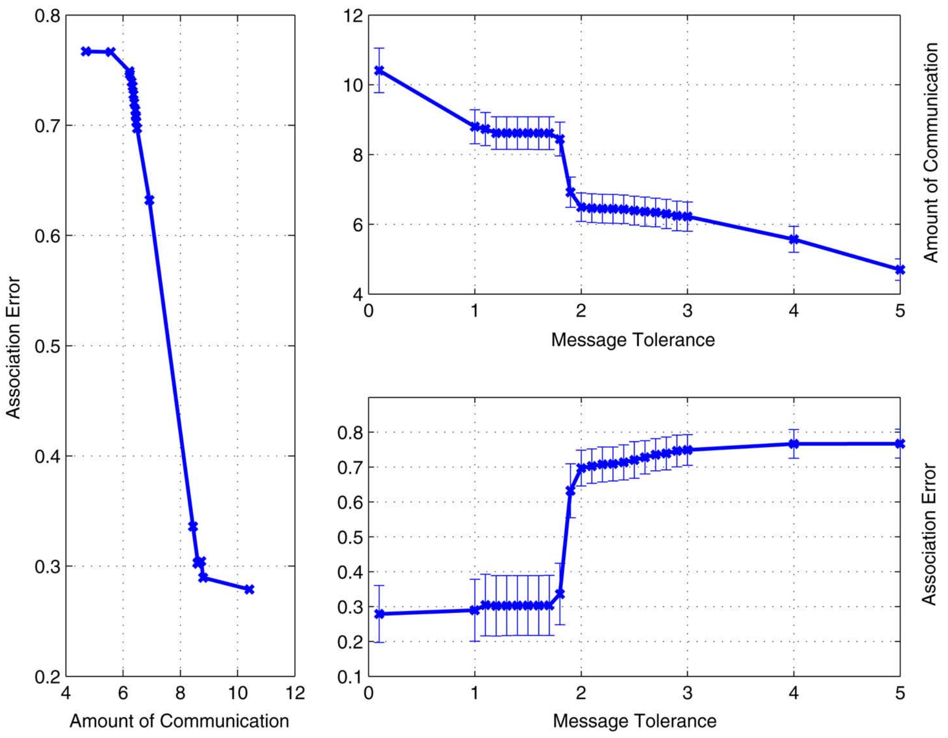

Fig. 9. Trade-off between communication and performance for the data association problem using CSMP. The length of the error bar at each data point shows two times the standard deviation. With $\epsilon \leq 1.7$, the association error rate keeps near-optimal although the amount of communication is far less than the amount required by TRMP and standard MP. The error rate increases sharply when $1.7 < \epsilon < 2$, where some crucial messages are not transmitted.

In particular, as the information from the cluster of nodes around s_7 and o_4 arrives, node s_{12} resumes sending messages in iteration five because it may find its previous estimate is inaccurate or the incoming information is important to its neighbors. The message from s_{12} eventually wakes up o_7 in iteration six.

6.2. *Target distribution estimation for network self-organization*

We have carried out experiments for target distribution estimation using simulated data. The sensors are proximity indicators, each of which generates a noisy likelihood function for the number of targets in its surveillance region. Sensors are randomly deployed into a unit surveillance area, thereby dividing the surveillance area into disjoint subregions according to the overlapping coverage. In each subregion, a random number of targets are placed according to a prior distribution. The prior of $\mathbf{x}_i$, the number of targets in each subregion, is of Poisson distribution with mean $\lambda_i \propto a_i$, where a_i is the area of the subregion. We assume that we know the number of targets in each subregion within an accuracy of ± 2 of the true number of targets.

