A Neural Network Approach to Approximating MAP in Belief Networks

Yun Peng[a] and Miao Jin
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250
ypeng@csee.umbc.edu

Abstract

Bayesian belief networks (BBN) are a widely studied graphical model for representing uncertainty and probabilistic interdependence among variables. One of the factors that restricts the model's wide acceptance in practical applications is that the general inference with BBN is *NP-hard*. This is also true for the maximum *a posteriori* probability (MAP) problem, which is to find the most probable joint value assignment to all uninstantiated variables, given instantiation of some variables in a BBN. To circumvent the difficulty caused by MAP's computational complexity, we suggest in this paper a neural network approximation approach. With this approach, a BBN is treated as a neural network without any change or transformation of the network structure, and the node activation functions are derived based on an energy function defined over a given BBN. Three methods are developed, they are the hill-climbing style *discrete method*, the *simulated annealing method*, and the *continuous method* based on the mean field theory. All three methods are for BBN of general structures, with the restriction that nodes of BBN are binary variables. In addition, rules for applying these methods to noisy-or networks are also developed, which may lead to more efficient computation in some cases. These methods' convergence is analyzed, and their validity tested through a series of computer experiments with two BBN of moderate size and complexity. Although additional theoretical and empirical work is needed, the analysis and experiments suggest that this approach may lead to effective and accurate approximation for

MAP problems.

Key words: Bayesian belief networks, neural networks, probabilistic inference, simulated annealing, mean field theory, MAP problems.

1 Introduction

Bayesian belief networks (BBN) are a widely studied graphical model for representing uncertainty and probabilistic interdependence among variables in supporting various probabilistic inference [27, 23]. Three related yet distinct types of probabilistic inference have received wide attention in BBN community[27, 25]. They all start with an evidence e, represented as a set of instantiated variables, but differ on what are to be inferred.

- *Belief update*, which computes the posterior probability $P(X_i|e)$ for any uninstantiated variable X_i.
- *Maximum a posteriori probability* (MAP), also known as *belief revision* [27], whose goal is to find the most probable joint value assignment to all uninstantiated variables, in a belief network, given e.
- *Most probable explanation* (MPE), which is similar to MAP, but instead of finding the best joint assignment for all uninstantiated variables, it finds the most probable joint assignment for all "hypothesis" or "explanation" variables, given e. Such a joint assignment forms an explanation for e.

Note that some authors called the second problem MPE, the third MAP, as they consider a joint assignment to all uninstantiated variables forms an explanation [27, 25]. In this paper, we choose to focus on the MAP problem because of its wide practical applications [35, 9, 1] and that it is more difficult to solve than the other two types of inference [34, 2, 25].

It has been well established that general probabilistic inference, including those mentioned above, in BBN is *NP-hard* [6]. Methods for exact solutions of such inference (e.g., *conditioning* [27], *junction-tree* [18], and *bucket elimination* [9]) in BBN of general network structures have exponential complexity, and thus are practically intractable with large, densely connected networks.

To circumvent this problem, researchers have attempted to develop methods for good *approximate* solutions. These include Gibbs sampling [26, 22] and genetic algorithms [1, 32, 5], both of which require a large number of iterations for accurate solutions. Recently, a number of researchers have demonstrated empirically that the belief propagation algorithms based on local message passing [27] that Pearl proposed for exact solutions for belief update and belief revision in singly connected BBN can achieve very good performance for some multi-connected (i.e., loopy) networks [39, 20, 21, 36]. As for MAP problems, it has been established that, regardless the network topology and probability distribution, if the belief propagation algorithm converges to a fix point, then the joint assignment, although not guaranteed to be the globally optimal solution, is the best in a region much larger than the nearest neighborhood. However, the algorithm may not always converge, and there may be a large number of fix points [37].

In this paper, we take another, namely the neural network approach to approximating MAP with BBN. There are several immediate advantages with this approach. First, a large collection of efficient approximation techniques for combinatorial optimization has been developed in the neural network community. They include Hopfield models [13, 14], simulated annealing [17], and mean field approximation [31]. Secondly, since both BBN and neural networks are graphical representations of relationships among variables, adopting methods from neural networks to the belief network setting is natural and appealing. Finally, the localized computation in neural networks makes further speed-up possible through parallel computing.

Researchers have realized the potential benefits of applying neural network methods to belief networks as early as the mid 1980's when neural networks started to gain popularity, and a number of works aimed at exploring this synergism have been reported. For example, several neural network style methods have been developed, based on well-known neural network learning methods, for learning both the belief network structure (the DAG) and the associated conditional probability tables. These include Neal's method based on Boltzmann machine learning [22] and our own learning method that extends the Hebbian rule to belief networks [30]. There have also been numerous reports in the literature of using neural network techniques to solve abductive inference tasks, including diagnosis. Many of these works require the construction of a dedicated neural network from the case data for a given specific task. For example, [11] used the backpropagation training to construct a feedforward network that maps manifestations to plausible causative disorders for diagnostic problem-solving. Others [3] construct high order neural networks from a given BBN to solve the MAP problem. In contrast, all three of our methods carry their neural-style, local computation directly on the given BBN without requiring the construction of an additional neural network. This approach thus utilizes the advantages of two powerful mechanisms, BBN for representing domain knowledge independent of particular inference tasks, and the neural-style computing for efficient problem solving[b].

To help the reader to understand the background of this work, we start with a brief summary of Bayesian belief networks and a formal definition of the MAP problem before presenting the development of our neural network methods.

1.1 Bayesian belief networks

In the most general form, a BBN of n variables is a directed acyclic graph (DAG) of n nodes and a number of arcs, referred to as the network structure [27, 23]. Nodes in a DAG correspond to variables, denoted $X_i, i = 1, \cdots, n$. The lower case x_i denotes an instantiation of X_i, and $x = \{x_1, \cdots, x_n\}$ representing a joint assignment or an instantiation to the set of all variables $X = \{X_1, \cdots, X_n\}$. An arc $< X_i, X_j >$ represents a direct causal or influential relation from X_i to X_j. Also given in a BBN is the conditional probability table $P(X_i|\pi_i)$ for each variable X_i where π_i is the set of X_i's parent nodes. If X_i is a root in the DAG (i.e., a node without any parent), $P(X_i|\pi_i)$ becomes $P(X_i)$. A conditional independence assumption is made for belief networks: $P(X_i|\pi_i, S) = P(X_i|\pi_i)$, where S is any set of variables, excluding X_i, π_i, and all descendants of X_i. It can be shown from the conditional independence assumption that the probability distribution of each variable X_i is only influenced by its parents, its children, and all other parents of its children, which form the *Markov blanket* M_i of node X_i. Let $\bar{X}_i = X - \{X_i\}$ denote the set of all variables $X_j, j \neq i$, then

$$P(X_i|\bar{X}_i) = P(X_i|M_i). \tag{1}$$

The Markov blanket M_i is known to be the minimal set of variables which makes X_i independent of the rest of variables [27].

We pay a special attention to a class of simpler BBN of *binary* nodes, known as the *noisy-or networks* [27, 23, 29]. Instead of using the conditional probabilities $P(X_i|\pi_i)$, a noisy-or network associates a single probability measure, called *causal strength* and denoted c_{ij}, to each arc $<X_i, X_j>$ to capture the degree of uncertainty of the causal relation from X_i to X_j. c_{ij} can be interpreted as the probability of $X_i = 1$ alone, among the parents of X_j, to cause X_j to become 1[c]. In addition to a tremendous reduction of the number of conditional probabilities needed to specify a BBN, the one-to-one correspondence between the arcs and the causal strengths makes noisy-or networks more natural to humans than those using the more general form of the conditional probabilities $P(X_i|\pi_i)$.

One of the most important features of BBN is that the joint probability distribution of X can be computed from the conditional distributions in the network as [27]

$$P(X = x) = \prod_{i=1}^{n} P(x_i|\pi_i). \tag{2}$$

For noisy-or networks, it can be shown [29] that

$$P(x_i = 1|\pi_i) = 1 - \prod_{X_k \in \pi_i} (1 - c_{ki}x_k). \tag{3}$$

That is, $P(x_i = 1|\pi_i)$ is a function of inputs x_k from the parent nodes X_k, weighted with the respective causal strengths c_{ki}. For unspecified x_i, we have[d]

$$P(x_i|\pi_i) = x_i(1 - \prod_{X_k \in \pi_i} (1 - c_{ki}x_k)) + (1 - x_i) \prod_{X_k \in \pi_i} (1 - c_{ki}x_k). \tag{4}$$

The three neural network methods developed in this paper are for general BBN of binary variables, not restricted to noisy-or networks. We also describe how to apply these methods for noisy-or networks, which in some cases may lead to more efficient computation. The restriction to binary variables in this paper is mainly for the sake of presentation clarity, and due to the fact that noisy-or networks allow only binary nodes. We will discuss briefly the issue of extending these methods to multi-valued variables at the end of this paper.

1.2 The MAP problem

Formally, a MAP problem can be stated as follows.

> Let v^+ be a given instantiation of $V \subseteq X$. Then a MAP problem is to find an instantiation u^* of $U = X - V$ such that $P(u^*|v^+) = \max_u P(u|v^+)$. Since v^+ is fixed, u^* also maximizes $P(u, v^+)$.

Figure 1 is about here

The MAP problem can be considered as a mathematical abstraction of the abductive inference, which is in general taken as a process of finding a plausible explanation for a given set of observed facts. Under this interpretation, v^+ represents the set of observed facts for variables V and u^* the most probable explanation of v^+. Abductive inference has applications in various domains, including diagnosis, image understanding, theory formation, and natural language understanding, to name just a few [29].

