Knowledge-Based Systems 239 (2022) 107996

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

An efficient approach for multiple probabilistic inferences with Deepwalk based Bayesian network embedding

Jiahui Wang [a], Kun Yue [a], Liang Duan [a,*], Zhiwei Qi [a], Shaojie Qiao [b]

[a] School of Information Science and Engineering, Yunnan University, Kunming, China
[b] School of Software Engineering, Chengdu University of Information Technology, Chengdu, China

ARTICLE INFO

Article history:
Received 6 February 2021
Received in revised form 16 December 2021
Accepted 17 December 2021
Available online 24 December 2021

Keywords:
Bayesian network
Probabilistic inference
Network embedding
Random walk

ABSTRACT

As a classical probabilistic graphic model, Bayesian network (BN) is widely used for representing and inferring dependence relationships with uncertainties. However, multiple probabilistic inferences on BN are quite inefficient, since each probabilistic inference on BN is extremely time-consuming and meanwhile the intermediate results cannot be reused. It is necessary to improve the overall efficiency of multiple probabilistic inferences on the same BN by incorporating an easy-to-calculate inference method and an easy-to-reuse technique for common calculations in multiple inference tasks. In this paper, we first propose a Deepwalk based method for BN embedding, as a specification of general graph embedding, which preserves both the graphical structure and conditional probabilities in BN. We then provide the algorithm to approximate probabilistic inferences via the distance among embeddings. We finally present an efficient approach to multiple probabilistic inferences. Extensive experiments illustrate that our proposed method is effective for BN embedding, and outperforms other state-of-the-art competitors by improving the inference efficiency with several orders of magnitude.

1. Introduction

As an important probabilistic graphical model (PGM), Bayesian network (BN) consists of a directed acyclic graph (DAG) of random variables as nodes and conditional probability tables (CPTs) to quantify the dependence relations between the nodes [1]. BN has been widely adopted as the framework of uncertainty representation and inference in medical diagnosis [2], fault diagnosis [3], financial analysis [4] and bioinformatics [5], etc. Fig. 1 shows a CANCER BN, on which probabilistic inferences could be made by calculating the probabilities of targets w.r.t. the given evidences, such as $P(Smoker = \text{true}|Dyspoea = \text{true})$ and $P(Smoker = \text{true}|Xray = \text{positive})$. For real-world applications, a doctor in a specific department could diagnose different patients according to their different symptoms by conducting multiple probabilistic inferences. For example, an oncologist could judge whether different patients have cancer or not respectively based on the CANCER BN and their relevant indicators.

However, the multiple probabilistic inferences on BN are usually inefficient, since both the exact and approximate probabilistic inference algorithms are #P-hard [6,7], such as Variable Elimination (VE) [8] and Gibbs Sampling (GS) [9], taking $O(2^n)$ time for a single inference task on a BN containing n variables. Thus, t times inferences will cost $O(t * 2^n)$ time when the inferences are executed one by one. This means that the more the probabilistic inferences on the same BN and the more complex the BN, the less the overall efficiency will be. Note that some common probabilities might be repeatedly calculated in multiple inferences with various targets and evidences (i.e., evidence variables with corresponding values). For example, to infer $P(Smoker = \text{true}|Dyspoea = \text{true})$ and $P(Smoker = \text{true}|Xray = \text{positive})$, the probabilities $P(Smoker = \text{true}|Cancer = \text{true})$ and $P(Smoker = \text{true}|Cancer = \text{false})$ w.r.t. *Cancer* are repeatedly calculated. Therefore, it is worthwhile to improve the overall efficiency of multiple probabilistic inferences on the same BN by incorporating with an easy-to-calculate inference algorithm and an easy-to-reuse technique for common calculations in multiple inferences. These are the problems to address in this paper.

Recently, graph embedding (GE) [10–12] has achieved widespread success in numerous graph-based applications by learning powerful representations of the nodes, edges, the whole or the parts of a graph with low dimensional vectors to preserve the structure and affiliated information [13–15]. GE not only provides powerful representations, but also simplifies the calculations on the graph [16,17]. Revisiting the above classical "search-based" probabilistic inferences of BN, the repeated search of CPTs is doomed, which will be much more expensive for the

* Corresponding author.
E-mail addresses: wjh@ynu.edu.cn (J. Wang), kyue@ynu.edu.cn (K. Yue), duanl@ynu.edu.cn (L. Duan), maryqizhiwei@ynu.edu.cn (Z. Qi), sjqiao@cuit.edu.cn (S. Qiao).

https://doi.org/10.1016/j.knosys.2021.107996
0950-7051/

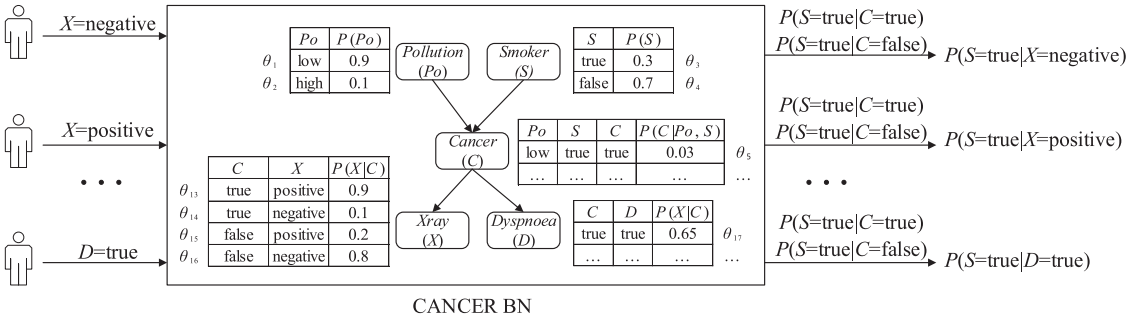


Fig. 1. BN Based Medical Diagnosis.

task with multiple inferences. To this end, we consider establishing an efficient framework for multiple inferences without repeated search of CPTs by incorporating the idea of GE.

