*Communications in Statistics—Theory and Methods*, 42: 56–75, 2013

ISSN: 0361-0926 print/1532-415X online
DOI: 10.1080/03610926.2011.573157

Taylor & Francis
Taylor & Francis Group

# A New Inference Framework for Dependency Networks

FENG CHEN[1], QIANG CHENG[1], HONG LIU[1], WENLI XU[1], AND SONG WANG[2]

[1]Department of Automation, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China
[2]Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina, USA

*Dependency networks (DNs) have been receiving more attention recently because their structures and parameters can be easily learned from data. The full conditional distributions (FCDs) are known conditions of DNs. Gibbs sampling is currently the most popular inference method on DNs. However, sampling methods converge slowly and it can be hard to diagnose their convergence. In this article, we introduce a set of linear equations to describe the relations between joint probability distributions (JPDs) and FCDs. These equations provide a novel perspective to understand reasoning on DNs. Based on these linear equations, we develop both exact and approximate algorithms for inference on DNs. Experiments show that the proposed approximate algorithms can produce effective results by maintaining low computational complexity.*

**Keywords** Approximate inference; Dependency network; Full conditional distribution.

**Mathematics Subject Classification** 62F30; 62F15.

## 1. Introduction

Graphical models such as Bayesian networks (Pearl, 1988), Markov random fields (Pearl, 1988; Smyth, 1997), and factor graphs (Kschischang et al., 2001) have been widely used to describe multivariate stochastic systems. Probabilistic inference on graphical models has many applications in the fields of statistics, artificial intelligence and pattern recognition. In recent years, many new algorithms have been developed for computing marginal or posterior probabilities on such graphics models. An incomplete list of the reasoning algorithms includes mean field (Peterson and Anderson, 1987), loopy belief propagation (Pearl, 1988), variable elimination (Zhang and Poole, 1994), junction tree propagation (Lepar, 1998), cluster variation

Received March 20, 2010; Accepted March 8, 2011
Address correspondence to Feng Chen, Department of Automation, Tsinghua University, Beijing 10084, China; E-mail: chenfeng@mail.tsinghua.edu.cn

method (Pelizzola, 2005), loop corrections (Mooij and Kappen, 2007), expectation propagation (Minka, 2001; Ypma and Heskes, 2005), free energy (Heskes et al., 2005; Yedidia et al., 2005), and the Monte Carlo Markov Chain (MCMC) method (Neal, 1993).

First introduced by Heckerman et al. (2001), dependency networks (DNs) have been attracting more attention recently. In particular, DNs have been successfully used for the applications of structure learning (Nägele et al., 2007; Tian et al., 2006), recommender systems (de Campos et al., 2007; Hernández del Olmo and Gaudioso, 2008), and computational biology (Carlson et al., 2008). DNs are designed to depict bidirectional probabilistic relations among variables, which makes it easier and more efficient to learn both structures and parameters from real data. Besides, the learned model is quite useful for encoding and displaying the predictive relationships (Heckerman et al., 2001). Up to now, Gibbs sampling is the most popular inference approach for DNs (Heckerman et al., 2001). To the best of our knowledge, sampling method perhaps is the most used inference method for DNs. In this paper, we first propose a new and uniform inference framework for DNs, and then develop several new inference algorithms, including both exact and approximate algorithms.

The main tasks of inference includes computing the marginal distributions and the most probable configuration. Usually, these tasks are conducted by summing up the joint probability that may be given by a model. For example, the joint probability can be factorized into the product of the conditional probabilities making use of the chain rule as in Bayesian networks. However, the known condition on a DN is a set of full conditional distributions (FCDs). Therefore, establishing the relation between the joint probability distributions (JPDs) and FCDs can significantly help studying the inference problems on DNs. Motivated by this idea, in this article we introduce a set of linear equations to describe the relation between JPDs and FCDs. Then the inference task on a DN can be reduced to the problem of solving these linear equations, which, we believe, provides a new perspective to understand the inference problem on DNs.

We introduce two exact algorithms to compute all the posterior probabilities by solving these equations. However, these exact algorithms are computationally expensive. In addition, we do not need to calculate all the posterior probabilities in many applications: we may only need the marginal distributions over a subset. Then several approximate algorithms are proposed. More specifically, we develop a local-mean-field-based algorithm (LMF), which can efficiently compute the marginal distributions over a subset. Compared with the mean field method (MF), which is based on a global mutual independence assumption, our approximation is based on local mutual independence assumption, which is much weaker than that of mean field. Finally, the validity of LMF is illustrated by a series of random experiments.

The remainder of this article is organized as follows. The basics about DNs are introduced in Sec. 2. The relation between JPDs and FCDs is discussed in Sec. 3. The existence and uniqueness of solution is discussed in Sec. 4. Section 5 introduces several new approximate inference algorithms, and Sec. 6 shows the comparison between our framework and other methods.

## 2. Dependency Networks

A dependency network (DN) (Heckerman et al., 2001) is a directed probabilistic graph, consisting of a set of nodes representing random variables and a set of

directed edges linking these nodes. The known probability conditions in a DN are full conditional distributions (FCDs), different from the conditional probabilities on Bayesian networks or the potential functions on Markov random fields (MRFs). In this section, we introduce some detailed definitions concerning with DNs.

Similar to the notations of Heckerman et al. (2001), in this article we denote a variable by a capitalized token (e.g., $X_i$, $V_i$), and the state or value of a corresponding variable by that same token in lower case (e.g., $x_i$, $v_i$). We denote a set of variables by a bold-face capitalized token (e.g., $\mathbf{X}$, $\mathbf{M}(X_i)$) and an assignment of state to each variable in a given set by the corresponding bold-face lower-case token (e.g., $\mathbf{x}$, $\mathbf{m}(x_i)$). The FCD of a variable is defined by Gilks and Spiegelhalter (1996).

**Definition 2.1.** Let $\mathbf{V}$ denote a set of variables $\{V_1, V_2, \ldots, V_N\}$. $P(V_i \mid \mathbf{V}\backslash V_i)$ is the full conditional distribution (FCD) of $V_i$ where $\mathbf{V}\backslash V_i = \{V_1, V_2, \ldots, V_{i-1}, V_{i+1}, \ldots, V_N\}$.

A DN for $\mathbf{V} = (V_1, \ldots, V_N)$ is a (usually cyclic) directed graph, where each node represents a variable in $\mathbf{V}$, together with the FCD of each variable. The parents of a node in a DN are defined by the Markov blanket of the node. A node in a DN is conditionally independent of other nodes given all its parents. Markov blanket conditional distribution (MCD) is often used to describe a compact and complete set of parents for every node.

**Definition 2.2.** If variables $\widehat{\mathbf{M}}(V_i) \subset \mathbf{V}\backslash V_i$ satisfy $P(V_i \mid \mathbf{V}\backslash V_i) = P(V_i \mid \widehat{\mathbf{M}}(V_i))$, then $\widehat{\mathbf{M}}(V_i)$ is a Markov blanket of variable $V_i$, and $P(V_i \mid \widehat{\mathbf{M}}(V_i))$ is a Markov blanket conditional distribution (MCDs) of variable $V_i$.

Markov blanket of a variable may not be unique. The cardinality of a Markov blanket is at least zero (when $\widehat{\mathbf{M}}(V_i)$ is empty) and at most $N-1$ (when $\widehat{\mathbf{M}}(V_i) = \mathbf{V}\backslash V_i$).

**Definition 2.3.** In constructing a DN for $\mathbf{V}$, we set the parents of node $V_i$ to be its Markov boundary $\mathbf{M}(V_i)$, which is the minimum Markov blanket $\widehat{\mathbf{M}}(V_i)$ of $V_i$.