Exact solutions for MAP problems in singly-connected networks can be obtained by the "max-product" version of Pearl's belief propagation algorithm with polynomial-time complexity [27]. Similarly, the "max-product" version of Junction tree, based on distribution law of max operations, can be used to find exact solutions for MAP problems of general BBN with exponential time complexity [19]. It has been established that, MAP is *NP-hard* if the network is multi-connected [34]. Moreover, approximating MAPs with a constant ratio bound is also shown to be *NP-hard* [2].

When developing approximation methods, MAP seems to be more difficult than belief update [25]. A belief update problem computes $P(u_i|v^+)$, i.e., updating the probability distributions of individual variables u_i, given an instantiation of a set of variables $V = v^+$. By Eq. 1, the computation of $P(u_i|v^+)$ can be easily localized to only involve variables in M_i. This idea has been used in several effective Gibbs sampling methods proposed for belief update [26, 22]. On the other hand, how to localize the computation for MAP, which is the key to the effective neural-style computing, is not that obvious. This is because MAP is a combinatorial optimization problem, and the relationship between local solutions (assigning values to individual variables) and the optimal global solution is much more complicated. This is probably the reason that the loopy propagation performs worse for belief revision (with max-product) than belief update (with sum-product) as reported in [36]

It is these difficulties that motivated us to take a neural network approach. Our hope is that efficient methods for good approximate solutions to MAP problems can be developed by adopting some neural network methods for general combinatorial optimization to the BBN setting.

1.3 The neural network approach

In this approach, the belief network, besides serving as a static representational vehicle for causal and probabilistic knowledge of a domain, is treated as a neural network, and thus a computational vehicle. Under this dual-role schema, the nodes of the network become processing units, arcs become communication channels among neighboring nodes, and the conditional probability distribution associated with each node serves as the inter-node connection weights. Each node updates its activation based on the information from its neighbors and the connection weights. This *local computation* is governed by the node's activation function (also known as the node update rule). A change of activation on a node in turn affects the activations of its neighbors. Therefore, the neural network defines a dynamic system, whose state, described by the collection of activations of all nodes in the network, changes as the activations of individual nodes change. If the activation functions are well chosen, the system can eventually settle down to a stable state approximating an optimal solution to a given inference task.

Although both belief networks and neural networks are graphical representation models, turning a belief network into a neural network that solves a particular problem such as MAP is by no means trivial. One of the fundamental differences between these two models is the way nodes are connected in a graph. Most neural networks can be viewed as fully connected undirected graphs whose links usually do not have semantics, whereas a belief network is an acyclic graph of selected connections where the directed arcs have semantics[e].

An arc $< X_i, X_j >$ in a BBN represents explicitly a direct *causal* relation from X_i to X_j, and implicitly an *abductive* or explanatory relation in the opposite direction from X_j to X_i. Therefore, a node is influenced by its parents and children in different ways. These relationships cannot be captured by the simple neighborhood relations in an ordinary neural network where all neighbors affect a node in much the same way. The questions we need to answer here are, can the MAP problems be solved by local computation at each node, and if so, what kinds of information does a node need from its parents and children, respectively, and how can we synthesize the received information at each node to make appropriate local decision? These questions will be answered in the subsequent sections with the development of three neural network methods for the MAP problems.

The rest of this paper is organized as follows. Section 2 formulates the MAP problem as a neural network optimization problem by defining an energy function for a given BBN. Section 3 presents the discrete neural network method for MAP problems, which is fast but often inaccurate. Section 4 then extends the discrete method to a simulated annealing method to improve the solution accuracy. The continuous method is developed in Section 5 to speed up the annealing process based on the mean field theory. Section 6 describes a post process method called *resettling* that can be used to further improve the solution accuracy with a moderate increase in the computational cost. Section 7 describes computer experiments in which the proposed methods were tested with two non-trivial BBN of moderate size and density. Finally, Section 8 concludes with discussions and suggestions for future research directions.

2 A Neural Network Formulation of MAP Problems

In this section, we discuss how we treat a belief network as a neural network for solving MAP. Different suggestions have been made in the literature to connect belief networks and neural networks. The most noticeable among them are the neural style representations of sigmoid and noisy-or belief networks made by Neal and followed by others [22, 33]. These representations allow one to treat each node of a network as a processing unit and conduct neural style inference and learning similar to Boltzmann machine. In contrast, our approach works directly on the conventional BBN representation without requiring the re-formulation of the network. In our approach, each node X_i in BBN now becomes a processing unit with activation $x_i(t)$ at time t. For a MAP problem, activations of nodes in V are fixed (clamped) to their given instantiations during the entire computation, while activations of nodes in U are free to change. In the discrete and simulated annealing methods (to be developed in Sections 3 and 4), node activations can take only binary values 0 and 1. Therefore, different activations represent different instantiations for a given node. In the continuous method (Section 5), however, activations take real values between 0 and 1, $u_i(t)$ thus can be interpreted as a partial instantiation of node U_i at time t. As will be seen in Section 5, $u_i(t)$ approximates the average or mean value of U_i. A node is only connected to its direct causal predecessors (parents) and successors (children) as specified by the arcs in a BBN. Its activation therefore influences and is influenced by the activations of these neighboring nodes. Moreover, as can be seen shortly, activations from parent nodes and child nodes play different roles and are used in different ways in determining the activation of a given node.

If the set V is empty, then $U = X$, and the MAP problem becomes the problem of finding the most probable unconditional joint assignment to all variables in X, i.e., to find x^* such that $P(x^*) = \max_x P(x)$. For simplicity, we derive our node activation update rules with empty V. The resulting activation rules can be applied to problems with non-empty V by permanently clampping $V_i \in V$ to v_i^+, its given instantiations.

Analogous to the development of many neural network methods, we define an energy function for MAP problems that will serve as a basis for our three methods. At any time t, the state of the system is represented by $x(t)$, the current activations of all variables in the system. The energy function for the system at time t is then defined as

$$E(x(t)) = -\ln P(x(t)). \tag{5}$$

To maximize $P(x)$, which is our objective, is to minimize $E(x)$, the system's energy.

Next we define a quantity r_i, which will be seen to play a central role in local decisions for our three methods. Consider two states related to state $x(t)$ with respect to a chosen node X_i. These two states, denoted $(x_i = 1, \bar{x}_i(t))$ and $(x_i = 0, \bar{x}_i(t))$, respectively, are the same as $x(t)$ except X_i is set to 1 in one state and 0 in another. Here $\bar{x}_i = \{x_j, j \neq i\}$ denotes the set of all variables, excluding x_i. Then, the energy difference between the two states is

$$\begin{aligned} \Delta E_i(x(t)) &= E(x_i = 1, \bar{x}_i(t)) - E(x_i = 0, \bar{x}_i(t)) \\ &= -\ln \frac{P(x_i = 1, \bar{x}_i(t))}{P(x_i = 0, \bar{x}_i(t))}. \end{aligned} \tag{6}$$

We define the ratio of the probabilities of the two states

$$r_i(t) = \frac{P(x_i = 1, \bar{x}_i(t))}{P(x_i = 0, \bar{x}_i(t))} = \frac{P(x_i = 1|\pi_i(t))}{P(x_i = 0|\pi_i(t))} \prod_{X_j \neq X_i} \frac{P(x_j(t)|\pi_j(t))_{x_i=1}}{P(x_j(t)|\pi_j(t))_{x_i=0}} \tag{7}$$

where the second equality comes from Eq. 2, and $(.)_{x_i=1}$ and $(.)_{x_i=0}$ indicate that x_i is set to 1 and 0, respectively, in π_j. By Eq. 6, we have

$$\ln r_i(t) = -\Delta E_i(t). \tag{8}$$

Note that if $r_i(t) > 1$ then $\Delta E_(t) < 0$, and if $r_i(t) < 1$ then $\Delta E_(t) > 0$. Therefore, like a partial derivative, $r_i(t)$ determines the direction of change of energy with respect to the instantiation of X_i when all other variables are kept unchanged.

The computation of $r_i(t)$ is not as involved as it appears to be. Note that all variables, except x_i, have the same instantiations in both the numerator and the denominator of Eq. 7. Therefore we can simplify Eq. 7 by canceling all conditional probabilities in the two products except those that involve x_i. The remaining terms include $P(x_i = 1|\pi_i(t))$ and $P(x_i = 0|\pi_i(t))$, as well as conditional probabilities of the children of X_i. Then we have

$$r_i(t) = \frac{P(x_i = 1|\pi_i(t))}{P(x_i = 0|\pi_i(t))} \prod_{X_j \in Child(X_i)} \frac{P(x_j(t)|\pi_j(t))_{x_i=1}}{P(x_j(t)|\pi_j(t))_{x_i=0}}, \tag{9}$$

By Eq. 9, the computation of $r_i(t)$ is completely local, involving only the parents and children of X_i, which are connected by causal arcs with X_i in the network. To compute $r_i(t)$, different types of information are required from the parents and children of node X_i, respectively, reflecting the different semantics of the two directions of an arc (causal and abductive). For $X_k \in \pi_i$, we need its current activation $x_k(t)$, then $P(x_i = 1|\pi_i(t))$ and $P(x_i = 0|\pi_i(t))$ can be computed from X_i's local conditional probability table. For $X_j \in Child(X_i)$, we need $P(x_j(t)|\pi_j(t))_{x_i=1}/P(x_j(t)|\pi_j(t))_{x_i=0}$, which can be computed at X_j with its conditional probability table, after obtaining the activations of its parents. Therefore, as expected, $r_i(t)$ is completely determined by activations of all nodes in X_i's Markov blanket M_i. From Eq. 1 and the definition of r_i (Eq. 7), we have

$$r_i(t) = P(x_i = 1|M_i(t))/(1 - P(x_i = 1|M_i(t))). \tag{10}$$

Eq. 9 gives the operational rule to compute $r_i(t)$ from activations of nodes in $M_i(t)$.