It is known that conditional probabilities in the CPT of each variable represent the probabilities of the states of this variable under those of its parents. Naturally, we could construct a weighted directed graph (WDG) to facilitate the inferences by representing all the states corresponding to the conditional probabilities of variables in BN. Specifically, each conditional probability parameter is regarded as a node in WDG. An edge in WDG describes the dependence relationship from the conditional probability parameter of the parent node to that of the child node. The conditional probability of the child node given its parents could be adopted as the weight on the edge, which we regard as the transition probability between dependent parameters. Actually, a subgraph in WDG is a state of a BN given a specific assignment of all variables, and thus joint probabilities (JPs) could be calculated by the product of weights on the edges in the subgraph. This means that probabilistic inferences could be fulfilled by searching the WDG only once. However, the brute-force of subgraph search makes the calculation of JPs quite inefficient even for one-time probabilistic inference, although the repeated search of CPTs could be avoided. It is expected that the calculation of JPs could be fulfilled by only knowing the weights concerned in the subgraph, which makes sense in random walk [18].

It is conceivable to adopt the idea of random walk into GE, so as to generate low dimensional embedding of BN based on WDG, and then make probabilistic inferences efficiently with the embedding. However, traditional GE methods [18–21] do not work due to the inability of preserving CPTs that express both the graphical structure and quantitative dependencies, as the most important affiliated attributes of the DAG [1,6,7]. Fortunately, by using random walk for node sampling, Deepwalk [18] is a classical GE method based on Skip-Gram [22] for unweighted and undirected graphs without affiliated information. By means of Deepwalk, the distance between two nodes in WDG could be evaluated to describe their JP. Thus, in this paper, we explore Deepwalk based BN embedding (BNE) that transforms BN into a low-dimensional vector space for easy-to-calculate and easy-to-reuse probabilistic inferences.

Note that the sampling of random walks, preserving the first order proximity in Deepwalk, is analogous to the sampling of variables in the BN inference method like Gibbs sampling [9], where random samples w.r.t. JPs are generated by "search-based" iterations, and then the probabilities are approximated. Thus, for BN embedding, we extend Deepwalk by changing the strategy for generating random walks based on the states with transition probabilities in WDG. First, we generate probability-biased random walks from *START* to simulate the sampling of variables with different values, where *START* represents the initial state that all variables are not assigned to any values. Following, we use Skip-Gram with the generated probability-biased random walks to achieve BNE so that the nodes with larger transition probabilities still keep close after embedding.

Then, we discuss the efficient method for probabilistic inferences with BNE. Based on the Euclidean distance, we first define the marginal probability P_m based on the distance from *START* to a certain state of a variable and the total distance from *START* to all the states of this variable. We then provide a method to approximate the JP of the states of *START* and other concerned variables by a probability P_j. The larger the probability, the larger the P_j will be, since the probability-biased random walks preserve the probabilistic proximity of BN. Following, we calculate the probabilities according to Bayes formula with the obtained JPs.

It is worth noting that the variables' states and their transition probabilities are preserved in the WDG that has been constructed once without repeated search of CPTs. Moreover, the same embedding of a BN is reused for common calculations in multiple probabilistic inferences, which are efficiently fulfilled by easy-to-calculate approximation of the distance with BNE.

Generally, our contributions are summarized as follows:

- We propose a novel method to construct WDG for representing the states of variables in BN. By describing the dependence relationships between pairs of conditional probability parameters, all states with transition probabilities implied in CPTs are preserved for probabilistic inferences.
- We propose the Deepwalk based method of BNE with probability-biased random walks to transform the nodes of WDG into a low-dimensional vector space, which reduces the expensive evaluations of probabilistic inferences into efficient calculations on vectors.
- We provide a method PIBNE to approximate the probabilities of targets and evidences based on BNE. Then, we extend this method to implement a search-and-calculate based approximate algorithm for multiple probabilistic inferences.
- We conduct experiments on five different sized BNs to verify the efficiency and effectiveness of our proposed method. Experimental results show that our method outperforms other state-of-the-art competitors by improving the efficiency with several orders of magnitude.

The rest of this paper is organized as follows. Section 2 introduces preliminaries and related work. Section 3 gives the method of BNE based on probability-biased random walks. Section 4 describes the approximate algorithm for probabilistic inferences with BNE. Section 5 gives experimental results and performance studies. Section 6 concludes and discusses future work.

Table 1
Notations and descriptions.

Notations	Descriptions
$\mathcal{B} = (G, \Theta)$	BN with DAG G and conditional probabilities Θ
$G = (\mathcal{V}, \mathcal{E})$	Graph with the nodes (a.k.a. variables in BN) $\mathcal{V}$ and edges $\mathcal{E}$
$\Theta = \{\theta \mid \theta = P(x \mid Pa(X))\}$	Set of conditional probabilities
$Pa(X)$	Parents of variable X
d	Dimensionality of embedding
$f : \theta_i \longrightarrow \boldsymbol{y_i} \in \mathbb{R}^d, 1 \leqslant i \leqslant n_c$	Mapping function in BN embedding
$\boldsymbol{Y}$	Embedding result of BN
$P(\tau \mid \varsigma)$	Probabilistic function of τ given ς
t	Times of multiple probabilistic inferences
n	Number of variables in BN
W_{v_i}	Random walks rooted at node v_i

2. Preliminaries and related work

In this section, we first present some background knowledge and notations, and then introduce related work.