In general, Markov boundary of a variable in a DN is unique and the Markov boundary conditional probabilities $P(V_i \mid \mathbf{M}(V_i))$ for all variables are sufficient for inference tasks. However, there are special cases that the Markov boundary of a variable is not unique. In these cases, the conditional dependence probability between different variables in the Markov blanket may be 1 or 0. These special cases can be easily transformed to general DNs. An example of a Bayesian network and the corresponding DN is shown in Fig. 1. Here MCDs of the DN are computed from the Bayesian network.

## 3. Posterior Distributions and FCDs

The goal of inference is to compute the posterior probabilities, such as the joint probabilities, marginal distributions, and most probable configurations. For Bayesian networks, many conditional probability-based methods, such as belief propagation, bucket elimination, and cluster tree, have been developed to compute

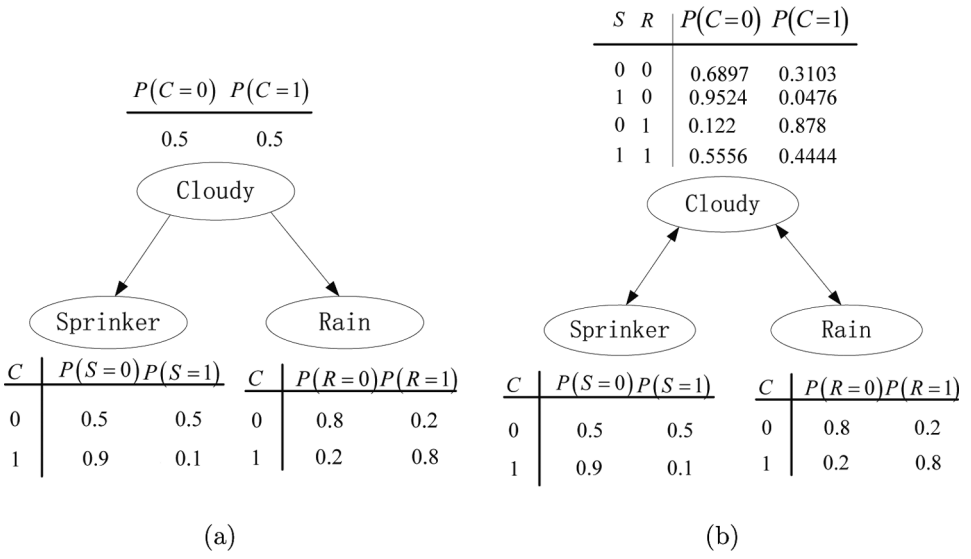


**Figure 1.** (a) A Bayesian network; (b) the dependency network corresponding to (a).

the posterior probabilities. Different from Bayesian networks, the probability component of a DN is a set of full conditional distributions (FCDs) (i.e., Markov blanket conditional distributions). To do inference on a DN, it needs to compute the joint probability distributions (JPDs) based on FCDs. In this section, we explore a set of linear equations to describe the relation between JPDs and FCDs. JPDs can be obtained by solving these equations and marginal distributions can be computed by marginalizing JPDs. This linear equation set serves as a new perspective for inference and can facilitate the inference problems.

In a graphical model, the vertex set (all variables) $\mathbf{V} = \{V_1, V_2, \ldots, V_N\}$ can be divided into two disjoint subsets: the hidden variable set $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ and the evidence $\mathbf{E} = \{E_1, E_2, \ldots, E_{N-n}\}$. $P(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$ is known as posterior joint probability distribution. The joint distribution of all variables, $P(\mathbf{V})$, is a special case of $P(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$ when $\mathbf{E}$ is an empty set. In this article, we assume that each variable $X_i$ has $K$ states labeled as $\{x_i^1, x_i^2, \ldots, x_i^K\}$, and all variables' scales are identical, and $P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, \mathbf{E} = \mathbf{e})$ is full conditional probability of variable $X_i$. From Definition 2.2 and Definition 2.3, $P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, \mathbf{E} = \mathbf{e}) = P(x_i \mid \mathbf{m}(x_i))$, where $\mathbf{m}(x_i)$ is the instantiations of the Markov boundary $\mathbf{M}(X_i)$ and takes corresponding values in $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, \mathbf{E} = \mathbf{e}\}$.

Using the multiplication formula, we have

$$P(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n \mid \mathbf{E} = \mathbf{e}) = P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, \mathbf{E} = \mathbf{e}) \cdot P(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \mid \mathbf{E} = \mathbf{e}). \quad (1)$$

On the right-hand side of the above equation, the first term is full conditional probability of $x_i$ and the second term can be further expanded as

$$P(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \mid \mathbf{E} = \mathbf{e}) = \sum_{k=1}^{K} P(x_1, \ldots, x_{i-1}, X_i = x_i^k, x_{i+1}, \ldots, x_n \mid \mathbf{E} = \mathbf{e}). \quad (2)$$

Then Eq. (1) can be rewritten as

$$P(x_i, \mathbf{x}\backslash x_i \mid \mathbf{E} = \mathbf{e}) = P(x_i \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) \cdot \sum_{k=1}^{K} P(X_i = x_i^k, \mathbf{x}\backslash x_i \mid \mathbf{E} = \mathbf{e}). \tag{3}$$

This formula describes the relations between the posterior probabilities and the full conditional probabilities.

Equation (3) can be regarded as a constraint on a joint probability distribution, where FCDs are the weights. It is a natural idea to use a $K$-dimensional column vector $\mathbf{p}_i(x_i, \mathbf{x}\backslash x_i)$ to represent the joint probabilities associated with the $K$ different states of the variable $X_i$, i.e.,

$$\begin{aligned}\mathbf{p}_i(x_i, \mathbf{x}\backslash x_i) = [&P(x_1, \ldots, x_{i-1}, X_i = x_i^1, x_{i+1}, \ldots, x_n \mid \mathbf{E} = \mathbf{e}),\\ &P(x_1, \ldots, x_{i-1}, X_i = x_i^2, x_{i+1}, \ldots, x_n \mid \mathbf{E} = \mathbf{e}),\\ &\ldots\ldots,\\ &P(x_1, \ldots, x_{i-1}, X_i = x_i^K, x_{i+1}, \ldots, x_n \mid \mathbf{E} = \mathbf{e})]^T.\end{aligned}$$

We can then rewrite Eq. (3) as

$$\mathbf{p}_i^T(x_i, \mathbf{x}\backslash x_i) = \mathbf{p}_i^T(x_i, \mathbf{x}\backslash x_i) \cdot \mathscr{F}_i, \tag{4}$$

where $\mathscr{F}_i$ is a $K \times K$ weight matrix consisting of FCDs:

$$\mathscr{F}_i = \begin{bmatrix} P(X_i = x_i^1 \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) & P(X_i = x_i^2 \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) & \ldots & P(X_i = x_i^K \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) \\ P(X_i = x_i^1 \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) & P(X_i = x_i^2 \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) & \ldots & P(X_i = x_i^K \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) \\ \ldots & \ldots & \ldots & \ldots \\ P(X_i = x_i^1 \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) & P(X_i = x_i^2 \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) & \ldots & P(X_i = x_i^K \mid \mathbf{x}\backslash x_i, \mathbf{E} = \mathbf{e}) \end{bmatrix}.$$

Equation (4) is a linear equation set and Eq. (3) is just one of it.