For noisy-or networks, we substitute Eq. 4 into Eq. 9, and obtain

$$\begin{aligned} r_i(t) = {} & \frac{1 - \prod_{X_k \in \pi_i}(1 - c_{ki}x_k(t))}{\prod_{X_k \in \pi_i}(1 - c_{ki}x_k(t))} \prod_{X_j \in Child(X_i)} \\ & \frac{x_j(t)(1 - \prod_{X_k \in \pi_j}(1 - c_{kj}x_k(t)))_{x_i=1} + (1 - x_j(t)) \prod_{X_k \in \pi_j}(1 - c_{kj}x_k(t))_{x_i=1}}{x_j(t)(1 - \prod_{X_k \in \pi_j}(1 - c_{kj}x_k(t)))_{x_i=0} + (1 - x_j(t)) \prod_{X_k \in \pi_j}(1 - c_{kj}x_k(t))_{x_i=0}}. \end{aligned} \tag{11}$$

Figure 2 below depicts information flows from the parents and children of X_i when computing $r_i(t)$.

Figure 2 is about here

---

3 The Discrete method

Our discrete method works in a way similar to the discrete Hopfield model [13]. It starts with an arbitrary initial instantiation $x(0)$ of X. At each step, a *single* node X_i is randomly selected for updating its activation. The update is done by setting the activation of X_i to the value (0 or 1) that corresponds to a lower system energy, while values of all other variables are kept constant. Specifically, the node activation function is defined as follows. At time t,

$$x_i(t+1) = \begin{cases} 1 & \text{if } r_i(t) > 1 \\ x_i(t) & \text{if } r_i(t) = 1 \\ 0 & \text{if } r_i(t) < 1 \end{cases} \tag{12}$$

It is clear from Eqs. 6 and 9 that $x_i(t+1)$ is set to 1 only if $r_i(t) > 1$,which implies $\Delta E_i(t) < 0$, and thus E is lower than setting $x_i(t+1)$ to 0. Similarly, E is lower when $x_i(t+1)$ is set to 0 if $r_i(t) < 1$. Therefore, $E(t)$ monotonically decreases with each update. This, combined with the fact that $E(x(t)) \geq 0$ is bounded from below, indicates $E(t)$ is a Lyapunov function. It is a well-known result from the study of dynamic systems that, if a Lyapunov function can be defined for a system, then the system is guaranteed to converge to a stable state with a local minimum function value. With $E(x(t))$ as its Lyapunov function, the system defined by the discrete method will move in a hill-climbing fashion toward a (local) minimum energy state $E(x)$ (and thus a local maximum $P(x)$). This state is a stable state representing a local optimal solution to the given MAP problem.

The outline of the algorithm for the discrete method goes as follows.

algorithm for the discrete method

begin

1. initialization:

   x is set to an arbitrary instantiation

   $t \leftarrow 0$

2. repeat until the system reaches a stable state

   (a) randomly pick a node X_i

   (b) compute $r_i(t)$ by Eq. 9

   (c) set x_i according to Eq. 12

   (d) $t \leftarrow t + 1$

end

The termination condition in Step 2 can be determined by periodically calling a procedure that checks if none of the nodes will change its state according to Eqs. 9 and 12.

4 The Simulated Annealing method

Like all hill-climbing style computation, the system of the discrete method described in the previous section will be trapped in the stable state it reaches during the climbing, which may be a locally but not globally optimal solution. This problem becomes serious if a MAP problem has many solutions that are only locally optimal. Simulated annealing (SA), a well-known stochastic method, offers an elegant way for a dynamic system to escape from a stable state and continue searching for other, potentially better, solutions [17]. Our discrete method can be easily converted to a SA method. To do so, we only need to change the way $r_i(t)$ is used from a deterministic to a stochastic fashion, and allow the system to slowly cool down from an initial high (artificial) temperature. In the discrete method, $r_i(t)$ is used to compute the activation of X_i at $t+1$. In the SA method, the same ratio is used to determine the *probability* of X_i to have a particular activation at $t+1$. Specifically, when X_i is selected to update at time t, its activation is assigned to 1 with the probability

$$\begin{aligned} P(X_i \leftarrow 1 | T(t)) &= (1 + \exp(\Delta E_i(t)/T(t)))^{-1} \\ &= (1 + r_i(t)^{-\frac{1}{T(t)}})^{-1} \end{aligned} \tag{13}$$

where $T(t)$ is the temperature at t, and $r_i(t)$ is computed by Eq. 9. Here the first equality is the general equation of simulated annealing, indicating that $P(X_i \leftarrow 1|T(t))$ is dependent on the energy difference of the system when X_i is assigned to 1 or 0. The second equality comes from Eq. 8. From Eqs. 8 and 13, when $r_i(t) > 1$ ($\Delta E_i(t) < 0$), X_i is not set to 1, but has a probability of $(1 + r_i(t)^{-1/T(t)})^{-1} > 0.5$ being set to 1. At the same time, X_i has a non-zero probability of $1-(1+r_i(t)^{-1/T(t)})^{-1})^{-1} < 0.5$ being set to 0. Similar argument holds for the case $r_i(t) < 1$. Therefore, at any given time there is always a non-zero probability that $P(x(t))$ will decrease, although this probability is smaller than the probability that $P(x(t))$ will increase. The existence of the non-zero probability that $P(x(t))$ will decrease at each iteration is crucial for the SA method. It is this probability that provides the system the opportunity to move out of local optima. This probability is large at the early time of annealing (with large T), and diminishes when T becomes very small. With Eq. 13, x_i's become random variables, and the system becomes stochastic. It has been proven [10] that when the system arrives at thermal equilibrium at a given temperature, its states obey Boltzmann-Gibbs distribution,

$$P(X = x|T) \quad = \quad \exp(-E(x)/T)/Z \tag{14}$$

where the normalization factor $Z = \sum_x \exp(-E(x)/T)$, which is a constant at any given T at equilibrium. Since the parameter t is used to index the steps in simulating the annealing process, it is immaterial, and thus omitted, when describing the distribution of the system states at a given temperature. Furthermore, by Eq. 5,

$$\begin{aligned} P(X = x|T) &= \exp(\ln P(x)/T)/Z \\ &= P(x)^{\frac{1}{T}}/Z. \end{aligned} \tag{15}$$

When the system reaches equilibrium at temperature $T = 1$, $P(X = x|T = 1) = P(x)$ because $Z = \sum_x P(x) = 1$. That is, the probability distribution of the system's states becomes the same as that of the original BBN. At this temperature, node activation updating rule Eq. 13 becomes

$$P(X_i \leftarrow 1|T(t)=1) = (1 + r_i(t)^{-1})^{-1} = P(x_i = 1|M_i(t)), \tag{16}$$

where the last equality come from Eq. 10. Eq. 16 is equivalent to Gibbs sampling, as suggested in [26, 22]. However, this does not hold when $T \neq 1$. In particular, when $T(t)$ slowly approaches zero following a proper cooling schedule, the Boltzmann-Gibbs distribution guarantees that the probability that the system is at a global optimal state (with globally minimum $E(x(t)$ and maximum $P(x(t)|T(t)))$ is infinitely higher than at any other states. It is this property that gives the simulated annealing method the theoretical capability of global optimization.

The outline of the algorithm for the SA method is given below where $T_0 \gg 0$ and T_∞ close to 0 are the start and termination temperatures.

algorithm for the SA method

begin

1. initialization:

   x is set to an arbitrary instantiation

   $T \leftarrow T_0$

   $t \leftarrow 0$

2. repeat until the system reaches equilibrium

   (a) randomly pick a node X_i

   (b) compute $r_i(t)$ by Eq. 9

   (c) set x_i to 1 with the probability $(1 + r_i(t)^{-1/T(t)})^{-1}$

   (d) $t \leftarrow t + 1$

3. if $T \leq T_\infty$ then stop

   else

(a) reduce T by a cooling schedule

(b) go to step 2.

end

It is expensive to test if a system of large BBN has reached equilibrium (the condition to end the inner loop of Step 2). We approximate this by letting the loop repeat for a large number of times. There are several commonly used cooling schedules for SA methods. The one we used in the simulation experiments in this paper is simply $T = T * \delta$ where $0 < \delta < 1$.

5 The Continuous method

Simulated annealing may significantly improve the solution accuracy, but the annealing process often takes a long time to complete. This is because the cooling must be very slow in order to avoid the system being stuck in a local minimum energy state prematurely, and the system takes a long time to reach equilibrium at each temperature [10]. To overcome this difficulty, we turn to *mean field theory* (MFT), which allows us to approximate the stochastic system of the SA method by a deterministic system. The result is our continuous method in which the stochastic binary nodes in the SA method are replaced by deterministic continuous ones.

Mean field theory has its origin in theoretical physics [24]. It deals with a physical system of many interacting particles where the interactions between particles make the calculation of the precise behavior of the system practically impossible. The rationale behind the mean field approximation is that, since each particle in such a system is influenced by many other particles, each influence is weak and the total influence is roughly additive [15]. Therefore, the detailed state fluctuation of each particle can be approximated by the average (mean) values of the particles' states. MFT was first introduced to the neural network community by

Peterson and Anderson for approximating the time consuming Boltzmann machine learning [31]. It was also applied to analyze the celebrated Hopfield models, leading to the conclusion that the continuous Hopfield model [14] is a mean field approximation of his discrete model [4].