2.1. Preliminaries

Let $\mathcal{X}$ denote a set of random variables $\{X_1, \ldots, X_n\}$. A BN for $\mathcal{X}$ is a pair $\mathcal{B} = (G, \Theta)$, where G is a directed acyclic graph (DAG) of $\mathcal{X}$ and Θ is a set of conditional probabilities in conditional probability tables (CPTs). The network structure $G = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ corresponding to the variables $\mathcal{X}$ and a set of edges $\mathcal{E}$ to represent the dependencies between variables in $\mathcal{V}$. A directed edge exists from X to Y in G means that X is a *parent* of Y and Y is a *child* of X, and $Pa(X)$ is used to denote the parents of X. The probabilistic dependency between X and its parents is represented by $\theta (\theta \in \Theta)$ that denotes $P(x|Pa(X))$. We use capital letters, small letters, calligraphy letters (e.g., $\mathcal{X}$ and $\mathcal{V}$), bold letters to denote variables, values of variables, set of variables, vectors, respectively.

BN embedding is to represent BN with low dimensional vectors while the DAG structure and CPTs in BN are preserved. Given a BN $\mathcal{B} = (G, \Theta)$ and a dimensionality d ($d \ll |\Theta|$), BN embedding is a mapping $f : \theta_i \longrightarrow \boldsymbol{y_i} \in \mathbb{R}^d, 1 \leqslant i \leqslant |\Theta|$, also abbreviated as $\Theta \longrightarrow \boldsymbol{Y}$, in which G and Θ are preserved as much as possible. With this mapping, all conditional probabilities in Θ are represented with d dimensional vectors.

Probabilistic inference of BN is to calculate the probability $P(\tau|\varsigma)$, where τ and ς, for simplification of expression, denotes the given targets (target variables with corresponding values) and evidences (evidence variables with corresponding values) respectively. *Multiple probabilistic inferences* of BN aim to calculate t ($t > 1$) times different $P(\tau|\varsigma)$ with different targets τ and evidences ς on the same BN.

We summarize the key notations and their descriptions in Table 1.

2.2. Related work

Probabilistic inference of BN. For single connected networks, the message passing algorithm [23] propagates probability to neighbor nodes, and the computation cost is proportional to the path length during evidence propagation. The junction tree algorithms [24,25] convert BN into a junction tree and then perform calculations by message passing, while finding the maximum clique is NP-hard. The methods for exact probabilistic inferences are not suitable for instant situations on large complex BNs due to the complicated DAG structure and massive conditional probabilities in CPTs. Random sampling algorithms, such as importance sampling [26] and Gibbs sampling [9], approximate probabilities from the samples generated by probabilistic distributions, and the degree of effectiveness of these algorithms is proportional to the size of samples and the efficiency is inversely proportional to the size of samples. These approximate algorithms converge slowly w.r.t. small probabilities of evidence.

Vasimuddin et al. [27] proposed a scalable distributed-memory parallel algorithm for exact inferences. Dal et al. [28] proposed a parallel algorithm for probabilistic inferences with weighted model counting. Zheng et al. [29] investigated a machine learning approach to accelerate the parallel junction tree algorithms. Nishihara et al. [30] presented a parallelizable Markov chain Monte Carlo algorithm for efficient sampling from continuous probability distributions. Gonzalez et al. [31] adopted graph coloring to construct a direct parallelization of the classical sequential scan Gibbs sampler. For multiple probabilistic inferences on a large complex BN, these parallel algorithms could be improved by using global parallel strategies and reusing previous intermediate results for different inferences.

Graph Embedding. The GE methods based on matrix factorization [20], random walk [18,32], deep learning [19] and other models [21,33] make graphs represented and evaluated efficiently. Intra [34] was proposed by Markov Random Field (MRF) of factor graphs with the embedding of mentions and entities to disambiguate named entities. By these two kinds of methods, PGM could be embedded into the low-dimension vector space with efficient representations.

GE with random walks is widely used to approximate the properties in both homogeneous and heterogeneous graphs, such as node centrality and similarity, and is quite useful when the graph is either observed partially or too large to be measured globally. A homogeneous graph is a graph in which there is only one type of node and edge, otherwise, it is a heterogeneous graph. Deepwalk [18] and Node2vec [32] are classical GE methods with random walks, which mainly focus on simple homogeneous graphs. Deepwalk generates a sequence of nodes with random length to save the local similarity between nodes and embeds graphs with Skip-Gram model. Node2vec extends Deepwalk by adding breadth-first search (BFS) and depth-first search (DFS) to save both the local and global similarity of nodes. The different types of nodes (i.e., variables) in BN and the affiliated information (i.e., CPTs) indicate that BN is a heterogeneous graph. The affiliated information of BN makes it hard to preserve all information by the aforementioned methods. Metapath2vec [35] generates random metapaths to make the embedding of heterogeneous graphs more efficient than general GE methods. GERI [36] takes a novel biased random walk to explore the constructed heterogeneous graph and uses modified heterogeneous Skip-Gram to learn graph embedding. Metapath2vec and GERI focus on heterogeneous graphs. Planetoid [37] samples node pairs according to the labels and structure of graphs. DeepCas [38] uses Markov chain based random walks to preserve information cascade sequences. Schltterer et al. [39] proposed a novel extension to random walk based graph embedding, which removes a percentage of the least frequent nodes from the walks at different levels to retain the relations to distant nodes. By the methods in [37–39], the nodes could be sampled with attributes of graphs. The attributes are

discrete and the states of nodes are determined. However, the CPTs of BN still cannot be preserved directly, since the values in CPTs are continuous and relevant to each other directly or indirectly.