Since each variable $X_i$ has $K$ states, there are $K^{n-1}$ states of $(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$ and we denote these $K^{n-1}$ states as $(x_i, \mathbf{x}\backslash x_i)_1$, $(x_i, \mathbf{x}\backslash x_i)_2, \ldots, (x_i, \mathbf{x}\backslash x_i)_{K^{n-1}}$. Equation (4) holds for each of these $K^{n-1}$ states. By combining all of them, we have

$$\begin{aligned}&[\mathbf{p}_1((x_i, \mathbf{x}\backslash x_i)_1), \mathbf{p}_2((x_i, \mathbf{x}\backslash x_i)_2), \ldots, \mathbf{p}_{K^{n-1}}((x_i, \mathbf{x}\backslash x_i)_{K^{n-1}})]^T\\ &\quad = [\mathbf{p}_1((x_i, \mathbf{x}\backslash x_i)_1), \mathbf{p}_2((x_i, \mathbf{x}\backslash x_i)_2), \ldots, \mathbf{p}_{K^{n-1}}((x_i, \mathbf{x}\backslash x_i)_{K^{n-1}})]^T\\ &\qquad \cdot diag(\mathscr{F}_1, \mathscr{F}_2, \ldots, \mathscr{F}_{K^{n-1}}),\end{aligned} \tag{5}$$

where $diag(\mathscr{F}_1, \mathscr{F}_2, \ldots, \mathscr{F}_{K^{n-1}})$ is a $K^n \times K^n$ matrix constructed by concatenating $K^{n-1}$ different $\mathscr{F}_i$ where $i = 1, 2, \ldots, K^{n-1}$ along the matrix diagonal and filling all the other elements with zero.

There are $K^n$ equations in total in Eq. (5), each having the form similar to Eq. (3). The elements of the $K^n$ dimensional vector on the left-hand side of Eq. (5) are the joint probabilities of $\mathbf{X}$, i.e., $(\mathbf{x})_1, (\mathbf{x})_2, \ldots, (\mathbf{x})_{K^n}$. To compact the expression, all these joint probabilities can be reordered into a vector:

$$\mathbf{p} = [P(X = (\mathbf{x})_1 \mid \mathbf{E} = \mathbf{e}), P(X = (\mathbf{x})_2 \mid \mathbf{E} = \mathbf{e}), \ldots, P(X = (\mathbf{x})_{K^n} \mid \mathbf{E} = \mathbf{e})]^T.$$

Then Eq. (5) is can be written as

$$\mathbf{p}^T = \mathbf{p}^T \cdot \mathscr{A}_i. \tag{6}$$

$\mathscr{A}_i$ is a $K^n \times K^n$ matrix consisting of FCDs and there are in total $n$ such matrices, with the above $diag(\mathscr{F}_1, \mathscr{F}_2, \ldots, \mathscr{F}_{K^{n-1}})$ being one of them. Each row of $\mathscr{A}_i$ has $K$ non zero elements corresponding to $K$ instantiations of $X_i$ in $\mathbf{p}(X_i, \mathbf{x}\backslash x_i)$. So the summation of the elements along any row of $\mathscr{A}_i$ is one. Therefore, the following proposition can be easily obtained

**Proposition 3.1.** *$\mathscr{A}_i$ is a random matrix.*

Note that $\mathscr{A}_i$ are known FCDs and our goal is to find $\mathbf{p}$, which represents the joint posterior probabilities. Therefore, we only need to solve Eq. (6) for $\mathbf{p}$. However, $\mathscr{A}_i$ is not a full rank matrix: its rank is at most $K^{n-1}$. Therefore, only using the FCDs for one variable $X_i$ is not sufficient for solving for the desired JPDs. We can consider FCDs for all the variables and get the following theorem:

**Theorem 3.1.** *The joint posterior probabilities are the elements of* $\mathbf{p}$*, which are the solutions of the following equations*:

$$\begin{aligned} \mathbf{p}^T &= \mathbf{p}^T \cdot \mathscr{A}_i, \quad i = 1, 2, \ldots, n \\ \sum_{j=1}^{K^n} p_j &= 1, \end{aligned} \tag{7}$$

*where* $p_j = P(X = (\mathbf{x})_j \mid \mathbf{E} = \mathbf{e})$ *is the probability of the jth state of variables* $\mathbf{X}$*, which also is an element of* $\mathbf{p}$*. The second equation in* (7) *enforces the probability of all states is equal to* 1.

## 4. Existence and Uniqueness of the Solution

For Eq. (7), the existence of a solution is obvious: the joint probability distribution of all the hidden variables is one of its solution. To analyze the uniqueness of solution, we first define a combined matrix $\mathscr{A} = \prod_{i=1}^{n} \mathscr{A}_i$ and then derive the following equations from Eq. (7):

$$\begin{aligned} \mathbf{p}^T &= \mathbf{p}^T \cdot \mathscr{A} \\ \sum_{j=1}^{K^n} p_j &= 1, \end{aligned} \tag{8}$$

where $\mathscr{A}$ is still a random matrix. If $\mathbf{p}$ is the solution of Eq. (7), then it is the solution of Eq. (8). Therefore, if the solution of Eq. (8) is unique, then the solution of Eq. (7) is also unique. We will prove the uniqueness of the solution following this way.

### 4.1. *Uniqueness of the Solution*

The random matrix $\mathscr{A} = \{a_{lj}\}_{K^n \times K^n}$ can be viewed as a probability transition matrix of a Markov chain with $K^n$ states. The element $a_{lj}$ at the $l$th row and the $j$th column is the transition probability from the $l$th state to the $j$th state. Accordingly, the problem of solving these linear equations can be transformed to computing the stationary distribution of this Markov chain. The stationary distribution of a finite-state Markov chain always exists, which ensures the existence of a solution of Eq. (8). Recalling the theory of Markov processes on the existence of a unique final state, we have the following theorem:

**Theorem 4.1.** *A necessary and sufficient condition for the existence and uniqueness of a solution of Eq.* (8) *is that the Markov chain with the transition matrix* $\mathscr{A}$ *has one and only one recurrent closed set. A sufficient condition for the existence and uniqueness of a solution of Eq.* (8) *is that the Markov chain with the transition matrix* $\mathscr{A}$ *is irreducible, i.e., all states can communicate mutually.*

Here, we roughly prove that matrix $A$ is irreducible. Specifically, with $\mathscr{A} = \prod_{i=1}^{n} \mathscr{A}_i$, then $\mathbf{p}^T = \mathbf{p}^T\mathscr{A} = \mathbf{p}^T\mathscr{A}_1\mathscr{A}_2 \ldots \mathscr{A}_n$. From the sense and structure of $\mathscr{A}_i$, it is known that one state can communicate with all the others and the transition probabilities are defined in matrix $\mathscr{A}$. This indicates that all the states defined by $\mathscr{A}$ can communicate mutually. We can then obtain the following corollaries.

**Corollary 4.1.** *If all MCDs in a DN of a stochastic system show nonzero probability for every state and* $\mathscr{A} = \prod_{i=1}^{n} \mathscr{A}_i$, *then Eq.* (8) *has a unique solution which determines the JPD of all the hidden variables.*

**Corollary 4.2.** *If all the MCDs in a DN of a stochastic system show nonzero probability for every state and* $\mathscr{A} = \sum_{i=1}^{n} \alpha_i \mathscr{A}_i$ ($\alpha_i > 0$ *and* $\sum_{i=1}^{n} \alpha_i = 1$), *then Eq.* (8) *has a unique solution which determines the JPD of all the hidden variables.*

If any Markov boundary of the DN is not unique, the solution $\mathbf{p}$ is not unique. In that case, each zero-probability (or one-probability) variable can be combined with its neighbor as a single variable, and we can then abtain a unique solution based on the new matrix $\mathscr{A}$.

### 4.2. *Exact Solutions*

There are many ways to solve Eq. (8) for an exact solution. In this section, we introduce two popular methods.

First, the solution of Eq. (8) is the unit eigenvector of $\mathscr{A}^T$ that corresponds to the eigenvalue 1. Therefore, we can solve Eq. (8) directly by computing this unit eigenvector.

The second method is summarized in Algorithm 1.

**Algorithm 1** Exact Algorithm

1: Let $\mathcal{H}_1 = \mathcal{A}^T - \mathcal{I}$, where $\mathcal{I}$ is an identity matrix;

2: Replace the first row of $\mathcal{H}_1$ by $[1, 1, ..., 1]$, obtaining matrix $\mathcal{H}_2$;

3: Compute $\mathcal{H}_2^{-1}$, the inverse matrix of $\mathcal{H}_2$;

4: Return the first column of matrix $\mathcal{H}_2^{-1}$ as the desired solution $\mathbf{p}$.