The application of MFT to belief networks was pioneered by Saul, Jordan, and Jaakkola in their work on approximating the probability distribution of sigmoid belief networks [33]. They later extended their work to compute probability bounds for noisy-or networks, and pointed out that their mean field formulations can be considered a simplest form of a more general approximation framework known as variational method [16]. Their work aims at approximating belief update, not belief revision or MAP problems. Haft *et al* later developed a more general mean field model which is not restricted to any particular type of network structure [12]. Our continuous method is a further development from the model of Haft *et al* in several important aspects. Before presenting the derivation of our mean field equation, it is helpful to give a brief summary of Haft *et al*'s work.

The main objective of their work is to approximate $P(X)$, a complex probability distribution such as that represented by a BBN, by a simpler distribution $Q(X)$ over the same domain. In particular, variables X_i in $Q(X)$ are assumed to be independent of each other, i.e., $Q(X = x) = \Pi_{i=1}^{n} Q(x_i)$. The closeness of the two distributions is measured by the cross-entropy (Kullback-Leibler distance) [8]

$$D(Q||P) \quad = \quad \sum_x Q(x) \ln \frac{Q(x)}{P(x)} \equiv \left\langle \ln \frac{Q(x)}{P(x)} \right\rangle_{Q(X)}, \tag{17}$$

where $\langle . \rangle_{Q(X)}$ denotes the expectation with respect to the distribution $Q(X)$. $D(Q||P)$ is non-negative and is zero if and only if $Q(X) = P(X)$. It might not be possible to find the $Q(X)$ that minimizes $D(Q||P)$ to zero, but a good approximation may be obtained. Haft *et al* propose an iterative, gradient descent approach for such an approximation. Each time, a randomly selected $Q(X_i)$ is updated, according to a mean field equation, to minimize

$D(Q||P)$ while $Q(X_j)$ for all $j \neq i$ are fixed. The mean field equation is derived by solving the following equation,

$$\frac{\partial}{\partial Q(x_i)} \left[D(Q||P) - \lambda(\sum_{x_i} Q(x_i) - 1) \right] = 0, \tag{18}$$

where the Lagrange parameter λ is used to accommodate the normalization constraint $\sum_{x_i} Q(x_i) = 1$. Eq. 18 can be solved straightforwardly if we split $D(Q||P)$ using the relations $P(X) = P(\bar{X}_i)P(X_i|\bar{X}_i)$ and $Q(X) = Q(\bar{X}_i)Q(X_i)$. Solving this equation we have

$$Q(x_i) = \frac{1}{\exp(1-\lambda)} \exp\langle \ln P(x_i|\bar{X}_i) \rangle_{Q(\bar{X}_i)}. \tag{19}$$

The Lagrange parameter λ or the normalization constant $\exp(1-\lambda)$ can be computed easily

$$\exp(1-\lambda) = \sum_{x_i} \exp\langle \ln P(x_i|\bar{X}_i) \rangle_{Q(\bar{X}_i)}. \tag{20}$$

where the summation is over all possible values of x_i.

Eq. 19 is the unique solution to Eq. 18. It corresponds to the global minimum of $D(Q||P)$ with respect to $Q(X_i)$, given the current estimates of $Q(X_j)$, $j \neq i$. Eq. 19 is viewed as a mean field equation because $Q(X_i)$ is based on the mean value of X_i, given $\bar{X}_i$, and is determined by $Q(X_j)$, the mean values of all other variables X_j. Eq. 19 is in a general form because no model assumptions are made. When applied to BBN, since X_i is only affected by variables in its Markov blanket $M_i \subseteq \bar{X}_i$, Eq. 19 can be simplified to

$$Q(x_i) \propto \exp\langle \ln P(x_i|M_i) \rangle_{Q(M_i)}. \tag{21}$$

The mean field equations Eq. 19 and 21 of Haft *et al* can be applied directly to some inference BBN tasks such as belief update. However, several problems prevent it from being applied to effectively solving MAP problems. First of all, these equations are used to find a *distribution* $Q(X)$ that approximates another distribution $P(X)$, they do not provide a mechanism directly applicable to finding a *most probable instantiation* to X under the distribution, as required by the MAP problems. Secondly, even with simplified Eq. 21 for BBNs, the calculation of expectation $\langle \ln P(x_i|M_i)\rangle_{Q(M_i)}$ involves calculating $P(x_i|M_i)Q(M_i)$ for *every possible instantiation* of M_i. Since the total number of such instantiations is exponential to $|M_i|$, the process is computationally expensive when the network is bushy.

In what follows, we derive our mean field equations for MAP problems by extending Haft *et al*'s model to overcome these difficulties. The first difficulty is overcome by incorporating annealing process into the mean field approximation so that the system can settle down to a state corresponding to a (locally) most probable instantiation to X when the temperature approaches zero. The second difficulty is overcome by introducing an additional approximation assumption so that the computation of $\exp\langle \ln P(x_i|M_i)\rangle_{Q(M_i)}$ in Eq. 21 is simplified.

Recall that in the SA method, $P(X|T) = \exp(-E(X)/T)/Z$ obeys Boltzmann-Gibbs distribution at any given temperature T at equilibrium. Our goal now is to approximate this distribution by $Q(X|T)$ for all T while T is gradually reduced to approach 0. For network of binary variables, Eq. 20 becomes

$$\exp(1-\lambda) = \exp\langle \ln P(x_i=1|\bar{X}_i)\rangle_{Q(\bar{X}_i)} + \exp\langle \ln P(x_i=0|\bar{X}_i)\rangle_{Q(\bar{X}_i)},$$

and Eq. 19 becomes

$$\begin{aligned} Q(x_i=1) &= (1+(\frac{\exp\langle \ln P(x_i=1|\bar{X}_i)\rangle_{Q(\bar{X}_i)}}{\exp\langle \ln P(x_i=0|\bar{X}_i)\rangle_{Q(\bar{X}_i)}})^{-1})^{-1} \\ &= (1+(\frac{\exp\langle \ln P(x_i=1,\bar{X}_i)\rangle_{Q(\bar{X}_i)}}{\exp\langle \ln P(x_i=0,\bar{X}_i)\rangle_{Q(\bar{X}_i)}})^{-1})^{-1}, \end{aligned} \tag{22}$$

where the last equality comes from the fact that $\langle \ln P(x_i|\bar{X}_i)\rangle_{Q(\bar{X}_i)} = \langle \ln P(x_i,\bar{X}_i)\rangle_{Q(\bar{X}_i)} - \langle \ln P(\bar{X}_i)\rangle_{Q(\bar{X}_i)}$. We introduce annealing into the mean field equations by replacing $P(X)$ and $Q(X)$ by $P(X|T)$ and $Q(X|T)$, respectively, Eq. 22 is then changed to

$$Q(x_i=1|T) = (1+(\frac{\exp\langle \ln P(x_i=1,\bar{X}_i|T)\rangle_{Q(\bar{X}_i|T)}}{\exp\langle \ln P(x_i=0,\bar{X}_i|T)\rangle_{Q(\bar{X}_i|T)}})^{-1})^{-1}. \tag{23}$$

From Eqs. 15 and 2, we have

$$\ln P(x_i,\bar{X}_i|T) = \ln P(x_i,\bar{X}_i)^{\frac{1}{T}}/Z$$

$$= \ln \prod_{X_j \in X} (P(x_j|\pi_j))^{\frac{1}{T}}/Z$$
$$= \frac{1}{T} \sum_{X_j \in X} \ln P(x_j|\pi_j) - \ln Z. \quad (24)$$

Substituting Eq. 24 into Eq. 23, we have

$$\begin{aligned}
&Q(x_i = 1|T) \\
&= (1 + (\frac{\exp\langle \frac{1}{T}\sum_{X_j\in X} \ln P(x_j|\pi_j) - \ln Z\rangle_{Q(\bar{X}_i|T),x_i=1}}{\exp\langle \frac{1}{T}\sum_{X_j\in X} \ln P(x_j|\pi_j) - \ln Z\rangle_{Q(\bar{X}_i|T),x_i=0}})^{-1})^{-1} \\
&= (1 + (\prod_{X_j\in X} \frac{\exp(\frac{1}{T}\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=1}/Z)}{\exp(\frac{1}{T}\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=0}/Z)})^{-1})^{-1} \\
&= (1 + (\prod_{X_j\in X} \frac{\exp\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=1}}{\exp\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=0}})^{-\frac{1}{T}})^{-1} \\
&= (1 + (\frac{\exp\langle \ln P(x_i=1|\pi_i)\rangle_{Q(\bar{X}_i|T)}}{\exp\langle \ln P(x_i=0|\pi_i)\rangle_{Q(\bar{X}_i|T)}} \prod_{X_j\in ChildX_i} \frac{\exp\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=1}}{\exp\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=0}})^{-\frac{1}{T}})^{-1}. \quad (25)
\end{aligned}$$

The second and third equalities in Eq. 25 come from standard expectation operations, the last equality is again from the fact that the numerator and demoninator are same, and thus can cancel each other for all $X_j \in X$ except X_i and its children.

Eq. 25 is our mean field equation with annealing. As with Haft *et al*, the system with Eq. 25 will converge at a given T with $Q(X|T)$ approximating $P(X|T)$. Whether the system converges to x^*, the globally optimal solution, depends on the difference $D(Q(x|T)||P(X|T))$. If the difference is very small, then $Q(x^*|T)$ is much higher than $Q(x|T), x \neq x^*$ because $P(x^*|T)/P(x|T)$ approaches infinite as T approaching zero, and thus the system will very likely converge to x^*. Otherwise, the system will converge to some suboptimal solution.