3. Deepwalk based Bayesian network embedding

In this section, we first propose the method to preserve the DAG and CPTs of BN into a WDG. We then provide a Deepwalk based method to transform the WDG into a low dimensional vector space to fulfill BNE.

3.1. WDG construction

The polynomial of a BN is a multilinear function, in which each term is an instantiation of the BN, and the arithmetic circuit of network polynomial is designed for efficient probabilistic inferences. Inspired by the arithmetic circuit of network polynomial, we designed the WDG to preserve both structure and parameters of BN [40]. To preserve the DAG and CPTs, we generate a WDG $G' = (\mathcal{V}', \mathcal{E}', \mathcal{W}')$ from BN, where $\mathcal{V}'$, $\mathcal{E}'$ and $\mathcal{W}'$ is the set of nodes, edges and weights of edges, respectively. We first describe the construction of nodes in WDG, and then the generation of weighted edges.

Node construction. Each node $V \in \mathcal{V}'$ describes a conditional probability $\theta = P(x|Pa(X))$ in CPTs referring to the probability of '$X = x$ given $Pa(X) = \boldsymbol{pa}$' ($\boldsymbol{pa}$ denotes the m values $(pa_1, \ldots, pa_m)$ assigned to the parents of X), also denoted as $P(X = x|Pa(X) = \boldsymbol{pa})$). Note that the probability θ contains not only the *local structure* of BN, i.e., existing edges from $Pa(X)$ to X, but also the probabilistic dependency between X and its parents $Pa(X)$ when X is assigned to x and $Pa(X)$ are assigned to $\boldsymbol{pa}$. Thus, to represent each $\theta = P(x|Pa(X))$, we construct a node with three attributes: *parent*, *variable* and *probability*. The *parent* is a list of tuples, each of which consists of one of the X's parents and the assigned value, i.e., *parent* = $\{(Pa(X)_1, pa_1), \ldots, (Pa(X)_m, pa_m)\}$. The *variable* is a tuple (X, x), and the *probability* is the conditional probability of θ. For the nodes without parents, we use *START* as their parents.

Weighted edge generation. An edge $\langle V_i, V_j \rangle$ is generated from V_i to V_j if the *variable* of V_i is in the list of V_j's *parent* and the weight W_{ij} on this edge is assigned to V_j's *probability*, assume that V_i's *probability* is $X_1 = x_1$ and V_j's *probability* is $X_2 = x_2$, then the W_{ij} is $\theta = P(X_1 = x_1|X_2 = x_2, \ldots)$. Let X_i and x_i be the variable and assigned value in V_i's *variable* respectively. V_i only connects to the nodes constructed from the conditional probabilities of X_i's children in the DAG with the same assigned value as x_i. The procedure of WDG construction is shown in Algorithm 1.

The time complexity of Algorithm 1 is $O(|\Theta|ck)$, where c is the average number of children of each variable, and k is the average number of conditional probabilities in each variable's CPT. Note that WDG preserves both the structure and conditional probability parameters of the given BN, since the nodes and edges in WDG are the *variables* with their assigned values and dependence relationships derived from CPTs. The weights of the edges in WDG denote the transition probabilities of states represented by the nodes in WDG. Thus, probabilistic inferences could be conducted on the WDG without repeated search of CPTs. Example 1 shows a WDG constructed from BN.

Algorithm 1 WDG Construction

Input: BN: $\mathcal{B} = (G, \Theta)$
Output: $G' = (\mathcal{V}', \mathcal{E}', \mathcal{W}')$ // WDG of $\mathcal{B}$
Initialization: $\mathcal{V}' \leftarrow \varnothing, \mathcal{E}' \leftarrow \varnothing, \mathcal{W} \leftarrow \varnothing$
for each $\theta \in \Theta$ **do**
  Construct a node V from θ
  $\mathcal{V}' \leftarrow \mathcal{V}' \cup \{V\}$
end for
for each $V \in \mathcal{V}'$ **do**
  Set X as *variable* of V
  Set $\mathcal{V}_X$ as the children of X in G
  for each $V_i \in \mathcal{V}_X$ **do**
    Construct nodes $\mathcal{V}_N$ from the CPT of V_i
    for each $V_j \in \mathcal{V}_N$ **do**
      if V's *variable* $\in V_j$'s *parent* **then**
        $\mathcal{E}' \leftarrow \mathcal{E}' \cup \{< V, V_j >\}$
        $\mathcal{W}' \leftarrow \mathcal{W}' \cup \{V_j$'s *proability* $\}$
      end if
    end for
  end for
end for
return G'

Example 1. The nodes and weighted edges of WDG constructed from the CANCER BN in Fig. 1 are shown in Fig. 2(a) and Fig. 2(b) respectively. Note that a fully connected subgraph whose nodes consisted of one of the states of all variables in BN is a state of the BN and the probability of this state is the product of all the probabilities of nodes in the subgraph. For example, the red nodes and edges in Fig. 2(b) consists a state of CANCER BN that *Pollution* = *low*, *Smoker* = *true*, *Cancer* = *true*, *Xray* = *true*, *Dyspnoea* = *true*, and the probability of the state is the product of the probabilities of θ_1, θ_3, θ_5, θ_{13} and θ_{17}. Thus, all inference tasks can be conducted based on these subgraphs.