Many other exact linear-equation solvers can also used to solve Eq. (8). However, with high computational complexity, it is difficult to apply such exact methods when the dimension of $\mathscr{A}$ is high. Then, it is necessary to explore efficient approximate inference methods.

## 5. Approximate Inference Methods

In this section we introduce two approximate methods to solve Eq. (8). The first algorithm uses an iterative approach, and the second algorithm explores the relation between the marginal probabilities and FCDs, which can compute the marginal probabilities without marginalizing JPDs.

### 5.1. *General and Square Iteration Algorithms*

From Eq. (8), we have $\mathbf{p}^T = \mathbf{p}^T\mathscr{A}^2$, $\mathbf{p}^T = \mathbf{p}^T\mathscr{A}^3$, ..., $\mathbf{p}^T = \mathbf{p}^T\mathscr{A}^m$. Hence, the following *general iteration algorithm* (Algorithm 2) can be used to calculate the desired $\mathbf{p}$.

**Algorithm 2** General Iteration Algorithm

1: Initialize $\mathbf{p}_{old}$ and $\mathbf{p}_{new}^T = \mathbf{p}_{old}^T \cdot \mathcal{A}$

2: **while** $\|\mathbf{p}_{new} - \mathbf{p}_{old}\| > \varepsilon$ **do**

3: $\quad \mathbf{p}_{old} = \mathbf{p}_{new}$

4: $\quad \mathbf{p}_{new}^T = \mathbf{p}_{old}^T \cdot \mathcal{A}$

5: **end while**

6: Return $\mathbf{p}_{new}$

When the solution of Eq. (8) is unique, the sequence $\mathscr{A}^m$, $m = 1, 2, \ldots$ will converge to $\mathscr{A}^\infty$ (Todorovic, 1992). It is easy to prove that all the elements in each column of matrix $\mathscr{A}^\infty$ are the same, and the element in the $j$th column is the posterior probability of the $j$th state. Therefore, $\mathbf{p}$ is the same as a row in $\mathscr{A}^\infty$. From this point of view, $mean(\mathscr{A}^m, col)$, or the column average of $\mathscr{A}^m$, $m > 0$, can be used to estimate $\mathbf{p}$. The iteration algorithm can be summarized in Algorithm 3.

**Algorithm 3** Square Iteration Algorithm

1: Initialize $\mathcal{A}_s = \mathcal{A}$ and $\mathbf{p}_{old}^T = mean(\mathcal{A}_s, col)$
2: $\mathcal{A}_s = \mathcal{A}_s^2$
3: $\mathbf{p}_{new}^T = mean(\mathcal{A}_s, col)$
4: **while** $\|\mathbf{p}_{new} - \mathbf{p}_{old}\| > \varepsilon$ **do**
5: $\quad \mathbf{p}_{old} = \mathbf{p}_{new}$
6: $\quad \mathcal{A}_s = \mathcal{A}_s^2$
7: $\quad \mathbf{p}_{new}^T = mean(\mathcal{A}_s, col)$
8: **end while**
9: Return $\mathbf{p}_{new}$

By multiplying matrix $\mathcal{A}$ by itself $2^m$ times in $m$ iterations, the *square iteration algorithm* usually takes fewer iterations than the *general iteration algorithm* for convergence.

### 5.2. *Local-Mean-Field-Based Algorithm*

The above methods compute the joint distributions of all the hidden variables. When computing marginal probabilities, it needs to sum up these joint distributions. In this section, we derive the relation between the marginal distributions and FCDs, with which we can compute the marginal distributions directly.

Specifically, we have

$$P\left(x_i, \mathbf{m}\left(x_i\right) \mid \mathbf{E} = \mathbf{e}\right) = P\left(\mathbf{m}(x_i) \mid \mathbf{E} = \mathbf{e}\right) P\left(x_i \mid \mathbf{m}\left(x_i\right)\right). \tag{9}$$

Denote $\mathbf{S}_i$ to be the set of all possible combined states of $\mathbf{M}(X_i)$. The cardinality of $\mathbf{S}_i$ is then $K^{|\mathbf{M}(X_i)|}$, where $|\mathbf{M}(X_i)|$ is the number of hidden variables in $\mathbf{M}(X_i)$. On both sides of Eq. (9), taking a summation over $\mathbf{S}_i$, we have

$$P\left(x_i \mid \mathbf{E} = \mathbf{e}\right) = \sum_{\mathbf{m}(x_i) \in \mathbf{S}_i} P\left(\mathbf{m}\left(x_i\right) \mid \mathbf{E} = \mathbf{e}\right) P\left(x_i \mid \mathbf{m}\left(x_i\right)\right). \tag{10}$$

If all the variables in the Markov blanket are mutually independent given the evidence, then

$$P\left(\mathbf{m}\left(x_i\right) \mid \mathbf{E} = \mathbf{e}\right) = \prod_{x_j \in \mathbf{m}(m_i)} P\left(x_j \mid \mathbf{E} = \mathbf{e}\right). \tag{11}$$

In general, Markov blanket variables are not absolutely independent with each other and we only obtain an approximate equation by combining Eqs. (10) and (11):

$$P\left(x_i \mid \mathbf{E} = \mathbf{e}\right) \approx \sum_{\mathbf{m}(x_i) \in \mathbf{S}_i} \prod_{x_j \in \mathbf{m}(x_i)} P\left(x_j \mid \mathbf{E} = \mathbf{e}\right) P\left(x_i \mid \mathbf{m}\left(x_i\right)\right). \tag{12}$$

We introduce the following iteration algorithm (Algorithm 4) to solve the marginal probability $P(x_i \mid \mathbf{E} = \mathbf{e})$, $i = 1, 2, \ldots, n$ based on Eq. (12).

**Algorithm 4** Local-mean-field-based Algorithm (LMF)

1: Initialize $P_{old}(x_i|\mathbf{E}=\mathbf{e})$, $x_i \in \{x_i^1, x_i^2, \ldots, x_i^K\}$, $i = 1, 2, \ldots, n$

2: **for** $i = 1, 2, ..., n$ **do**

3: $P_{new}(x_i|\mathbf{E}=\mathbf{e}) = \sum_{\mathbf{m}(x_i)\in\mathbf{S}_i} \prod_{x_j\in\mathbf{m}(x_i)} P_{old}(x_j|\mathbf{E}=\mathbf{e})P(x_i|\mathbf{m}(x_i))$, $x_i \in \{x_i^1, x_i^2, \ldots, x_i^K\}$

4: **end for**

5: **while** $\sum_{k=1}^{K}\sum_{i=1}^{n} \left|P_{new}\left(x_i^k|\mathbf{E}=\mathbf{e}\right) - P_{old}\left(x_i^k|\mathbf{E}=\mathbf{e}\right)\right| > \varepsilon$ **do**

6: $P_{old}(x_i|\mathbf{E}=\mathbf{e}) = P_{new}(x_i|\mathbf{E}=\mathbf{e}), i = 1, 2, \ldots, n$

7: **for** $i = 1, 2, ..., n$ **do**

8: $P_{new}(x_i|\mathbf{E}=\mathbf{e}) = \sum_{\mathbf{m}(x_i)\in\mathbf{S}_i} \prod_{x_j\in\mathbf{m}(x_i)} P_{old}(x_j|\mathbf{E}=\mathbf{e})P(x_i|\mathbf{m}(x_i)), x_i \in \{x_i^1, x_i^2, \ldots, x_i^K\},$

9: **end for**

10: **end while**

11: Return $P_{new}(x_i|\mathbf{E}=\mathbf{e}), x_i \in \{x_i^1, x_i^2, \ldots, x_i^K\}, i = 1, 2, ..., n$

The algorithm is exact when Markov blanket variables are mutually independent given the evidence. In general, there is correlation among the variables of Markov blanket and this algorithm only gives an approximate solution. However, experiments show that this algorithm can obtain good approximations even if there are strong correlations among the variables.