We enumerate the possible numbers of targets that the sensor could detect in each subregion and use them as the states of the auxiliary variables $\mathbf{z}_{ij}$. The detection rate of each sensor is set to $P_D = 0.8$, and the number of false alarms generated by sensor s_i for r_i obeys a certain Poisson distribution with mean proportional to the area of the subregion. With $\mathbf{Z}_i = \{\mathbf{z}_{ij}\}$, each sensor produces an observation according to $p(y_i \mid \mathbf{Z}_i) = \mathcal{N}(y_i; \sum_j \mathbf{z}_{ij}, \sigma^2)$, where $\sigma = 1$. Sensor–subregion hybrid models are constructed and the sum–product algorithm is applied to obtain the marginal distribution of $\mathbf{x}_i$. To evaluate the performance of the estimation, we pick $\hat{\mathbf{x}}_i = \arg\max_{x_i} p(x_i \mid \mathbf{y})$ after we obtain $p(x_i \mid \mathbf{y})$, and the error rate is calculated as

$$\mathrm{Err} = \frac{\sum_{i=1}^{M} \frac{|\hat{\mathbf{x}}_i - \bar{x}_i|}{n}}{M}, \tag{20}$$

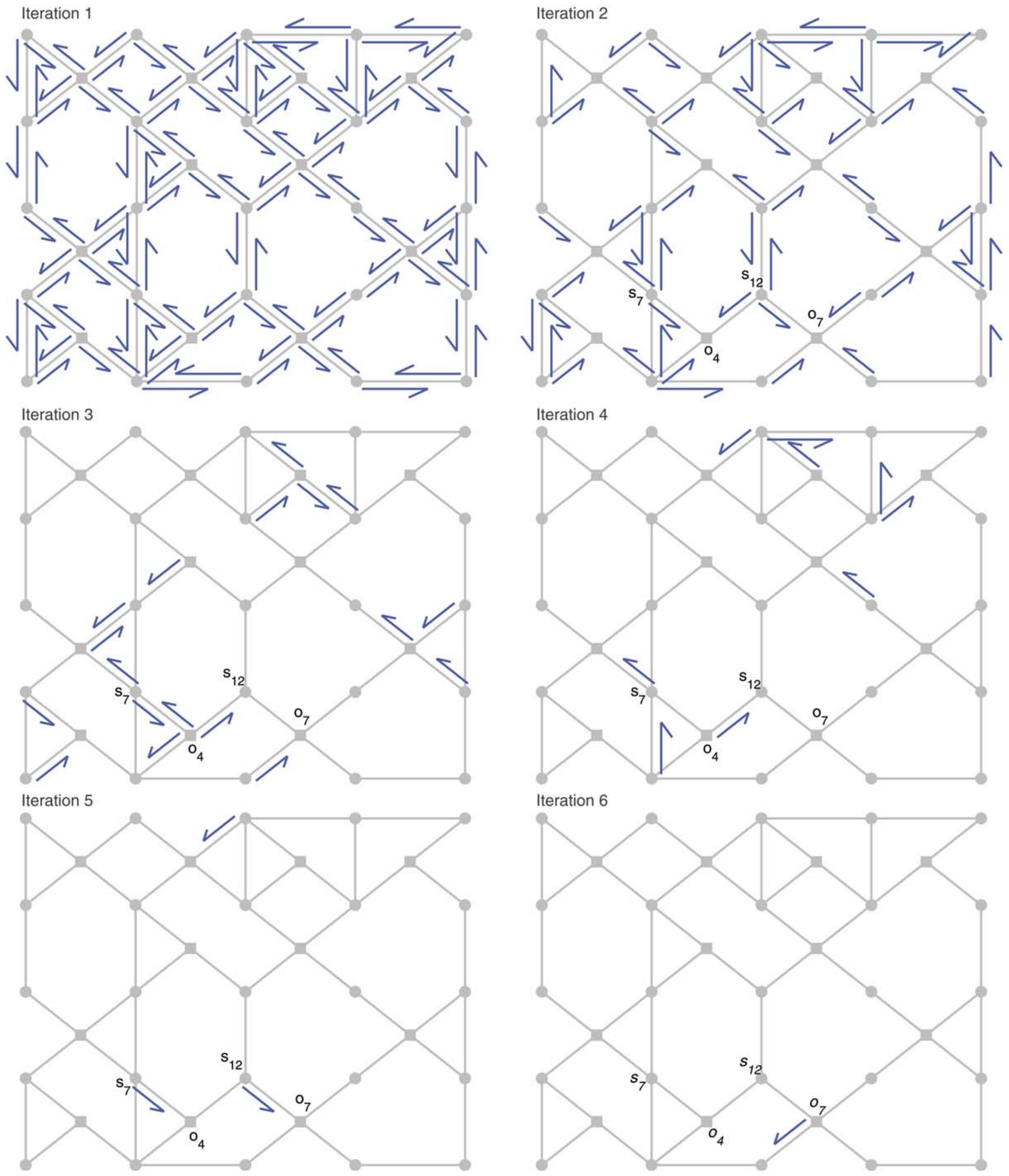

Fig. 10. Information flow dynamics for a data association problem using CSMP. Circle nodes correspond to sensors, and square nodes correspond to targets covered by more than two sensors. The arrows adjacent to edges indicate active communication links and message direction. Sensor s_{12} resumes sending a message in iteration five after it was shut off in iteration three. Consequently target node o_7 is woken up in iteration six.