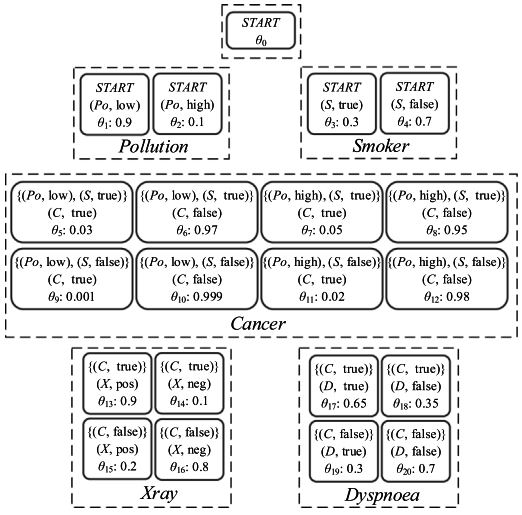
3.2. Embedding WDG with random walks

In order to fulfill BNE and probabilistic inferences with the embedding results, we are to make the probabilities of BN embedded in the vector space. In other words, the target of our proposed BNE is to make samples of different states in BN embedded. That is, the nodes with larger JPs will be closer than those with smaller JPs. With regard to the original BN, the states with larger transition probabilities are more likely to co-occurrence than states with lower transition probabilities. As for WDG, we are to make the close nodes in WDG have similar embeddings, which means to make $P(\theta_{i-w}, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_{i+w}|\theta_i)$ (w is the window size) maximized when given the node sampling sequences $\theta_{i-w}, \ldots, \theta_{i-1}, \theta_i, \theta_{i+1}, \ldots, \theta_{i+w}$ around the node θ_i. Then, we give the following objective function to fulfill and simplify BNE:

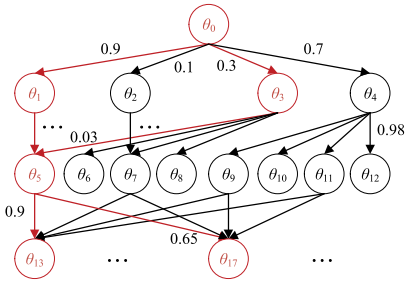
$$\arg\min_{\boldsymbol{Y}} -\log P(\theta_{i-w}, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_{i+w}|\boldsymbol{Y}(\theta_i)) \quad (1)$$

where Y is the vector to represent the result of BN embeddings. Eq. (1) guarantees that the nodes around θ_i (i.e., $\theta_{i-w}, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_{i+w}$) are close to the node θ_i in the embedding vector space and the distance among the embedding vectors will be inversely proportional to the co-occurrence probability of nodes, also denoted as the transition probability.

Specifically, the WDG preserves all the structure and parameters of BN and the probabilities of any inference tasks can be calculated exactly on the WDG. As mentioned in Example 1, the subgraphs of WDG are the samples of various assigned values to all the variables in BN and could be used to generate random walks to preserve the probabilistic proximity, which is the transition probability from V_i to V_j, representing the weight W_{ij} on the edge $\langle V_i, V_j \rangle$. The larger the weight W_{ij} between V_i and V_j, the larger the probabilistic proximity between V_i and V_j will be. Thus, we will generate the subgraphs of WDG to sample the states of BN and then embed these states into the vector space.



(a) Nodes



(b) Weighted edges

Fig. 2. WDG constructed from CANCER BN.

First, we generate the subgraphs of WDG and propose probability-biased random walks prior to implementing BNE, while preserving the probabilistic proximity. The subgraphs are extracted by a depth-first searching strategy that generates the probability-biased depth-first node walks from the *START* node. To denote the transition probability from the current node to its neighbor nodes effectively, we use the ratio of the weight of a single neighbor node to that of the total neighbor nodes. Thus, for the current node $V_i \in \mathcal{V}'$, the next node $V_j \in \mathcal{V}'$ is selected by the following probability biased sample distribution iteratively.

$$P(V_j|V_i) = \begin{cases} \dfrac{W_{ij}}{\sum_k W_{ik}} & ,\langle V_i, V_j\rangle \in \mathcal{E}', \langle V_i, V_k\rangle \in \mathcal{E}' \\ 0 & ,\langle V_i, V_j\rangle \notin \mathcal{E}' \end{cases} \tag{2}$$

In order to generate a reasonable state of BN on the probability biased depth-first node walks, we adopt a sequential search on the node walks and drop out some redundant nodes. Specifically, we will drop out the selected node if one of the states of the corresponding variable has been selected previously. The sampling strategy guarantees that the probability biased random walks constitute fully connected subgraphs, which are states of the BN and can be used for probability inferences. Then, we change the obtained subgraphs to node walks R_s by topologically sorting. The order independence of nodes in the subgraphs better captures a sense of 'nearness' by random walks [18]. According to Eq. (2), if the transition probability between V_i and V_j is larger than that between V_i and V_k, that is $P(\{V_i, V_j\} \in R_s) > P(\{V_i, V_k\} \in R_s)$, then there are probably more walks from V_i to V_j than those from V_i to V_k. Thus, the larger the transition probability of a node pair, the more probability biased walks containing this node pair will be generated. In other words, the probability-biased random walks preserve the probabilistic proximity in BN. The corresponding BNE and generated probability-biased random walks make nodes with larger transition probability have closer embedding and vice versa.