### 5.3. *The Validity of LMF*

Here, we will show the reasonableness and the validity of LMF by analyzing the relationship between LMF and other approximate inference methods. To the best of our knowledge, Gibbs sampling method is the only inference algorithm currently used for DNs. However, the nature of Gibbs sampling method is completely different from that of LMF, and it is difficult to analyze their relationship theoretically. Specifically, LMF is a kind of iteration algorithm. In this section, we focus on exploring its close relationship with other iteration inference algorithms, such as loopy belief propagation algorithm (LBP) and mean field algorithm (MF). However, LBP is designed for Bayesian networks or MRFs, not for DNs. To address this problem, we convert Bayesian networks to DNs and show that the formulations of LMF on DNs are similar to that of LBP on the corresponding Bayesian networks. This similarity can be regarded as indirect evidence of the validity of LMF.

Consider a typical fragment of a Bayesian network, as shown in Fig. 2(a), which consists of a node $X$, its parents $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$, and its children $\mathbf{Z} = \{Z_1, Z_2, \ldots, Z_m\}$. The fragment of the DN corresponding to this network is shown in Fig. 2(b) where $\mathbf{M}(X) = \{Y_1, Y_2, \ldots, Y_n, Z_1, Z_2, \ldots, Z_m, \mathbf{Y}_p\}$. The set $\mathbf{Y}_p$ represents the parents (excluding $X$) of all the elements in $\mathbf{Y}$. For this node $X$, Eq. (12) used in LMF can be written as

$$P(x \mid \mathbf{E}=\mathbf{e}) \approx \sum_{\mathbf{m}(x)\in\mathbf{S}_x} \prod_{u_j\in\mathbf{m}(x)} P\left(u_j \mid \mathbf{E}=\mathbf{e}\right) P(x \mid \mathbf{m}(x))$$

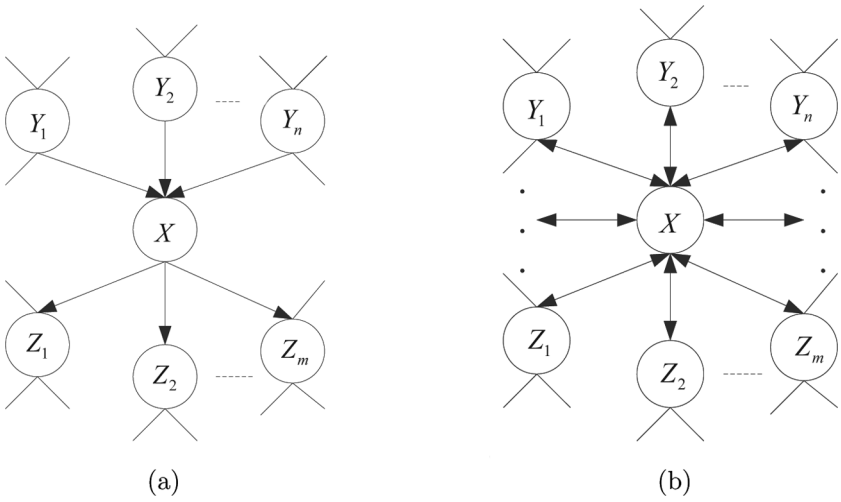


**Figure 2.** (a) A node with its neighbors in a BN; (b) a node with its neighbors in a DN.

$$= \sum_{\mathbf{m}(x)\in\mathbf{S}_x} P(x \mid \mathbf{m}(x)) \prod_{u_j\in\mathbf{m}(x)} P(u_j \mid \mathbf{E} = \mathbf{e}), \tag{13}$$

where $\mathbf{S}_x$ are all the combined states of $\mathbf{M}(X)$. The belief propagation (BP) algorithm of a singly connected Bayesian network is (Pearl, 1988)

$$\begin{aligned} P(x \mid \mathbf{E} = \mathbf{e}) &\propto \pi(x)\,\lambda(x) \\ &\propto \left[\sum_{\mathbf{y}\in\mathbf{S}_y} P(x \mid \mathbf{y}) \prod_{y_i\in\mathbf{y}} \pi_x(y_i)\right]\left[\prod_{z_j\in\mathbf{z}} \lambda_{z_j}(x)\right], \end{aligned} \tag{14}$$

where $\mathbf{S}_y$ are all the combined states of $\mathbf{Y}$.

As the edges in DNs are bidirectional, the two nodes combined by one edge can be regarded as the parent of one another. By comparison, the parents of one node are the Markov boundary of this node, and this iteration algorithm can be regarded as a belief propagation algorithm carried out on DNs, and

$$\begin{aligned} P(x \mid \mathbf{E} = \mathbf{e}) &\propto \pi(x) \\ &\propto \sum_{\mathbf{m}(x)\in\mathbf{S}_{\mathbf{m}(x)}} P(x \mid \mathbf{m}(x)) \prod_{u_j\in\mathbf{m}(x)} \pi_x(u_i). \end{aligned} \tag{15}$$

This comparison reveals the similarity between LMF and LBP, which implies that the results of LMF should be similar to those of LBP. In Sec. 6.1, we will show experiments to further justify the similarity between LMF and LBP.

### 5.4. *Markov Chain Monte Carlo (MCMC) Method*

Equation (8) can also be solved by the MCMC method. If $\mathscr{A} = \prod_{i=1}^{n} \mathscr{A}_i$, it is equivalent to Gibbs sampling algorithm (Bidyuk and Dechter, 2007; Gilks and Spiegelhalter, 1996; Heckerman et al., 2001). The stationary distribution of a

Markov chain defined by matrix $\mathcal{A}$ is $P(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$. Letting $\mathbf{X}^s = \{X_1^s, \ldots, X_n^s\}$ be the $s$th sample of $\mathbf{X}$ and $X_i^s$ be the $s$th sample of $X_i$, where $s \in \{1, \ldots, S\}$ and $S$ is the number of samples, the stationary distribution of $X_i^s$ is $P(X_i \mid \mathbf{E} = \mathbf{e})$. According to the law of large number, if the sample size $S$ is infinite, the JPD of all the hidden variables can be calculated by

$$P(\mathbf{X} = \mathbf{x} \mid \mathbf{E} = \mathbf{e}) = \frac{1}{S} \sum_{s=1}^{S} \delta(\mathbf{X}^s, \mathbf{x}), \tag{16}$$

where $\delta(\mathbf{X}^s, \mathbf{x}) = 1$ when $\mathbf{X}^s = \mathbf{x}$, and $\delta(\mathbf{X}^s, \mathbf{x}) = 0$ otherwise. The JPD of a single variable $X_i$ can be calculated by

$$P(X_i = x_i \mid \mathbf{E} = \mathbf{e}) = \frac{1}{S} \sum_{s=1}^{S} \delta(X_i^s, x_i), \tag{17}$$

where $\delta(X_i^s, x_i) = 1$ when $X_i^s = x_i$, and $\delta(X_i^s, x_i) = 0$ otherwise.

The MCMC method has low efficiency and may take long time to converge to a stationary distribution (Gilks and Spiegelhalter, 1996; Sahu and Roberts, 1998).

## 6. Experiments

In this section, we conduct experiments to test the algorithms mentioned above and show the efficiency of the proposed algorithms. The test algorithms include the exact algorithm, the *general iteration algorithm*, *square iteration algorithm,* and *local-mean-field-based algorithm*. As a comparison, we also include the MCMC sampling method in the testing.