Eq. 25 requires computing expectations $\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=1}$ and $\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=0}$ for every $X_j \in \pi_i$, which are still time-consuming for large π_j. To simplify this computation, an additional approximation is made. By Jensen's inequality [8], $\langle \ln P(x_j|\pi_j)\rangle_{Q(X|T)} \leq \ln\langle P(x_j|\pi_j)\rangle_{Q(X|T)}$. We assume that

$$\langle \ln P(x_j|\pi_j)\rangle_{Q(X|T)} \approx \ln\langle P(x_j|\pi_j)\rangle_{Q(X|T)}. \quad (26)$$

This assumption can be justified by the following arguments. First, the approximation errors caused by Jensen's inequality to both the numerator and the denominator in Eq. 25 can partially cancel each other since both are an increase from their respective original values. Secondly, the errors will be reduced when T approaches zero. This can be argued as follows. As will be shown shortly, if there is no tie (i.e., no two distinct states have the same system energy), then the system will converge to a local minimum energy state x^{min} in which each $\hat{x}_i$ is very close to one of its extreme values (1 or 0). Therefore, $\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i} = \sum_{\bar{x}_i} P(x_j|\pi_j)Q(\bar{x}_i, x_i)$ is strongly dominated by the *single* term in the summation in which $\bar{X}_i = \bar{x}_i^{min}$ with $\bar{x}_i^{min}$ consistent with x^{min}. Thus, when $T \to 0$, $\langle \ln P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i} \approx (\ln P(x_j|\pi_j))_{\bar{x}_i^{min},x_i} = \ln\langle P(x_j|\pi_j))\rangle_{Q(\bar{X}_i=x_i^{min}),x_i} \approx \ln\langle P(x_j|\pi_j))\rangle_{Q(\bar{X}_i|T),x_i}$. The situation with ties will be discussed later.

With the new assumption of Eq. 26, Eq. 25 changes to

$$
\begin{aligned}
&Q(x_i = 1|T) \\
&\quad\approx\ (1 + (\frac{\exp\ln\langle P(x_i=1|\pi_i)\rangle_{Q(\bar{X}_i|T)}}{\exp\ln\langle P(x_i=0|\pi_i)\rangle_{Q(\bar{X}_i|T)}} \prod_{X_j \in ChildX_i} \frac{\exp\ln\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=1}}{\exp\ln\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=0}})^{-\frac{1}{T}})^{-1} \\
&\quad=\ (1 + (\frac{\langle P(x_i=1|\pi_i)\rangle_{Q(\bar{X}_i|T)}}{\langle P(x_i=0|\pi_i)\rangle_{Q(\bar{X}_i|T)}} \prod_{X_j \in Children(x_i)} \frac{\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=1}}{\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=0}})^{-\frac{1}{T}})^{-1}. \qquad (27)
\end{aligned}
$$

We now define

$$
\hat{r}_i(t) = (\frac{\langle P(x_i=1|\pi_i)\rangle_{Q(\bar{X}_i|T)}}{\langle P(x_i=0|\pi_i)\rangle_{Q(\bar{X}_i|T)}} \prod_{X_j \in Children(x_i)} \frac{\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=1}}{\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i=0}}. \qquad (28)
$$

This ratio for our mean field method is the same as $r_i(t)$ define by Eq. 7 except that probabilities of $P(x_i|\pi_i)$ and $P(x_j|\pi_j)_{x_i}$ are replaced by their expectations (mean values). Changing $\approx$ to $=$, the mean field equation Eq. 27 then becomes

$$
Q(x_i = 1|T) = (1 + \hat{r}_i(t)^{-\frac{1}{T}})^{-1}. \qquad (29)
$$

Eq. 28 can be viewed as the annealed mean field version of $r_i(t)$ (Eq. 7) for both discrete and SA methods, and Eq. 29 the annealed mean field version of node update rule of the SA method (Eq. 13).

Computation of expectations in Eq. 29 can be further simplified for noisy-or networks. By Eq. 3, we have

$$\begin{aligned}
\langle P(x_i = 1|\pi_i)\rangle_{Q(\bar{X}_i|T)} &= \langle 1 - \prod_{X_k \in \pi_i} (1 - c_{ki} x_k)\rangle_{Q(\bar{X}_i|T)} \\
&= 1 - \prod_{X_k \in \pi_i} (1 - c_{ki} \langle x_k \rangle_{Q(\bar{X}_i|T)}) \\
&= 1 - \prod_{X_k \in \pi_i} (1 - c_{ki} \hat{x}_k). \qquad (30)
\end{aligned}$$

The second equality comes from the fact that all x_k are independent under $Q(.)$. For brevity, in the last equality we denote the mean value $\langle x_k \rangle_{Q(\bar{X}_i|T)}$ by $\hat{x}_k$. Similarly, we have

$$\langle P(x_i = 0|\pi_i)\rangle_{Q(\bar{X}_i|T)} = \prod_{X_k \in \pi_i} (1 - c_{ki} \hat{x}_k). \qquad (31)$$

For $j \neq i$, we have

$$\begin{aligned}
\langle P(x_j|\pi_j)\rangle_{Q(\bar{X}_i|T),x_i} &= Q(x_j = 1)(1 - \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k))_{x_i} + Q(x_j = 0) \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k)_{x_i} \\
&= \hat{x}_j(1 - \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k))_{x_i} + (1 - \hat{x}_j) \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k)_{x_i} \qquad (32)
\end{aligned}$$

where in the last equality we denote $Q(x_j = 1)$ by $\hat{x}_j$, the mean value of X_j since for binary variables, $Q(x_j = 1) = \langle x_j \rangle$.

Substituting Eqs. 30, 31, and 32 into 29, we have

$$\hat{x}_i(t) = (1 + \hat{r}_i(t)^{-\frac{1}{T}})^{-1}, \qquad (33)$$

where

$$\begin{aligned}
\hat{r}_i(t) &= \frac{1 - \prod_{X_k \in \pi_i} (1 - c_{ki}\hat{x}_k(t))}{\prod_{X_k \in \pi_i} (1 - c_{ki}\hat{x}_k(t))} \times \prod_{X_j \in Child(X_i)} \\
&\frac{\hat{x}_j(t)(1 - \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k(t))_{x_i=1} + (1 - \hat{x}_j(t)) \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k(t))_{x_i=1}}{\hat{x}_j(t)(1 - \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k(t))_{x_i=0} + (1 - \hat{x}_j(t)) \prod_{X_k \in \pi_j} (1 - c_{kj}\hat{x}_k(t))_{x_i=0}}. \qquad (34)
\end{aligned}$$

Also note that Eq. 34 for $\hat{r}_i(t)$ looks very much like Eq. 11 for $r_i(t)$ used in both discrete and SA methods for noisy-or networks. Again, here all binary x_k's are replaced by their mean values $\hat{x}_k$.

To establish the convergence of the continuous method, consider the system's behavior when $T \to 0$. Since $\hat{r}_i(t)$ is bounded, then we have $\hat{x}_i(t) \approx 1$ if $\hat{r}_i(t) > 1$ and $\hat{x}_i(t) \approx 0$ if $\hat{r}_i(t) < 1$ by Eq. 33 because $T \to 0$. The system now behaves in the same way as the discrete mouthed, and likewise a node switches its value (between 1 and 0) only if doing so reduces the system's energy. However, when $T \to 0$, the energy is been reduced toward a local minimum, changing the state for any variable between 0 and 1 by Eq. 33 can only further reduce the energy by an infinitesimal amount. If there does not exist a tie, i.e., no two distinct system states have the same energy, then no variable can switch from its current state to another state (say from 1 to 0) because doing so would have caused the system energy to reduce by an amount greater than some constant, determined by the energy difference between two states $(x_i = 1, \bar{x}_i)$ and $(x_i = 0, \bar{x}_i)$. Therefore, the system converges. If the current system state is tied to some other states, then oscillations may occur between them because the energy does not change much when the system oscillates. To avoid such oscillation, we can break the ties as follows. A common practice to avoid system oscillation in neural networks is to update the system state one variables at a time as we do with the discrete and SA methods. This would break a tie between two states which differ by more than one variable, and oscillation will not occur because switching state of any single variable would have moved the system to a new state with energy higher than a constant. When the two states differ by only one variable, say x_i, then we have $r_i = P(x_i = 1|M_i)/P(x_i = 0|M_i) = 1$. Since $\hat{r}_i$ can approach 1 alternately from both directions, it may fluctuate around 1, causing $\hat{x}_i$ to oscillate between 0 and 1. However, since $r_i = 1$, assigning $\hat{x}_i$ to either 1 or 0 would not change the system's energy, if other variables stay the same. Therefore we can break this kind of ties by arbitrarily assigning $\hat{x}_i$ to either 1 or 0 when it is observed to oscillate at the end of the process.