Then, we combine both the random walks and Skip-Gram model [22] to fulfill BNE, where Skip-Gram is a language model to maximize the co-occurrence probability among the words appearing within a window in a sentence. With the biased random walk on WDG, the subgraphs corresponding to different states of BN are generated. Following, we use Skip-Gram to represent WDG by low dimensional vectors w.r.t. the generated random walks, in which α is the learning rate and could be assigned to a small constant initially.

Algorithm 2 Skip-Gram

Input: Random walks: R_{V_i}
Window size: w
Output: Embedding result: $\boldsymbol{Y}$
for each $V_j \in R_{V_i}$ **do**
  for each $U_k \in R_{V_i}[j-w : j+w]$ **do**
    $J(\boldsymbol{Y}) \leftarrow -logP(U_k|\boldsymbol{Y}(V_j))$
    $\boldsymbol{Y} \leftarrow \boldsymbol{Y} - \alpha * \frac{\partial J}{\partial \boldsymbol{Y}}$
  end for
end for
return $\boldsymbol{Y}$

Given a BN $\mathcal{B} = (G, \Theta)$, we first generate the WDG $G' = (\mathcal{V}', \mathcal{E}', \mathcal{W}')$ by Algorithm 1. Then, we generate the probability-biased random walks on the WDG to simulate the states of all variables in BN with different assigned values. Taking as input the generated random walks, Algorithm 3 generates BNE by Skip-Gram.

The time complexity of Algorithm 2 is $O(\log|\Theta|)$, so the time complexity of embedding of WDG is $O(|\Theta|\log|\Theta|)$ and the total time complexity of Algorithm 3 is $O(|\Theta|\log|\Theta|)$.

Algorithm 3 BN Embedding (BNE)

Input: BN: $\mathcal{B} = (G, \Theta)$
Window size: w
Embedding dimensionality: d
Number of walks of each node: γ
Length of walks: l
Output: Embedding result: $\boldsymbol{Y} \in \mathbb{R}^{|\mathcal{V}'|*d}$
Generate G' by Algorithm 1
for $i = 1$ to γ **do**
  Generate R_0 by depth-first searching with Eq. (2)
  Generate G'_s by droping out redundant nodes and edges in R_0
  Generate R_s by topologically sorting on G'_s
end for
$\boldsymbol{Y} \leftarrow$Skip-Gram($\boldsymbol{Y}, R_s, w$) // by Algorithm 2
return $\boldsymbol{Y}$

Example 2. The CANCER BN could be embedded into 2 dimensional vectors, shown as Fig. 3, in which each node represents a conditional probability parameter. Note that the red nodes and edges in Fig. 2(b) constitute a subgraph of the CANCER WDG and the corresponding probability biased random walk is $\{\theta_0, \theta_1, \theta_3, \theta_5, \theta_{13}, \theta_{17}\}$. Suppose the transition probability from θ_0 to θ_3 and that from θ_0 to θ_4 is 0.3 and 0.7 respectively, and some random walks are as follows: $\{\theta_0, \theta_1, \theta_3, \theta_5, \theta_{13}, \theta_{17}\}$, $\{\theta_0, \theta_1, \theta_4, \theta_{10}, \theta_{16}, \theta_{20}\}$, $\{\theta_0, \theta_1, \theta_4, \theta_9, \theta_{13}, \theta_{17}\}$, $\{\theta_0, \theta_2, \theta_4, \theta_{12}, \theta_{16}, \theta_{20}\}$. The random walks containing θ_0 and θ_4 are more than those containing θ_0 and θ_3. Thus, the distance between θ_0 and θ_4 is closer than that between θ_0 and θ_3, since the transition

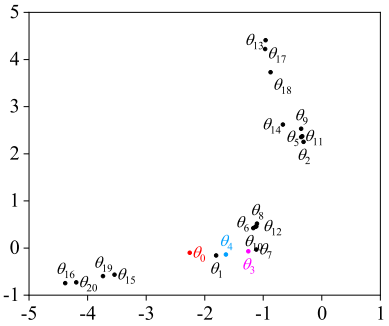


Fig. 3. 2-dimensional embedding of CANCER BN.

probability from the state contained by θ_0 to the state contained by θ_4 is larger than that to θ_3.

4. Multiple probabilistic inferences with BN embedding

In this section, we first present a search-and-calculate algorithm to fulfill probabilistic inferences with embedding vectors. Then, we give the complete framework of multiple probabilistic inferences.

4.1. Probabilistic inference with BNE

Probabilistic inference with BNE aims to calculate the probability given targets τ and evidences ς. Thus, we present how to calculate $P(\tau|\varsigma)$ with the embeddings $\boldsymbol{Y}$.

In the vector space of BNE, the states of variables are represented by different vectors containing the assigned value, and their probabilities have been represented by the distances among vectors. The relevant vector to a specific assigned value of a given variable is the embedding corresponding to the node containing the assigned value. During the sampling of random walks, the larger the transition probability of two nodes, the more the subgraphs (i.e., the random walks) include this node pair, and thus the closer of these two nodes in the embedding vector space. This means that the distance of relevant embedding vectors is smaller, and the concerned variables have a greater probability of co-occurrence(i.e., JP). Then, we calculate the reciprocals of distances among relevant vectors and normalize them in [0, 1] to approximate the probabilities.