### 6.1. *Exact and Iteration Algorithms*

DNs can be learned from Bayesian networks or from data directly. The graphical model shown in Fig. 3 is a DN corresponding to the popular waste

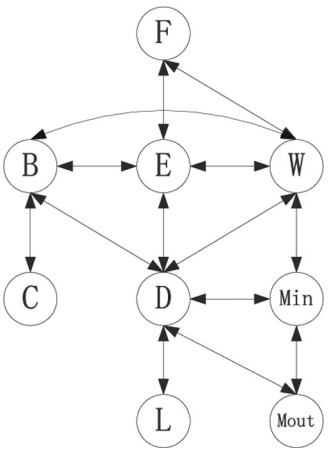


**Figure 3.** The dependency network of the waste incinerator example.

incinerator example (Lauritzen, 1992), and the full condition distributions of the DN are obtained from the conditional distributions. In our experiment, variables $C, E, M_{in}$ are set to be the evidence. The remaining six binary variables are hidden and follow the order of $F, W, B, D, M_{out}, L$ in the joint posterior probability $P(F, W, B, D, M_{out}, L \mid E = 0, C = 1, M_{in} = 0)$ throughout the experiment. We construct $\mathscr{A}_i$, $i = 1, 2, \ldots, 6$, for these six hidden variables, respectively. The joint posterior probability for all states is denoted by

$$\mathbf{p} = [P(000000), P(100000), \ldots, P(011111), P(111111)]^T,$$

which is initialized as

$$\left[\frac{1}{64}, \frac{1}{64}, \ldots, \frac{1}{64}\right]^T.$$

We test the *general iteration algorithm* and the *square iteration algorithm* to compute the joint probability distribution $P(F, W, B, D, M_{out}, L \mid E = 0, C = 1, M_{in} = 0)$. MCMC sampling can be used to compute both the joint probability distributions and the marginal probability distributions. Since it is much easier and faster to compute the marginal probability distributions, MCMC method is used to compute the marginal probability in this experiment.

The error for the *general iteration algorithm* and the *square iteration algorithm* is defined by the mean square error as follows:

$$Error = \frac{1}{N} \sum_{\mathbf{x} \in S_{\mathbf{x}}} \left(P(\mathbf{x}) - \widehat{P}(\mathbf{x})\right)^2, \tag{18}$$

where $S_{\mathbf{x}}$ is the set of all possible combined states of $\mathbf{X}$, with $N$ states, and $P(\mathbf{x})$ is the true probability. The error for MCMC sampling is defined by mean square error of marginal distribution similar to Eq. (18).

The complexity of the exact algorithms, *general iteration algorithm,* and *square iteration algorithm* are $O(d^3)$, $O(d^2)$, $O(d^3)$, where $d = K^n$ is the dimension of the matrix $\mathscr{A}$. The CPU time taken by different algorithms on this experiment is shown in Table 1. We implement all the algorithms in Matlab 7.8.0 and the reported CPU time is based on a computer with Dual Intel-Pentium CPUs (E2180@2.00GHz) and 2.00GB RAM.

The error curves of the *general iteration algorithm* and the *square iteration algorithm* are shown in Fig. 4, and the error curve of MCMC sampling is shown in Fig. 5, respectively, where $\mathscr{A} = \mathscr{A}_{prod}$ indicates the case that we use the product of all $\mathscr{A}_i$'s as $\mathscr{A}$ and $\mathscr{A} = \mathscr{A}_{sum}$ indicates the case that we use the average of all $\mathscr{A}_i$'s as $\mathscr{A}$. From the experimental results, we can see that the *general iteration algorithm* and the *square iteration algorithm* can produce approximate solutions very close to

**Table 1**
CPU time of different algorithms

| Algorithm | Exact | General iteration | Square iteration | MCMC |
|---|---|---|---|---|
| CPU Time (ms) | 0.73 | 0.31 | 5.70 | 936.84 |

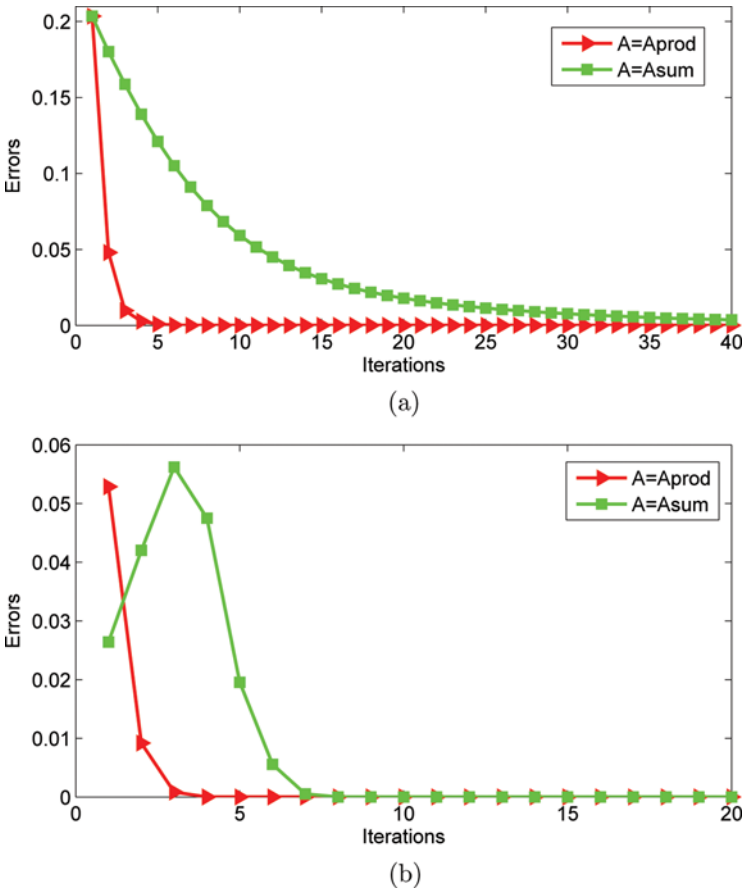


**Figure 4.** (a) Errors of the *general iteration algorithm*; (b) errors of the *square iteration algorithm*. (color figure available online.)

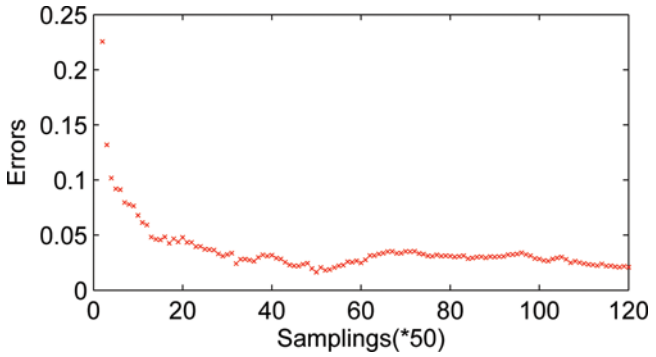


**Figure 5.** Errors of the MCMC method. (color figure available online.)

the exact solution, while the MCMC sampling method may produce a small error even though the sampling number is large enough. The approximate solution from the *general iteration algorithm* is almost identical to the exact solution after five iterations when using $\mathcal{A}_{prod}$, and after about 40 iterations when using $\mathcal{A}_{sum}$. The *square iteration algorithm* takes fewer iterations to converge than the *general iteration algorithm*. However, in total, the *square iteration algorithm* takes more CPU time than the *general iteration algorithm*. The MCMC algorithm takes more than $2,000$ samples in order to obtain good approximations. From Table 1, we can see that the proposed exact algorithm, *general iteration algorithm* and the *square iteration algorithm* are $10^4$ times faster than the MCMC algorithm in this experiment.