where $\bar{x}_i$ is the true number of targets in subregion r_i, n is the number of states for each $\mathbf{x}_i$, and M is the total number of subregions.

Fig. 11(a) and (b) show the estimation error achieved and the communication cost required by the sum–product algorithm at various sensor densities, respectively. The amount of communication is evaluated as the total number of messages sent until sum–product converges normalized by the number of sensors. Each data point is the average

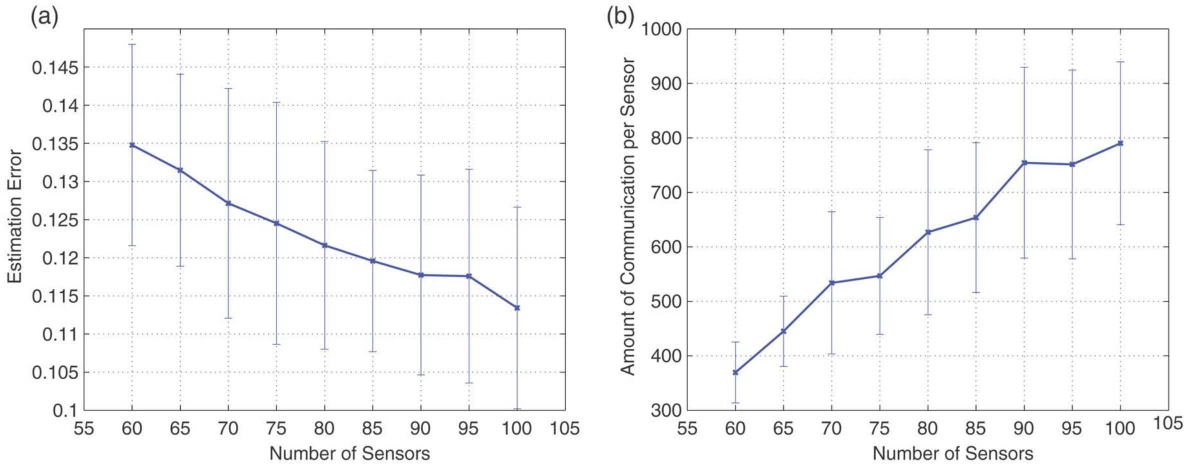

Fig. 11. Error rate and amount of communication for target number estimation using sum–product. The length of the error bar at each data point shows two times the standard deviation.

Table 2
Performance achieved and communication needed for target distribution estimation

Algorithm	Standard SP	CSSP, $\epsilon = 0.1$	CSSP, $\epsilon = 1$	CSSP, $\epsilon = 1.2$
Error	13.48%	12.64%	12.79%	13.62%
Communication	369.56	14.65	14.38	14.28

result of 50 trials. As the number of sensors increases, the number of subregions increases from 94 to 100. The plots show that with increasing number of sensors, we can achieve more accurate estimation. However, the amount of communication spent by each sensor increases accordingly.