Since all the subgraphs are sampled from the *START* node , the distance from *START* to the nodes containing the assigned values represent the co-occurrence probabilities between them. As for a specific state of a given variable, there are several nodes in WDG which represent the same state together. For example, the node θ_{13} and θ_{15} in Fig. 2 represent $Xray = postive$ together. Thus, we use the ratio of the sum of normalized distances between *START* and the nodes containing the assigned value of the given variable, to the sum of normalized distances between *START* and the nodes containing all kinds of values of the given node as the co-occurrence probability between them.

Given a variable X and its value x_i, we define P_m as the probability, which is approximated by the distance from *START* to x_i and that from *START* to all the states of X. Suppose that there are N different possible assigned values for X, and P_m is:

$$P_m(X = x_i) = \frac{r(\theta_0, x_i)}{\sum_{j=1}^{N} r(\theta_0, x_j)} \tag{3}$$

where $r(\cdot,\cdot)$ is defined in Eq. (4) by the sum of the reciprocal of the Euclidean distances:

$$r(\theta_0, x_i) = \sum_{k=1}^{n^{x_i}} \frac{1}{dis(\theta_0, \theta_k^{x_i})} \tag{4}$$

where $\theta_k^{x_i}$ is the kth node in WDG related to $X = x_i$, n^{x_i} is the number of relevant nodes, and $dis(\cdot,\cdot)$ is the Euclidean distance between two nodes' embeddings. Actually, P_m is the marginal probability of the given variable, i.e., $P(X = x_i) \approx P_m(X = x_i)$.

We then provide a method of approximating the JPs of two variables with their assigned values. Suppose that two variables X and Y are assigned to x_i and y_k respectively, then there are three situations for the two assigned variables. We use $P_j(X = x_i, Y = y_k)$ to approximate the joint probability $P(X = x_i, Y = y_k)$:

(1) If X and Y are independent,

$$P_j(X = x_i, Y = y_k) = P_m(X = x_i) \times P_m(Y = y_k) \tag{5}$$

(2) If Y is an ancestor of X,

$$P_j(X = x_i, Y = y_k) = P_m(Y = y_k) \times \sum_{l=1}^{n^{y_k}} \frac{r(\theta_l^{y_k}, x_i)}{\sum_{p=1}^{N} r(\theta_l^{y_k}, x_p)} \tag{6}$$

(3) If X and Y have a common ancestor and Z is the nearest ancestor of X and Y,

$$P_j(X = x_i, Y = y_k) = \sum_{l=1} P_m(Z = z_l) \times \sum_{p=1}^{n^{z_l}} \frac{r(\theta_p^{z_l}, y_k)}{\sum_{q=1} r(\theta_p^{z_l}, y_k)} \times \sum_{p=1}^{n^{z_l}} \frac{r(\theta_p^{z_l}, x_i)}{\sum_{q=1} r(\theta_p^{z_l}, x_i)} \tag{7}$$

Following, we use P_m and P_j to approximate the probability of τ given ς by giving Eq. (8) according to the Bayes formula:

$$P(\tau|\varsigma) = \frac{P(\varsigma, \tau)}{P(\varsigma)} \approx \frac{P_j(\tau, \varsigma)}{P_m(\varsigma)} \tag{8}$$

For the situation with multiple evidences, we can approximate the probabilities as follows:

$$P(\tau|\varsigma_1, \ldots, \varsigma_n) \approx \frac{P_j(\tau, \varsigma_1, \ldots \varsigma_n)}{P_m(\varsigma_1, \ldots \varsigma_n)} \tag{9}$$

The idea of probabilistic inference with BNE is summarized in Algorithm 4.

Algorithm 4 Probabilistic Inference with BNE (PIBNE)

Input: Embedding result: $\boldsymbol{Y}$
Evidences: ς
Targets: τ
Output: Probability: $P(\tau|\varsigma)$
1: Obtain all relevant vectors from $\boldsymbol{Y}$
2: **if** $|\varsigma| = 1$ **then**
3: $P_m(\varsigma) \leftarrow \frac{r(\theta_0,\varsigma)}{\sum_{i=1}^{N} r(\theta_0,\eta_i)}$ // Calculate the probability by Eq. (3)
4: Calculate $P_j(\tau, \varsigma)$ by Eq. (5), Eq. (6) and Eq. (7)
5: $P(\tau|\varsigma) \leftarrow \frac{P_j(\tau,\varsigma)}{P_m(\beta,\varsigma)}$
6: **end if**
7: **if** $|\varsigma| > 1$ **then**
8: $P(\tau|\varsigma) \leftarrow \frac{P_j(\tau,\varsigma)}{P_j(\varsigma)}$ // Calculate the probability by Eq. (9)
9: **end if**
10: **return** $P(\tau|\varsigma)$

The time complexity of Algorithm 4 is determined by the total number of vectors of all possible values of the given evidences, denoted as $O(N^2|\varsigma|)$. Note that both N and $|\varsigma|$ are far less than $|\Theta|$. So, the time complexity of Algorithm 4 is far less than $O(|\Theta|^2)$, which means that our proposed method is more efficient than other classical inference algorithms like VE and GS.

Table 2
Statistics of BNs.