### 6.2. *The Validity of LMF*

Here, we will conduct experiments to validate LMF by comparing the results of LMF to those of two other efficient approximate inference algorithms: LBP and MF methods. Specifically, we compare the average errors of these three methods. However, since LBP and MF are not suitable for DNs, we cannot run them directly on DNs. As discussed in Sec. 5.3, we start from Bayesian networks and MRFs and convert them into DNs, keeping the joint distribution to be the same. We then run LBP and MF on the MRFs and LMF on the corresponding DNs, and finally compare their inference errors. This way, LMF is valid and efficient if its results are similar to those of LBP and MF. Figure 6 illustrates a $10 \times 10$ lattice DN, which is converted from a lattice pair-wise MRF. The distribution of this MRF has the form $p(\mathbf{x}) \propto \exp\left(\sum_{i\in\mathbf{V}} \theta_i x_i + \sum_{(i,j)\in\mathbf{E}} \theta_{ij} x_i x_j\right)$, where $\theta_i$, $\theta_{ij}$ are parameters, $x_i \in \{\pm 1\}$, $\mathbf{V}$ is the set of nodes and $\mathbf{E}$ is the set of edges. The parameters $\theta_i$ are drawn uniformly from $\mathcal{U}\left[-d_f, d_f\right]$ where $d_f \in \{0.05, 1\}$. The parameters $\theta_{ij}$ are drawn uniformly from $\mathcal{U}[0, d_o]$, and $d_o \in \{0.2, 0.4, \ldots, 4\}$ are the interaction strength shown in Fig. 7. The mean $\ell_1-$error $\frac{1}{|\mathbf{V}|}\sum_{i\in\mathbf{V}} |p_i^{app}(x_i = 1) - p_i^{true}(x_i = 1)|$ is used to evaluate the results, which are averaged over 30 trials for each setting of the edge strength.

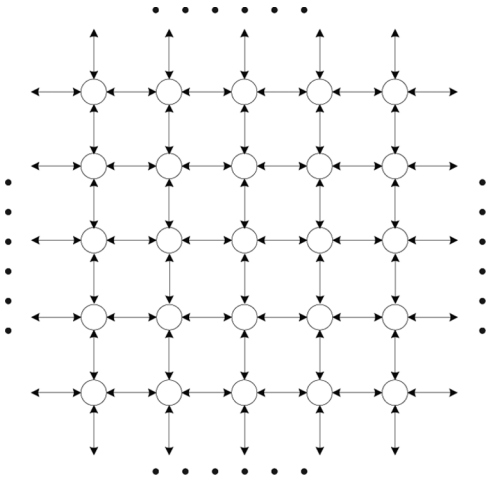

**Figure 6.** A $10 \times 10$ dependence networks.

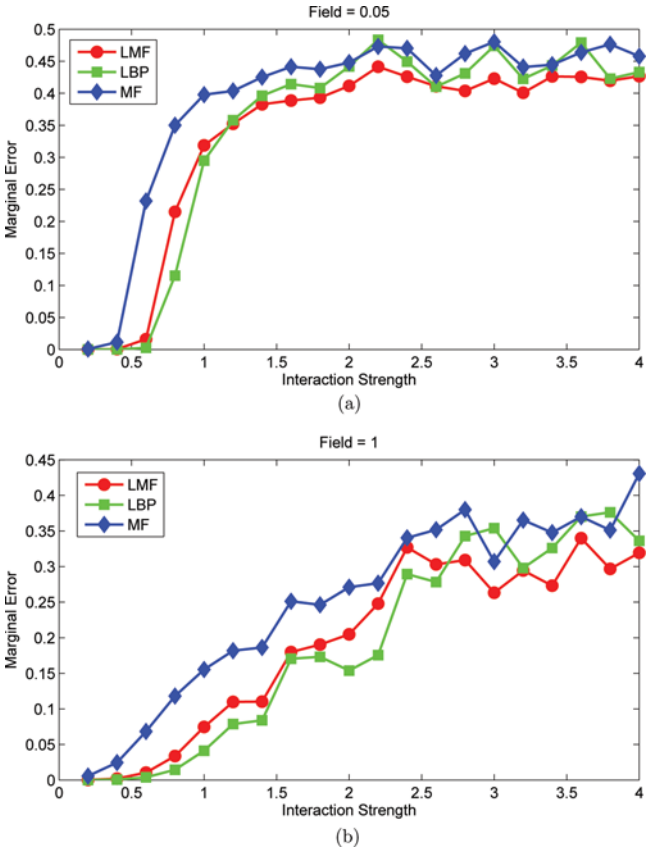


**Figure 7.** Illustration of the performance of LMF on different DNs. (color figure available online.)

Figure 7 shows the validity of LMF. More particularly, the results show that: LMF method yields better marginals than MF. The errors of LMF and LBP are similar, which is consistent with the discussion in Sec. 5.3.

We further test the convergence of LMF. The convergence of LMF is almost the same as LBP. However, in the nonconvergent cases, the results of LBP always fall into oscillation, while LMF may not. Some of the nonconvergent cases are shown in Fig. 8. This experiment shows good convergence property of LMF.

### 6.3. *Different Graph Structures and Parameters*

Here, we apply LMF to DNs with different graph structures and parameters to explore their influence to the performance of LMF. We first test LMF to DNs with different graph structures, as shown on the top of Fig. 9. Different graph structures reveal different dependencies relations among variables in the graphical model. Then, this experiment can illustrate the relation between the performance of LMF and the dependencies among variables. The testing results are shown at the bottom of Figure 9, where from the left to the right, the four curves show the errors after each iteration on the DNs with weaker and weaker dependencies among

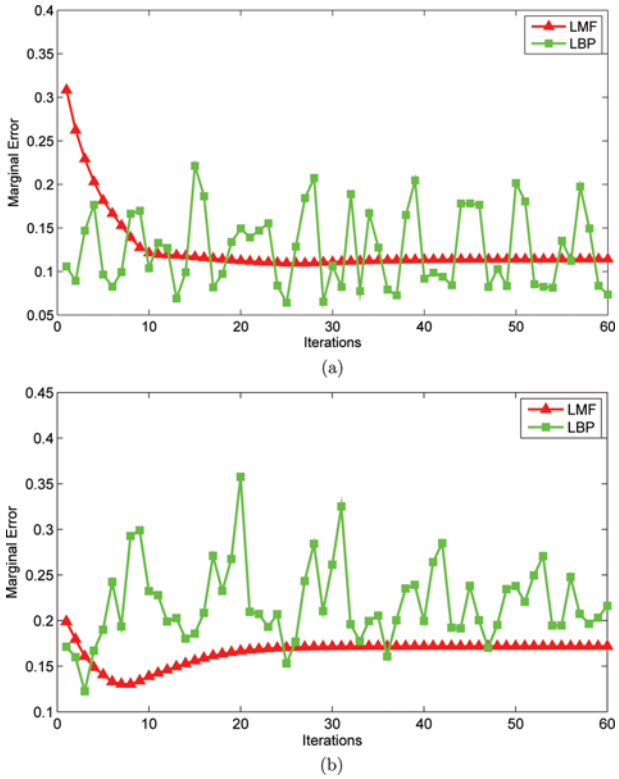


**Figure 8.** Illustration of the convergence of LMF on tested DNs. (color figure available online.)

the variables (sparser graph structures). From the rightmost curve shown in Fig. 9, we can see that LMF can produce a good approximate solution when the graph is sufficiently sparse, i.e., there are only weak dependencies among the variables.