Table 2 compares the performance achieved and the communication required by the standard sum–product (SP) algorithm and the communication-sensitive sum–product (CSSP) algorithm. The data corresponds to that in Fig. 11 when 60 sensors are used. With $\epsilon \leq 1.2$, the two algorithms have similar error rates but CSSP spends much less communication per sensor. One interesting phenomenon we have observed is that with very small message tolerances (for example, $\epsilon = 0.1$), CSSP yields smaller error rates than the standard sum–product algorithm, even though CSSP requires less communication. This prompts the question of whether the communication-sensitive versions of message-passing algorithms have the potential to achieve more accurate estimates by preventing rumor propagation. Fig. 12 shows the trade-off between communication and performance when various message tolerances are used on target number estimation with 60 sensors. When $\epsilon > 1.1$, the error rate increases dramatically. This reflects a critical point where further decrease of communication will result in loss of information that is crucial to accurate estimation. Fig. 13 shows the information flow dynamics for the target distribution estimation problem of 25 subregions and 20 sensors with $\epsilon = 0.01$. Notice that s_8 and s_9 cover the same four subregions. These two sensor nodes together with the four subregion nodes that they connect form a cluster of strong interaction. In the first three iterations, these six nodes actively send messages to each other and their influence propagates farther and farther. Even after they stop sending messages, the node s_3 is still passing the information from the cluster to nodes farther away. The node s_{17} reveals an oscillating pattern that frequently occurs in graphical models with parity-check-like compatibility functions. In the third iteration node s_{17} only receives messages and in the next iteration it only sends messages. This pattern is a result of the disagreement among the four nodes linked by s_{17}. Many more iterations are usually required for the standard message-passing updates to settle down in such situations. However, as shown in Fig. 13, the relaxed criterion of CSSP brings the nodes to harmony more swiftly.

7. Conclusion

In this paper, we introduced techniques using the framework of graphical models to solve data association problems arising in distributed sensing scenarios. We proposed several different approaches to modeling, in which nodes in

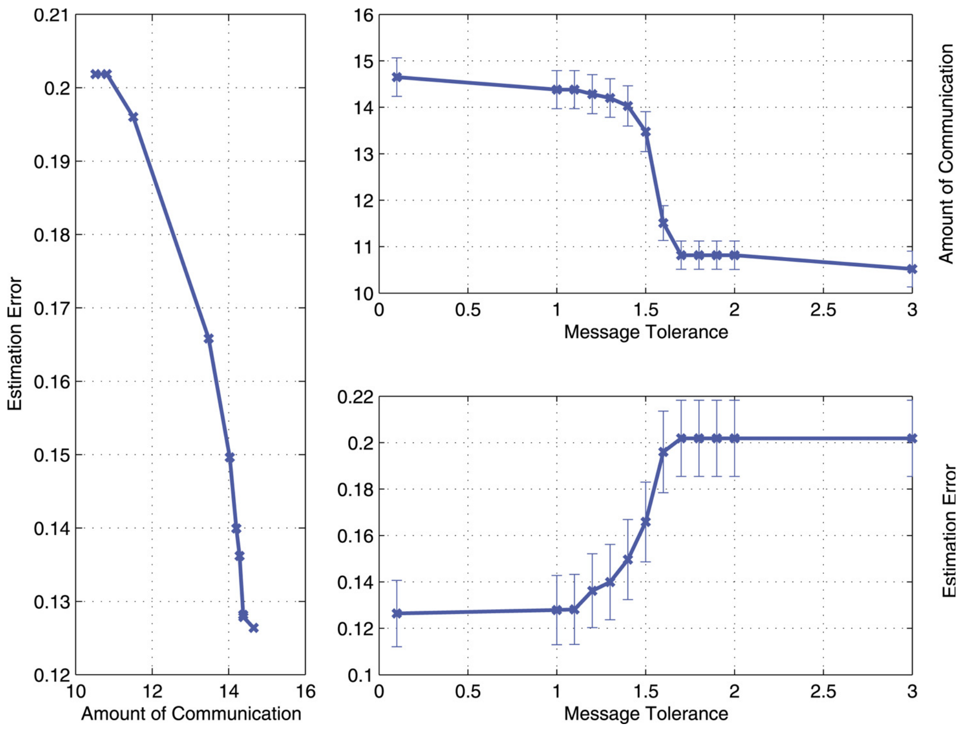

Fig. 12. Trade-off between communication and performance for target distribution estimation using CSSP with various message tolerances ϵ. The length of the error bar at each data point shows two times the standard deviation. When $\epsilon \leq 1.1$, CSSP achieves comparable performance to the standard sum–product algorithm but with far less amount of communication. In the range $1.1 < \epsilon < 1.7$, the error rate increases sharply as some crucial messages are ignored.