The algorithm for the continuous method then goes as follows. Since the two BBN used in our experiment have randomly generated probabilities, they would be very unlikely to contain ties. Therefore, to speed up the simulation, we choose to update the activations of all variables simultaneously at each iteration. In practical use, we may need to update one variable at a time, and break tie if oscillation is observed at the end of the execution as discussed above.

algorithm for the continuous method

begin

1. initialization:

   each $\hat{x}_i$ is set to a small random positive value

   $T \leftarrow T_0$

   $t \leftarrow 0$

2. repeat until $|\hat{x}_i(t) - \hat{x}_i(t-1)|$ is very close to 0 for all x_i

   (a) compute $\hat{r}_i(t)$ by Eq. 34

   (b) set $\hat{x}_i(t)$ according to Eq. 33

   (c) $T = T * \delta$ // $\delta < 1$ is the cooling factor

   (d) $t \leftarrow t + 1$

end

6 Resettling Process

Due to the approximation error, the assignment $X = \hat{x}$ obtained by the continuous method may not always be the same as the global optimal one x^*. In what follows, we describe a process, called resettling [29], which can be employed to further improve the solution accuracy at a cost. In general, there is no way to tell whether $\hat{x} \neq x^*$ because the true optimal x^* is unknown. However, if indeed $\hat{x} \neq x^*$, then at least one node, say k, has gone wrong when computed by the continuous method. That is, the correct assignment to X_k should be $1 - \hat{x}_k$, not $\hat{x}_k$. Therefore, a better solution might be found by permanently clamping one variable X_k to the value of $1-\hat{x}_k$ while allowing all others to change according to the continuous method. Since we don't know which of these n variables may have been assigned incorrect values, we need to repeat this process n times, each for a particular X_k. This leads to the following *resettling process*.

After $\hat{x}^0$ is computed by the continuous method, n more computations are performed. For the k^{th} of the n computations, we randomly assign an initial value to each of the variables except X_k, which is assigned and *clamped* to $1-\hat{x}_k^0$ through the computation. The computation then proceeds according to the continuous method until the system settles down to a stable state $\hat{x}^k$ (thus the name of re-settling). After the n resettling computations, the best among the $n+1$ results $\hat{x}^k, k=0,\cdots n$, according to their respective $P(\hat{x}^k)$, is selected as the final answer. Note that before $\hat{x}$ is computed, there are a total of $2n$ subspaces of dimension $n-1$ since each of the n variables has two instantiations. With the knowledge of $\hat{x}$, the resettling process has to only look into n of them. The price we pay for a possibly better result than $\hat{x}^0$ is the increase of computation time by a factor of n.

Note that this resettling process is different, and in general more powerful, than the iterative improvement based on nearest neighbor search as suggested by others [25] in which the current solution is *compared* with all its neighbors (differing by only one variable instantiation) and moved to the best neighbor if it is better. In our resettling, each of the $n-1$ subspaces, defined by clamping one variable to its opposite value with respect to the current solution, is actually *search* by letting the system resettle into a (sub)optimal state by Eq. 33. Also note that, if time permits, additional runs of resettling can be conducted for further improvement until no better solution can be found. However, diminished return is expected with additional resettling, and for this reason, we did not pursue this in our experiments.

7 Experimental Results

A series of computer experiments of limited scope were conducted. The main purpose of these experiments was to test the validity, accuracy, and efficiency of the three proposed methods, not to analyze their advantages and limitations in comparison with other approximation methods reported in the literature, nor to rigorously study the scalability of our methods. All experiments were done on a Pentium Pro PC, with all algorithms implemented in C.

Two example BBNs of moderate size and complexity were used in the experiments so that we can compute the exact solution for each problem instance within reasonable time. Experiments with larger, more complex networks might be more indicative, but obtaining exact MAP solution for them may be computationally intractable. The first network, BBN1, is a two-layer network of 10 roots and 10 leaves, with 40 randomly generated arcs between them. Another network, which has the same structure as the Alarm belief network [7], was used as BBN2. This network has 37 nodes (11 of which are leaves), 46 arcs, and depth 10. The structures of these two BBN are given in Figures 1 and 2 below. The prior probabilities for all roots and causal strengths for all arcs in these two BBN were chosen randomly.

Figure 3 is about here

Figure 4 is about here

All three of our methods allow V, the set of instantiated variables, to be any subset of X. However, in the experiments, we chose V as the set of all leaf nodes of a network. This is because in many abductive inference applications such as diagnosis, the leaf nodes represent the evidence and are the only variables whose values can be observed (while the values of others must be inferred). Also, MAP problems with leaf nodes instantiated are in general computationally more expensive, as these nodes are at the bottom of the BBN. $U = X - V$ then comprised of all non-leaf nodes. Each of the $2^{|V|}$ possible instantiations of V was used as v^+ to define a MAP problem instance. Therefore, 1024 problem instances for BBN1 and 2048 for BBN2 were tested. The search space (consisting of all possible instantiations of $|U = X - V|$ for the two BBN are of the size of $2^{10} = 1024$, and $2^{26} \approx 6.7 \times 10^7$. For each problem instance, V was clamped to v^+, and the solution instantiation u was obtained from each of the three methods.

To determine whether the solutions from our approximation methods are globally optimal, we also implemented an exhaustive search algorithm for exact MAP solutions. The search is depth-first so the space requirement is minimum. A pruning technique is used to make the search more efficient. A complete path in the search represent a solution. A partial path can be pruned if the upper bound of posterior probability of all solutions that can be generated from it is smaller than that of the currently known best solution. Then, backtracked from this point, the search continues with another partial path.

Approximation methods usually give local optimal solutions which may or may not be globally optimal. However, good suboptimals are often acceptable in many practical applications. We thus need some criterion to judge whether a solution obtained from our proposed methods in the experiment is acceptable. There are many reasonable criteria, e.g., it can be based on absolute or relative error of $P(u|v^+)$ with respect to $P(u^*|v^+)$, or based on the number or percentage of variables that differ in u and u^*. In the experiments we adopted a simple heuristic criterion: u is acceptable if it is among the first few best instantiations of U, given $V = v^+$, and somewhat arbitrarily picked up 3 as the limit. That is, we consider any solution that is one of the best three an acceptable approximation, then any other solution represents a failure. Our exact solution algorithm stores up to 3 best solutions found so far during the search, and when it stops, the three globally best solutions are returned. There is a slight danger that, with the pruning, the exact algorithm may not return three best solutions if the globally optimal one is found early in the search. However, we did not encounter such situation in the experiments.

To compare the time complexity of these methods, we recorded the average CPU time for solving a MAP instance for each of the three approximation methods for the two example networks. As a baseline, we also recorded the average time for our exact method, which is 28 ms for the small BBN1, and 3.1 minutes for BBN2. The substantially much higher computing time for BBN2 reflects its much larger search space.

Like many neural network methods, the performance of our methods may be affected by the network initialization and the selections of the parameters such as the initial temperature $T(0)$ and cooling factor δ. After some trial-and-error attempts, these parameters were chosen as follows for our experiments. For the discrete method and SA method, the networks were initialized with $x_i = 1 \; \forall i$. For the continuous method, $\hat{x}_i$ were initialized to small random numbers. For the SA method, we chose $T(0) = 1.0$ for BBN1 and $T(0) = 10.0$ for BBN2. For both BBN1 and BBN2, $\delta = 0.99$, and at each temperature the activations of 100 randomly selected nodes were updated, approximating the arrival at equilibrium at that temperature. For the continuous method, $T(0) = 3.0$ and $\delta = 0.95$ were used for BBN1, and $T(0) = 10.0$ and $\delta = 0.9$ for BBN2. We also tested the resettling process with the continuous method. The experiment results for the two BBN, together with their average computing time for a problem instance, were given in Tables 1 and 2, respectively.

Table 1 is about here

Table 2 is about here

It is clear from the experiments that the discrete method was the fastest, but it performed poorly in terms of the accuracy of the solutions it produced. It failed in more than 50% cases for the relatively simple BBN1, and 82.1% cases for the more complex BBN2. Data for SA method and continuous method show these two methods have comparable failure rates, which were substantially better than that of the discrete method. However, the computing time for SA method was very high even for these two example networks of moderate size. It is about 30 and 50 times greater than the continuous method for BBN1 and BBN2, respectively. Also, we saw that the failure rates of both SA and continuous methods were significantly higher for the larger and more complex BBN2 than for BBN1. On the other hand, comparing with the exact method, these two approximation methods achieved much higher saving on computing time for BBN2 than BBN1.

Preliminary experiments showed that resettling had only marginal effect on solution quality when applied to the discrete method (it caused a drop of only a few percentage points in the failure rate for the two BBN). It would be too expensive to apply resettling to the SA method (about 35 hours for BBN2). We thus chose to apply resettling only to the continuous method in the experiments, although it is in principle not restricted to a particular model. The experiment results suggest that the resettling process was well worth its cost as it improved the solution accuracy significantly. The result is remarkable for BBN1. It produced the globally optimal solutions for all but one of the 1024 problem instances. The results for BBN2 is also significant, as the failure rate reduced from as high as 41.3% to 7.3%. The computing time, although increased by about 26 times as there are 26 variables in BBN2, was still less than half of the time the SA method took without resettling, and significantly lower than that of the exact method.

8 Discussions

In this paper we took a neural network approach to solving the difficult probabilistic inference problem of MAP in BBN. The keys to convert a BBN to a neural network are 1) turning the BBN to a dynamic system by treating each node as a processing unit capable of determining its activation based on the activations of other units, and 2) defining the system energy function (Eq. 5) in such a way that the minimum energy state corresponds to the most probable solution. The definition of $r_i(t)$ (Eqs. 7 and 9) helps to localize the influence of variables to each other so that the activation of a node is only affected by its parents and children. This formulation allows a straightforward adoption of two well-known neural network optimization techniques to MAP problems, namely the discrete Hopfield model which leads to our hill-climbing style discrete method, and the simulated annealing method. The discrete method is very fast but suffers from poor solution quality. The SA method, due to its stochastic nature, is computationally very expensive, and thus not suitable for many practical problems represented by large and densely connected BBN. The most significant contribution of this work is the development of our continuous method that incorporates the annealing process into a mean field theory formulation proposed by Haft *et al* [12]. By combining the mean field theory and simulated annealing, this method is both fast and fairly accurate. As a mean field approximation, it replaces the stochastic variables with their mean values, thus avoiding the time-consuming process of reaching thermal equilibrium. At the same time, as with simulated annealing, since $P(X|T(t))$ obeys Boltzmann-Gibbs distribution, the system will be more likely to settle down to a global optimal solution at very low temperature if $Q(X|T(t))$ approximates $P(X|T(t))$ well.