We also conduct similar experiments on a more complex graphical model, as shown in Fig. 6. We use different parameters (such as the interaction strength) to

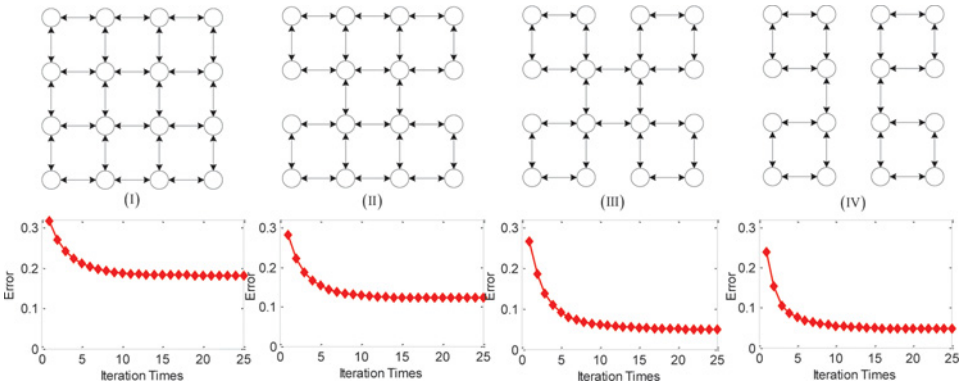


**Figure 9.** Errors of LMF on DNs with different structures. (color figure available online.)

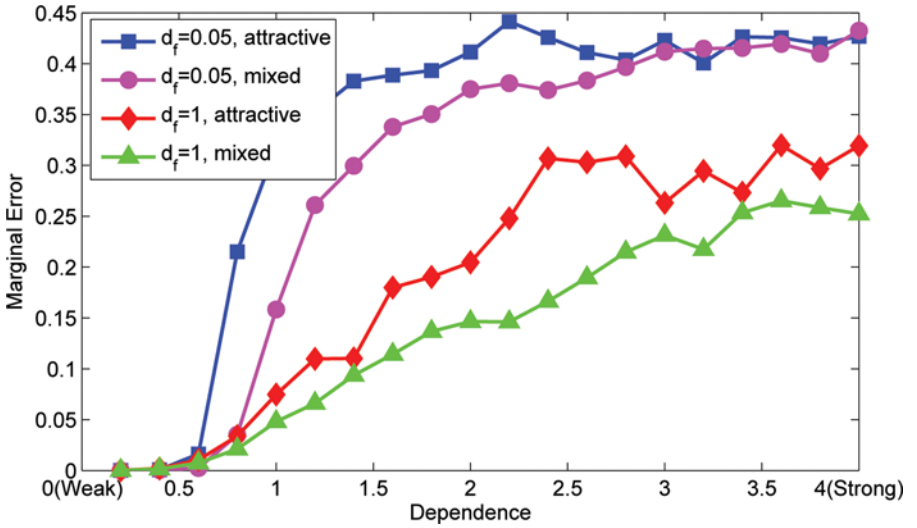


**Figure 10.** Error of LMF on DNs with different parameters. (color figure available online.)

represent different dependencies among variables. The values of $x-$axis represent the dependencies among variables, where large values denote strong dependencies. $d_f$ is the same as in the previous experiment, and "attractive" and "mixed" represent that the parameters $\theta_{ij}$ are drawn uniformly from $\mathcal{U}[0, d_o]$ and $\mathcal{U}[-d_o, d_o]$, respectively. We perform 30 trials for each parameter as before. Figure 10 shows the final errors corresponding to each dependency setting. We can see that LMF will produce better approximation when there are weak dependencies among variables.

## 7. Conclusion and Future Work

In this article, we developed a new inference framework for dependency networks, including deterministic algorithms, approximate algorithms, and sampling methods. A set of linear equations is introduced to describe the relation between FCDs and JPDs, which provides a novel interpretation for the inference on DNs. The proposed *local-mean-field-based algorithm* achieves high efficiency by assuming local independency among the variables in the Markov blanket and this algorithm can obtain good approximations even if there is a strong dependency among variables.

In the future, we plan to develop more efficient inference algorithm under this framework. The transition matrix used by the MCMC sampling algorithm is only a special sampling matrix in our framework. Actually, matrix $\mathcal{A}_i$ is much simpler and sparser than matrix $\mathcal{A}$. In the future, we plan to explore more efficient algorithms based directly on Eq. (7), or by making use of the sparseness of matrix $\mathcal{A}_i$. In addition, *local-mean-field-based algorithm* uses the simplest structure and naive mean field assumption. Finally, we plan to extend the proposed algorithms by introducing other assumptions, such as the region-based mean field assumption.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61071131), National Basic Research and Development Program of China (2009CB320602), Beijing Natural Science Foundation (4122040), ARL Under Cooperative Agreement Number W911NF-10-2-0060 and United Technologies Research Center (UTRC).

## References

Bidyuk, B., Dechter, R. (2007). Cutset sampling for Bayesian networks. *J. Artif. Intell. Res.* 28(1):1–48.

Carlson, J., Brumme, Z., Rousseau, C., Brumme, C., Matthews, P., Kadie, C., Mullins, J., Walker, B., Harrigan, P., Goulder, P. (2008). Phylogenetic dependency networks: inferring patterns of CTL escape and codon covariation in HIV-1 Gag. *PLoS Computat. Biol.* 4(11).

de Campos, L. M., Fernandez-Luna, J. M., Huete, J. F., Rueda-Morales, M. A. (2007). Group recommending: a methodological approach based on bayesian networks In: *ICDEW '07: Proc. 2007 IEEE 23rd Int. Conf. Data Eng. Workshop*. Washington, D.C.: IEEE Computer Society, pp. 835–844.

Gilks, W. (1996). Full conditional distributions. *Markov Chain Monte Carlo in Practice*. pp. 75–88.

Gilks, W., Spiegelhalter, D. (1996). *Markov Chain Monte Carlo in Practice*. London: Chapman & Hall/CRC.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., Kadie, C. (2001). Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.* 1:49–75.

Hernández del Olmo, F., Gaudioso, E. (2008). Evaluation of recommender systems: A new approach. *Expert Syst. Applic.* 35(3):790–804.

Heskes, T., Opper, M., Wiegerinck, W., Winther, O., Zoeter, O. (2005). Approximate inference techniques with expectation constraints. *J. Statist. Mech. Theor. Experi.* 2005:P11015.

Kschischang, F., Frey, B., Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theor.* 47(2):498–519.

Lauritzen, S. (1992). Propagation of probabilities, means, and variances in mixed graphical association models. *J. Amer. Statist. Assoc.* 87(420):1098–1108.

Lepar, V. (1998). A comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer architectures for computing marginals of probability distributions. *Proc. 14th Conf. Uncertain. Artif. Int. (UAI-98)*, Madison, WI, pp. 328–337.

Minka, T. (2001). *A Family of Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Mooij, J., Kappen, H. (2007). Loop corrections for approximate inference on factor graphs. *J. Mach. Learn. Res.* 8:1143.

Nägele, A., Dejori, M., Stetter, M. (2007). Bayesian substructure learning – approximate learning of very large network structures. *ECML '07: Proc. 18th Eur. Conf. Mach. Learn.* Berlin, Heidelberg: Springer-Verlag, pp. 238–249.

Neal, R. (1993). Probabilistic Inference Using Markov Chain Monte Carlo Methods. *Technical report*. University of Toronto, Toronto, Canada.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann Publishers Inc.

Pelizzola, A. (2005). Cluster variation method in statistical physics and probabilistic graphical models. *J. Phys. Pt. A Gen.* 38:R309.

Peterson, C., Anderson, J. (1987). A mean field theory learning algorithm for neural networks. *Complex Syst.* 1(5):995–1019.

Sahu, S. K., Roberts, G. O. (1998). On convergence of the em algorithm and the gibbs sampler. *Statist. Comput.* 9:9–55.

Smyth, P. (1997). Belief networks, hidden markov models, and markov random fields: a unifying view. *Patt. Recog. Lett.* 18(11–13):1261–1268.

Tian, Y., Yang, Q., Huang, T., Ling, C. X., Gao, W. (2006). Learning contextual dependency network models for link-based classification. *IEEE Trans. Knowl. Data Eng.* 18(11):1482–1496.

Todorovic, P. (1992). *An Introduction to Stochastic Processes and Their Applications*. New York: Springer.

Yedidia, J., Freeman, W., Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. Inform. Theor.* 51(7):2282–2312.

Ypma, A., Heskes, T. (2005). Novel approximations for inference in nonlinear dynamical systems using expectation propagation. *Neurocomputing* 69(1–3):85–99.

Zhang, N., Poole, D. (1994). A simple approach to Bayesian network computations. *Proc. Biennial Confer. Can. Soc. Computat. Stud. Intell.* pp. 171–178.