the underlying graphical model were associated with different quantities in the sensor network. For instance, for the measurement-to-target data association problem in an organized network, we constructed graphical models by mapping sensors or targets to nodes. However, we also showed that new sensor–target hybrid models are better suited to applying local message-passing algorithms in sensor networks, as they exploit the structure and sparsity inherent to these data association problems. For the target distribution estimation problem, we developed graphical models with nodes corresponding to sensors and surveillance subregions. These data association and estimation problems can be solved efficiently by using local message-passing algorithms, and a suitably reweighted form of the max–product algorithm renders it possible to obtain the optimal (in the sense of maximum a posteriori) data association. Our methods scale well with the number of sensors in the network and are inherently distributed, which renders them well suited for application to distributed inference problems in large-scale sensor networks. We also proposed communication-sensitive versions of local message-passing, and found that they are capable of achieving near-optimal performance with a substantial saving in communication cost. Moreover, we found that consideration of a communication-sensitive version of message-passing yielded insight into the dynamics of the message-passing used to carry out information fusion. Experimental results based on simulated data show the effectiveness of our approach.

There are number of research directions that remain to be explored. First, it is of considerable interest to integrate our graphical model-based approach into tracking algorithms. Doing so requires addressing several issues that arise in the dynamic setting. For example, the uncertainty associated with the subset of targets seen by a given sensor leads to uncertainty in the graph structure of a partially organized network. Following the same idea as the deferred decision of multiple hypothesis tracking, we may keep several most probable graphical structure candidates, and wait for the future data to help resolve ambiguity in model structure. Doing so would require connecting models from several time frames so as to form a dynamic Bayesian network (DBN) [27]. Another possible extension of the dynamic setting is to the case of mobile sensors. Second, the relation between the graphical model and the underlying physical sensor network needs to be explored. Our work shows that the graphical model structure need not be the same as the topology of the physical sensor network. A node in a graphical model may correspond to a concept other than a sensor (for example, a target,