The key results (Eq. 9 used by both the discrete and SA methods and Eq. 25, the mean field equation with annealing for the continuous method) are applicable to general BBN of binary variables. For Noisy-or networks, an important subclass of BBN, MAP can be solved more efficiently using Eq. 11, and Eqs. 33 and 34, respectively. With a tolerable additional cost (by a factor of n), the solution quality of the continuous method can be further improved, sometimes significantly, by the resettling process.

The continuous method may also be applied to approximating belief update and MPE problems with minor changes. Compute $P(x_i|v^+)$ in belief update can be achieved by allowing the temperature T to gradually reduce to 1 (by a cooling schedule, say, $T(t+1) - 1 = (T(t) - 1) * \delta$) because when $T \rightarrow 1$, $Q(U|v^+)$ approximates $P(U|v^+)$. Here, as with MAP problems, the mean field treatment of variable update may lead to faster convergence than SA approach. A MPE (most probable explanation) problem can be expressed as finding the most probable joint assignment e^* to the (fixed) set of explanatory or hypothesis variables $E \subseteq X - V$, given $V = v^+$. That is, find e^* such that $P(e^*|v^+) = \max_e P(e|v^+)$. MPE are known to be more difficult than MAP when exact solutions are sought [25]. This is because, among other things, the free variables in $H = X - V - E$ make computing even the simple probability $P(E = e|v^+)$ for a particular e an intractable task when H is large, since $P(e|v^+) = \sum_h P(e, h|v^+)$ potentially involves enumerating all possible instantiations of H. However, we may be able to approximate MPE by our continuous method with two separate cooling schedules, one with $T_E \rightarrow 0$ for variables in E, the other with $T_H \rightarrow 1$ for variables in H. It would be conceivable that with schedule T_E, variables in E gradually move toward a joint assignment in much the same way as U moves to a solution u in MAP. At the same time, similar to belief update, $P(H|v^+)$ will be approximated by $Q(H|v^+)$ with schedule T_H. More work is needed before we can claim that the assignment e obtained from this schema is indeed a good approximation of e^*, because, among other thing, the interaction between the two sets of variables (E and H) may be more complicated than it appears to be, and it is not clear whether the system will converge.

In the three of our methods, the information received by X_i from its parents and children when computing r_i (see Eq. 7) bears some resemblance to π (parent to child) and λ (child to parent) in Pearl's belief propagation algorithms [27]. Like π messages, influences from parents are weighted in r_i by the conditional probability table $P(x_i|\pi_i)$, like λ messages, influences from children are multiplied together. However, there are several significant differences between Pearl's and our methods. First, in Pearl's methods, to complete the belief update or revision, the influence of an instantiated node (evidence) needs to traverse the network only once, while it takes many iterations in our methods until it reaches an equilibrium. Secondly, the way these influences are synthesized is different. For example, in Pearl's algorithms, π messages are synthesized by $\sum_{\pi_i} P(x_i|\pi_i) \prod_{X_j \in \pi_i} \pi_{msg}$ (sum-product for belief update) and $\max_{\pi_i} P(x_i|\pi_i) \prod_{X_j \in \pi_i} \pi_{msg}$ (max-product for belief revision), while in our methods we use the ratio $P(x_i = 1|\pi_i)/(1 - P(x_i = 1|\pi_i))$. Similar difference exists for synthesizing influences from children. This is because Pearl's update rules are derived by breaking the influence of all evidences to a variable into influence from each of its parent and child (this can be done in an exact form for singly-connected networks) while our update rules are aimed at energy minimization. Finally, if networks is singly-connected, Pearl's propagation methods guarantee to converge to the exact solutions, but convergence is not guaranteed for multi-connected networks, although they converge for many such networks with good solutions [36]. In contrast, the three of our methods are guaranteed to converge. Although our continuous method only guarantees convergence when no tie exists, ties can only make the system oscillating between states of equal energy, thus they do not affect the solution quality and can be easily broken. It would be interesting to see if some of our methods would guarantee exact solutions for singly connected networks.

In this paper we did not attempt to thoroughly compare our methods with other approximation methods for MAP problems (e.g., the genetic algorithm approach [2, 5], high order recurrent neural network approach [3], and loopy belief revision [21, 36]), and we make no claim that our methods would necessarily perform better than the others. Our purpose is to present an alternative approach to the research community with its unique features. However, we have noticed that the genetic algorithm approach is more general in that it applies to general BBN with *multi-valued variables*, and it has the advantage of incremental improvement of solution accuracy. On the other hand, this approach may require more computing resources (both time and memory) than our continuous method. The high order recurrent neural networks can be reduced to Hopfield networks [3], that approach thus may suffer from poor solution quality as our discrete method does. As for loopy belief revision method, it has been noted by others [36] that if the execution of the algorithm converges, it often gives a good quality solution, but the algorithm converges less often than for belief propagation.

We plan to conduct a systematic empirical study to compare the performance of our methods with other proposed approximation methods, especially the Gibbs sampling, the genetic algorithm and loopy belief revision. In the study, we plan to use many more BBNs with different size, complexity, and probability distributions. Through this empirical study we hope to obtain an objective comparison between our methods and the others in their performance (both solution accuracy and computation time). The study may also shed lights to what types of BBN each of these methods are particularly suitable for.

We also plan to investigate the scalability of the proposed neural network methods, both theoretically and empirically. Scalability involves two aspects of a method. The first aspect concerns the computational complexity of the method. If it is really sub-exponential, then we would expect the methods to be practically tractable for large networks. It is conceivable, and the experiment, though limited in the scope, seemed to suggest that the time complexities of our three methods are indeed sub-exponential[f]. However, more evidence, either analytical or empirical, is needed in order to establish this proposition on a form basis. The second aspect of scalability concerns the solution quality (accuracy) a method provides. If the solution accuracy deteriorates quickly as the problem size increases, the method would not be considered scalable. The limited experiment data indicates that the solution quality for BBN2 was significantly worse than that for BBN1. More evidence needs to be collected before a clear trend can be identified. A related problem that needs to be addressed more thoroughly is to what extent the time complexity and the solution quality are affected by the network structure (e.g., the depth of the network and the average in-degree of non-root nodes in the network) and the probability distribution.

Further work is also needed to extend this work to BBN of multi-valued variables. This would be relatively straightforward for our discrete method. Since the energy difference between two system states that differ only on the instantiations of X_i, say to a and b, is determined by the ratio of $P(X_i = a|M_i)/P(X_i = b|M_i)$, X_i can be updated to the state with the largest $P(x_i|M_i)$ among all its instantiations. However, extension of SA method and continuous method, which also involves annealing, would be more complicated. This is because ΔE, which is used in annealing equation 13, can be directly expressed by r_i for binary variables but not so for multi-valued variables. The extension may require a new problem formulation that, among other things, replaces $r_i(t)$ by some other measure which indicates preference among possible values of X_i and can be determined locally.

Acknowledgement This work was supported in part by U.S. NSF Grant IRI-9309136.

Footnotes

a. The contact author.

b. We have taken this approach in our earlier work of diagnostic problem-solving [28]. That work was limited in that 1) it was only applicable to two-layer belief networks; and 2) the method developed there was a heuristic solution without rigorous theoretical justification. It turns out that the node update rule derived in that work is a special case of our discrete method when the networks are restricted to two-layer belief networks, and all leaf nodes are instantiated.

c. Others [27] attach the quantity "$1-c_{ji}$" to arc $< X_i, X_j >$, representing the probability that $X_i = 1$ is ineffective in forcing X_j to be 1.

d. Others (e.g., [15]) use $P(x_i|\pi_i) = (1 - \prod_{X_k \in \pi_i}(1 - c_{ki}x_k))^{x_i} \cdot (\prod_{X_k \in \pi_i}(1 - c_{ki}x_k))^{1-x_i}$, which is equivalent to Eq. 4 when $x_i \in \{0, 1\}$. We use Eq. 4 to facilitate representation of expectations of x_i which is in [0, 1], in out continuous method.

e. It has been established [21] that each BBN can be transformed into a undirected graph known as Markov Random Field (MRF) through a unwrapping process. However, the transformation enlarges the size of the graph substantially [33].

f. The ratio of $|U|$ for the two BBN is 1:2.6,while the ratios of computing times of the three methods for the two BBN (without resettling) are all less than 1:2.5 in the experiments. In contrast, the ratio for our exact method, which is known to be exponential, is 1:6642 (28ms:3.1minutes).

References

[1] A.M. Abdelbar and S.M. Hedetniemi, "A Parallel Hybrid Genetic Algorithm Approach to Finding Probable Explanations on Bayesian Belief Networks", in *Proceedings IEEE International Conference on Neural Networks*, **1**, 450-455, 1997.

[2] A.M. Abdelbar and S.M. Hedetniemi, "Approximating MAPs for Belief Networks is NP-hard and Other Theorems", *Artificial Intelligence*, **102**, 21-38, 1998.

[3] A.M. Abdelbar and M. Assaggaf, "Recurrent High Order Networks for Probabilistic Explanation", *Proceedings International Joint Conference on Neural Networks (IJCNN)* Washington, DC, July, 1999.

[4] D.J. Amit, *Modeling Brain Function: The World of Attractor Neural Networks*, New York: Cambridge University Press, 1989.

[5] L.M. de Campos, J.A. Games, and S. Moral, "Partial Abduction in Bayesian Belief Networks Using a Genetic Algorithm", *Pattern Recognition Letters*, **20**, 1211-1217, 1999.

[6] G. Cooper, "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Network", *Artificial Intelligence*, **42**, 393-347, 1990.

[7] G. Cooper and E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from Data", *Machine Learning*, **9**, 309-347, 1992.

[8] T. Cover and J. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[9] R. Dechter, "Bucket Elimination: A Unifying Framework for Probabilistic Inference", in *Learning in Graphical Models*, M. Jordan (Ed.), Boston: Kluwer Academic Publishers, 1998, 75-104.

[10] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of a Images", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, **PAMI-6**, 721-741, 1984.

[11] A. Goel, J. Ramanujam, and P. Sadayappan, "Toward a Neural Architecture for Abductive Reasoning", *Proceedings of IEEE International Conference on Neural Networks*, **1**, 681-688, 1988.

[12] M. Haft, R. Hofmann, and V. Tresp, "Mean Field Inference in a General Probabilistic Setting", *Proceedings of The 7th International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Jan. 1999.

[13] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities ", *Proceedings of the National Science Academy, USA*, **79**, 2554-2558, 1982.

[14] J. J. Hopfield, "Neurons with Graded Responses Have Collective Computational Properties Like Those of Two-States Neurons", *Proceedings of the National Science Academy, USA*, **81**, 3088-3092, 1984.

[15] T.S. Jaakkola and M.I. Jordan, "Improving the Mean Field Approximation via the Use of Mixture Distributions", in *Learning in Graphical Models*, M. Jordan (Ed.), Boston: Kluwer Academic Publishers, 1998, 163-173.

[16] T.S. Jaakkola and M.I. Jordan, "Variational Probabilistic Inference and the QMR-DT Database", *Journal of Artificial Intelligence Research*, **10**, 1-15, 1999.

[17] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems", *Science*, **220**, 671-680, 1983.

[18] S.L. Lauritzen and D.J. Spiegelhalter, "Local Computation with Probabilities in Graphic Structures and Their Applications in Expert Systems", *Journal of the Royal Statistical Society, Series B*, **50**(2), 157-224, 1988.

[19] S.L. Lauritzen and F. Jensen, "Local Computation with Valuations from a Commutative Semigroup", *Annals of Mathematics and Artificial Intelligence*, **21**, 51-69, 1997.

[20] R.L. McEliece, D.J.C. MacKay, and J.F. Cheng, "Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm", *IEEE Journal on Selected Areas in Communication*, **16**(2), 140-152, 1998.

[21] K.P. Murphy, Y. Weiss, and M.I. Jordan, "Loopy Belief-Propagation for Approximate Inference: An Empirical Study", *Proceedings of Uncertainty in Artificial Intelligence*, 1999.

[22] R Neal, "Connectionist Learning of Belief Networks", *Artificial Intelligence*, **56**, 71-113, 1992.

[23] R. Neaplitan, *Probabilistic Reasoning in Expert Systems, Theory and Algorithms*, New York:John Wiley & Son, 1990.

[24] G. Parisi, *Statistical Field Theory*, MA: Addison-Wiley, Reading, 1988.

[25] J. Park and A. Darwiche, "Approximating MAP using Local Search", in *Proceedings of Uncertainty in Artificial Intelligence*, 2001.

[26] J. Pearl, "Evidential Reasoning Using Stochastic Simulation of Causal Models", *Artificial Intelligence* **32**, 245-257, 1987.

[27] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan Kauffman Publishers, 1988.

[28] Y. Peng and J. Reggia, "A Connectionist Model For Diagnostic Problem Solving", *IEEE Trans. SMC* **19** (2), 285-298, 1989.

[29] Y. Peng and J. Reggia, *Abductive Inference Models for Diagnostic Problem-Solving*, New York: Springer-Verlag, 1990.

[30] Y. Peng and Z. Zhou: "A Neural Network Learning Method for Belief Networks", *International Journal of Intelligent Systems*, **11**, 893-916, 1996.

[31] C. Peterson and J.R. Anderson, "A Mean Field Theory Learning Algorithm for Neural Networks", *Complex Systems*, **1**, 995-1019, 1987.

[32] C. Rojas-Guzman and M.A. Kramer, "GALGO: A Genetic ALGOrithm Decision Support Tool for Complex Uncertain System Modeled with Bayesian Belief Networks", in *Proceedings of Uncertainty in Artificial Intelligence*, 368-375, 1993.

[33] K.L. Saul, T. Jaakkola, and M.I. Jordan, "Mean Field Theory for Sigmoid Belief Networks", *Journal of Artificial Intelligence Research*, **4**, 61-76, 1996.

[34] S.E. Shimony, "Finding MAPs for belief networks is NP-hard", *Artificial Intelligence*, **68**, 399-410, 1994.

[35] S.E. Shimony and E. Charniack, "A New Algorithm for Finding MAP Assignments to Belief Networks", in *Uncertainty in Artificial Intelligence*, **6**, M. Henrion *et al* (eds.), North-Holland, 1991, 185-193.

[36] Y. Weiss, "Correctness of Local Probability Propagation in Graphical Models with Loops". *Neural Computation*, **12**, 1-41, 2000.

[37] Y. Weiss and W.T. Freeman, "On the Optimality of Solutions of the Max-Product Belief Propagation Algorithm in Arbitrary Graphs". *IEEE Transactions on Information Theory* , **47**2, 723-735, 2001.

[38] Y. Weiss, "Comparing the Mean Field Method and Belief Propagation for Approximate Inference in MRF", to appear in Saad and Opper (eds.) *Advanced Mean Field Method*, MIT Press, 2002.

[39] N. Wiberg, *Code and Decoding on General Graphs*, Ph.D. Thesis, Department of Electrical Engineering, University of Linkoping, Sweden, 1996.

Captions for Figures

Figure 1 An example MAP problem.

Figure 2 Information flows to a node from its parents and children.

Figure 3 BBN1: a 10 $\times$ 10 bipartite belief network.

Figure 4 BBN2: the Alarm belief network structure of 37 nodes and 46 links.

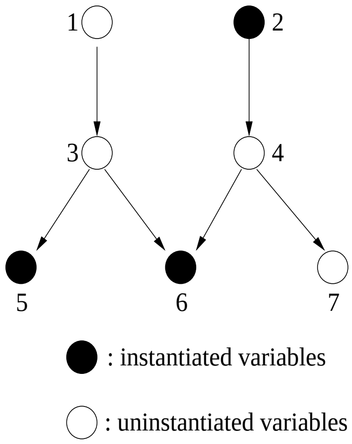

Figure 1: (Peng and Jin)

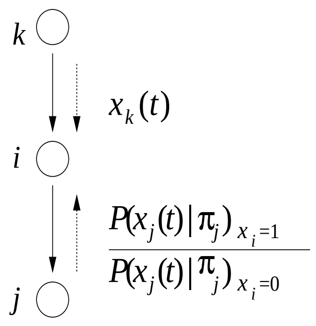

Figure 2: (Peng and Jin)

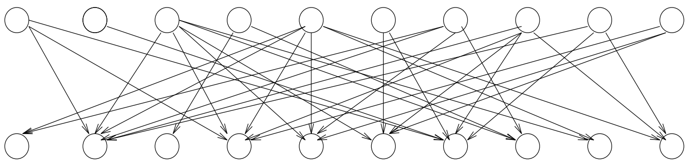

Figure 3: (Peng and Jin)

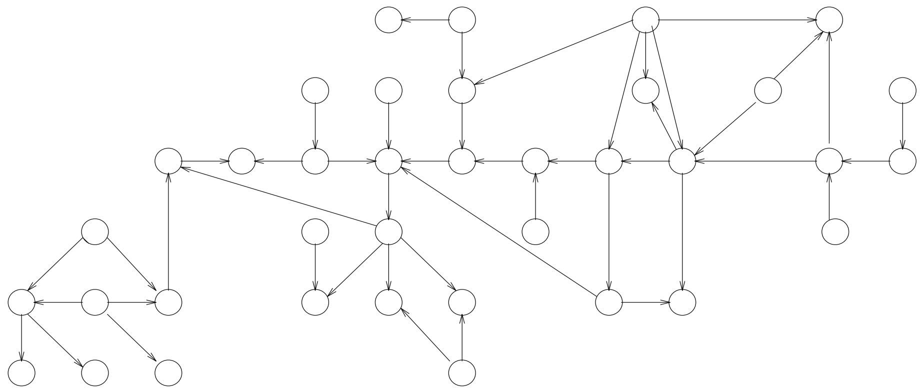

Figure 4: (Peng and Jin)

Table 1: Results for BBN1.

	1st	2nd	3rd	others	failure rate	time
Discrete method	370	58	39	557	54.4%	6ms
SA method	780	61	19	164	16%	1014ms
Continuous method without resettling	888	56	7	73	7.1%	31ms
Continuous method with resettling	1023	0	0	1	0.098%	312ms
exact method	1024					28ms

Table 2: Results for BBN2

	1st	2nd	3rd	others	failure rate	time
Discrete method	256	51	49	1682	82.1%	7ms
SA method	1101	91	94	762	37.2%	2511ms
Continuous method without resettling	1155	39	8	846	41.3%	49ms
Continuous method with resettling	1665	42	191	150	7.3%	1289ms
exact method	2048					3.1minutes