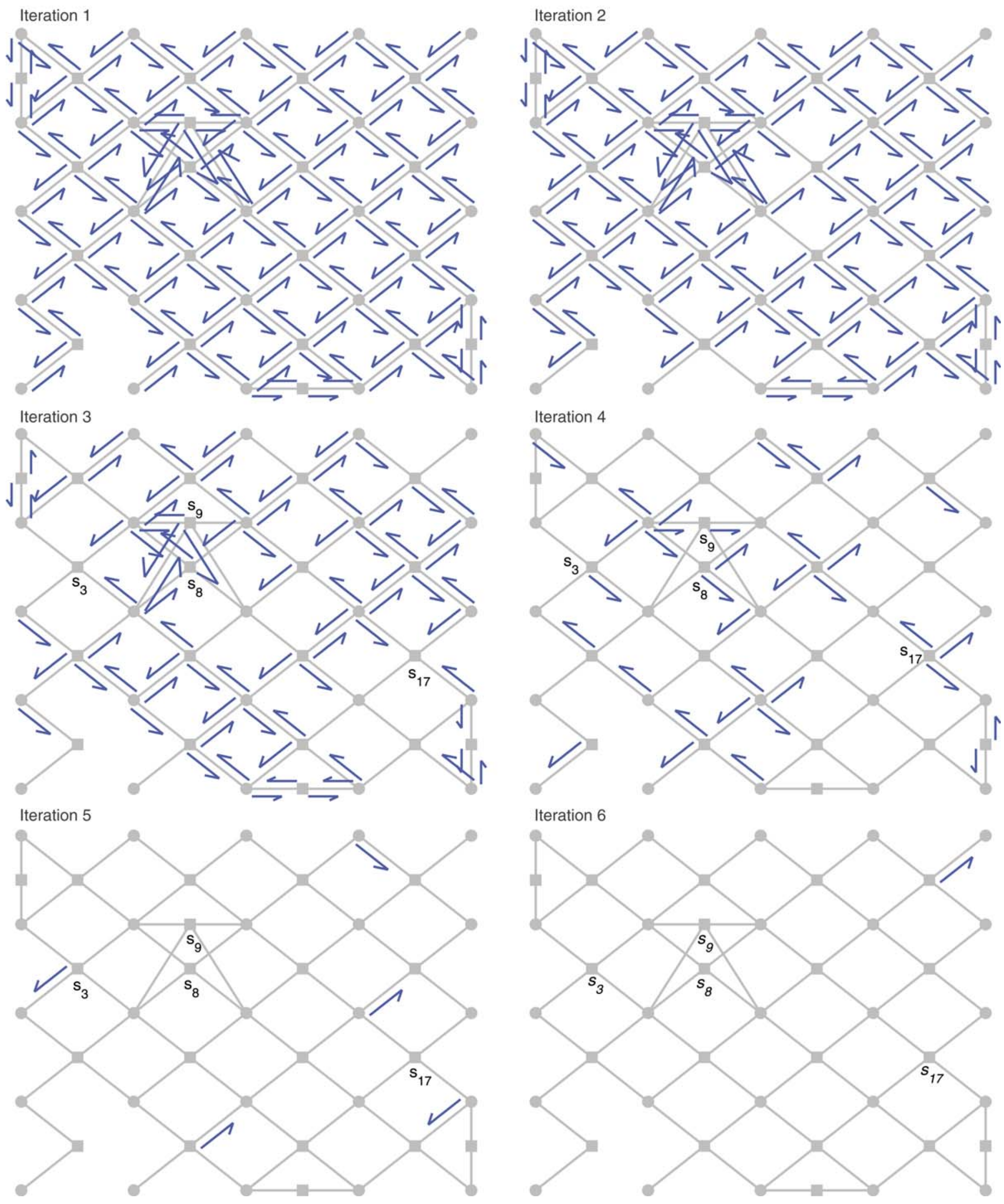

Fig. 13. Information flow dynamics for a target number estimation problem of 25 subregions and 20 sensors. Circle nodes represent subregions. Square nodes represent sensors. The sensors s_8 and s_9 overlap exactly with each other in terms of the coverage area and form a strong interaction cluster. The sensor s_{17} only receives messages in iteration three and only sends messages in iteration four. This oscillating pattern occurs frequently in graphical models with parity-check-like compatibility functions.

or a subregion). Messages may need to be transmitted between two nodes that have no direct communication links. Moreover, the target nodes in a sensor–target hybrid model need to be handed off dynamically among sensors. Finding the structure of graphical models for a mobile sensor network remains to be explored. Consequently, the protocol and a routing algorithm for implementing message-passing in a particular sensor network architecture constitute an interesting topic for further research. Third, it is of interest to provide more theoretical analysis of the communication-sensitive message-passing algorithms. An interesting open problem is how to identify in advance the trade-off between the performance achieved and the communication (or computation) saving. In addition, the CSMSG algorithm that we proposed in this work is only a simple way to address the communication challenge for sensor network applications; more advanced algorithms will be of interest.

Acknowledgments

This work was supported by the Air Force Office of Scientific Research under Grant F49620-01-1-0161 through Subaward Z835101 from the University of Maryland, and by the Army Research Office under Grant DAAD19-00-1-0466. We would like to thank Alex Ihler for helpful discussions on communication-sensitive versions of message-passing.

References

[1] C.Y. Chong, S. Kumar, Sensor networks: evolution, opportunities, and challenges, Proc. IEEE 91 (August) (2003) 1247–1256.
[2] F. Zhao, L. Guibas (Eds.), Information Processing in Sensor Networks, Springer-Verlag, Berlin, 2003.
[3] Y. Bar-Shalom, T.E. Fortmann, Tracking and Data Association, Academic Press, Orlando, 1988.
[4] S.S. Blackman, Multiple-Target Tracking with Radar Applications, Artech House, Dedham, MA, 1986.
[5] D.B. Reid, An algorithm for tracking multiple targets, IEEE Trans. Automat. Control AC-24 (1979) 843–854.
[6] C.Y. Chong, S. Mori, K.C. Chang, Distributed multitarget multisensor tracking, in: Y. Bar-Shalom (Ed.), Multitarget–Multisensor Tracking: Advanced Applications, vol. 1, Artech House, Norwood, MA, 1990, pp. 247–295.
[7] M.E. Liggins, C.Y. Chong, I. Kadar, M.G. Alford, V. Vannicola, S. Thomopoulos, Distributed fusion architectures and algorithms for target tracking, Proc. IEEE 85 (1997) 95–107.
[8] M.I. Jordan (Ed.), Learning in Graphical Models, MIT Press, Cambridge, MA, 1999.
[9] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, 1999.
[10] W.T. Freeman, E.C. Pasztor, O.T. Carmichael, Learning low-level vision, Int. J. Comput. Vis. 40 (October) (2000) 24–57.
[11] L. Rabiner, B.H. Juang, A tutorial on hidden Markov models, Proc. IEEE 77 (1989) 257–286.
[12] R.J. McEliece, D.J. Mckay, J.F. Cheng, Turbo decoding as an instance of Pearl's belief propagation algorithm, IEEE J. Sel. Commun. 16 (February) (1998) 140–152.
[13] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufman, San Mateo, 1988.
[14] K.C. Chou, A.S. Willsky, A. Benveniste, Multiscale recursive estimation, data fusion, and regularization, IEEE Trans. Automat. Control 39 (March) (1994) 464–478.
[15] Y. Weiss, Correctness of local probability propagation in graphical models with loops, Neural Comput. 12 (2000) 1–41.
[16] W.T. Freeman, Y. Weiss, On the optimality of solutions of the max–product belief propagation algorithm in arbitrary graphs, IEEE Trans. Inform. Theory 47 (2001) 736–744.
[17] M.J. Wainwright, T.S. Jaakkola, A.S. Willsky, Tree consistency and bounds on the max–product algorithm and its generalizations, Stat. Comput. 14 (April) (2004) 143–166.
[18] M.J. Wainwright, T.S. Jaakkola, A.S. Willsky, Tree-based reparameterization framework for analysis of sum–product and related algorithms, IEEE Trans. Inform. Theory 49 (May) (2003) 1120–1146.
[19] M.J. Wainwright, T.S. Jaakkola, A.S. Willsky, MAP estimation via agreement on (hyper)trees: message-passing and linear programming approaches, in: Proc. Allerton Conference on Communication, Control and Computing, Monticello, IL, October 2002.
[20] T. Kurien, Issues in the design of practical multitarget tracking algorithms, in: Y. Bar-Shalom (Ed.), Multitarget–Multisensor Tracking: Advanced Applications, vol. 1, Artech House, Norwood, MA, 1990, pp. 43–83.
[21] P. Brémaud, Markov Chains, Gibbs Fields, Monte Carlo Simulations, and Queues, Springer-Verlag, 1991.
[22] J. Yedidia, W.T. Freeman, Y. Weiss, Generalized belief propagation, in: Advances in Neural Information Processing Systems, vol. 13, MIT Press, Cambridge, MA, 2001, pp. 689–695.
[23] S.M. Aji, R.J. McEliece, The generalized distributive law, IEEE Trans. Inform. Theory 46 (March) (2000) 325–343.
[24] F.R. Kschichang, B.J. Frey, H. Loeliger, Factor graphs and the sum–product algorithm, IEEE Trans. Inform. Theory 47 (February) (2001) 498–519.
[25] R.G. Gallager, Low-Density Parity Check Codes, MIT Press, Cambridge, MA, 1963.
[26] T.M. Cover, J.A. Thomas, Elements of Information Theory, John Wiley, New York, 1991.
[27] Z. Ghahramani, Learning dynamic Bayesian networks, in: C.L. Giles, M. Gori (Eds.), Adaptive Processing of Sequences and Data Structures, Springer-Verlag, Berlin, 1998, pp. 168–197.