BN	Descriptions	Size	# Nodes	# Edges	# Parameters	# Probabilities
CANCER	Cancer diagnosing	Small	5	4	10	20
ALARM	Monitoring system	Medium	37	46	509	752
HEPAR2	Medical diagnosis of liver	Large	70	123	1,453	2,139
LINK	Linkage analyzing	Very Large	724	1,125	14,211	20,462
PATHFINDER	Medical assistance expert system	Very Large	109	195	77,155	97,851

4.2. Framework of multiple probabilistic inferences

Given a BN $\mathcal{B} = (G, \Theta)$ and the task with t times probabilistic inferences $\mathcal{P} = \{P(\tau_i|\varsigma_i), 1 \leq i \leq t\}$, we calculate P_j for different targets and evidences respectively and approximate the probabilities one by one, given by Algorithm 5. Thus, the multiple probabilistic inferences could be fulfilled efficiently, instead of repeated search of CPTs and computations of intermediate probabilities. For t times multiple probabilistic inferences, the time complexity of our method is $O(tN^2|\varsigma_i|)$.

Algorithm 5 Multiple Probabilistic Inferences

Input: BN: $\mathcal{B} = (G, \Theta)$
Window size: w
Embedding dimensionality: d
Number of walks of each node: γ
Length of walks: l
t times multiple probabilistic inferences: $\mathcal{P} = \{P(\tau_i|\varsigma_i), 1 \leq i \leq t\}$
Output: Result of multiple probabilistic inferences: $\mathcal{P}$
1: **Initialization:** $\mathcal{P} \leftarrow \varnothing$
2: $\boldsymbol{Y} \leftarrow$BNE($\mathcal{B}, w, d, l, \gamma$) // BNE by Algorithm 3
3: **for** $i = 1$ to t **do**
4: $\mathcal{P} \leftarrow \mathcal{P} \cup \{$PIBNE($\boldsymbol{Y}, \tau_i, \varsigma_i$)$\}$ // by Algorithm 4
5: **end for**
6: **return** $\mathcal{P}$

5. Experimental study

In this section, we present an extensive experimental study of our BN embedding method for efficient multiple probabilistic inferences. Using several classical BNs, we conduct three sets of experiments to evaluate: (1) the efficiency of our approach compared with traditional probabilistic inference algorithms, (2) the effectiveness of our approach compared with the traditional algorithms, and (3) the impacts of parameters.

5.1. Experimental settings

Datasets. We use 5 BNs from different domains for experiments and summarize the statistics about these datasets in Table 2.

Comparison algorithms. We have carefully chosen several exact and approximate probabilistic inference algorithms to compare with our method.

- VE is an exact probabilistic inference algorithm that searches the optimal elimination order to reduce calculation [8].
- GS is an approximate probabilistic inference algorithm to estimate the probability based on samples achieved from probability distributions, and the effectiveness (resp., efficiency) of the random sampling algorithm is proportional (resp., inversely proportional) to the number of samples [9].
- PGS is a parallel Gibbs sampling method that adopts graph coloring to construct a direct parallelization of the classical sequential scan Gibbs sampler [31].

Evaluation metrics. For effectiveness tests, we adopt mean absolute error (MAE) to measure the accuracy of each probabilistic inference method and use similarity and precision for evaluation w.r.t. random targets and evidences. For efficiency tests, we record the running time of each method for comparison.

- Given one of the multiple probabilistic inferences, denoted by x, the prediction accuracy is evaluated with MAE, defined as:

$$\text{MAE}(x) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{10}$$

where n is the number of predictive values of query nodes, $\hat{y}_i$ is the probability of the predictive value generated by x, and y_i is the corresponding probability inferred by VE, which is regarded as the ground truth for comparison. MAE has a range of [0, 1] with 0 being the perfect inference and 1 the worst.
- Degree of approximation (DoA) is the ratio of the number of times when the results of PIBNE are closer to that of VE than GS and the number of times when GS is better than PIBNE. DoA has a range of $[0, \infty)$, where DoA $= 1$ means that PIBNE performs as well as GS and DoA > 1 means that PIBNE works better than GS.
- Precision is the ratio of the samples predicted to be positive as real positive samples. Given the value of the target variables (i.e., state of target variables) corresponding to the maximum probability by VE as the ground truth, $precision = TP/(TP + FP)$, where true positive (TP) is the number of cases that PIBNE has the same prediction with VE and false positive (FP) is the number of cases that PIBNE is different from VE. Precision has a range of [0, 1] with 1 being the best prediction and 0 the worst.

Implementation. We implement all algorithms based on Python, and conduct all experiments on a machine with an Intel i7-6700 3.4 GHz CPU and 64 GB RAM, running Windows 10 operating system. All targets and evidences in probabilistic inferences for experiments are generated randomly. Each experiment is repeated for 5 times and the average is reported here. We set the dimension of embedding (d) as 20, the number of walks (γ) as 100, and iterations for GS and PGS as 1000.

5.2. Experimental results

5.2.1. Efficiency

In this set of tests, we first evaluate the impacts of network size on the efficiency of our BN embedding method BNE and probabilistic inference method PIBNE, and then we evaluate the efficiency of our multiple probabilistic inference method MPI compared with VE, GS and PGS.

Exp-1: Impacts of t. To evaluate the efficiency of our multiple probabilistic inference method MPI, we compare the running time of MPI with those of VE, GS and PGS by varying t, the number of inferences. All the targets and evidences for the inference tasks are generated randomly and the same inferences are set up for different methods to guarantee fairness. The results are reported in Fig. 4.

The results tell us that: (a) our method MPI outperforms other methods when the number of inferences is large enough on all BNs, since each inference of MPI uses the PIBNE method that takes much less time than other methods; (b) the efficiency

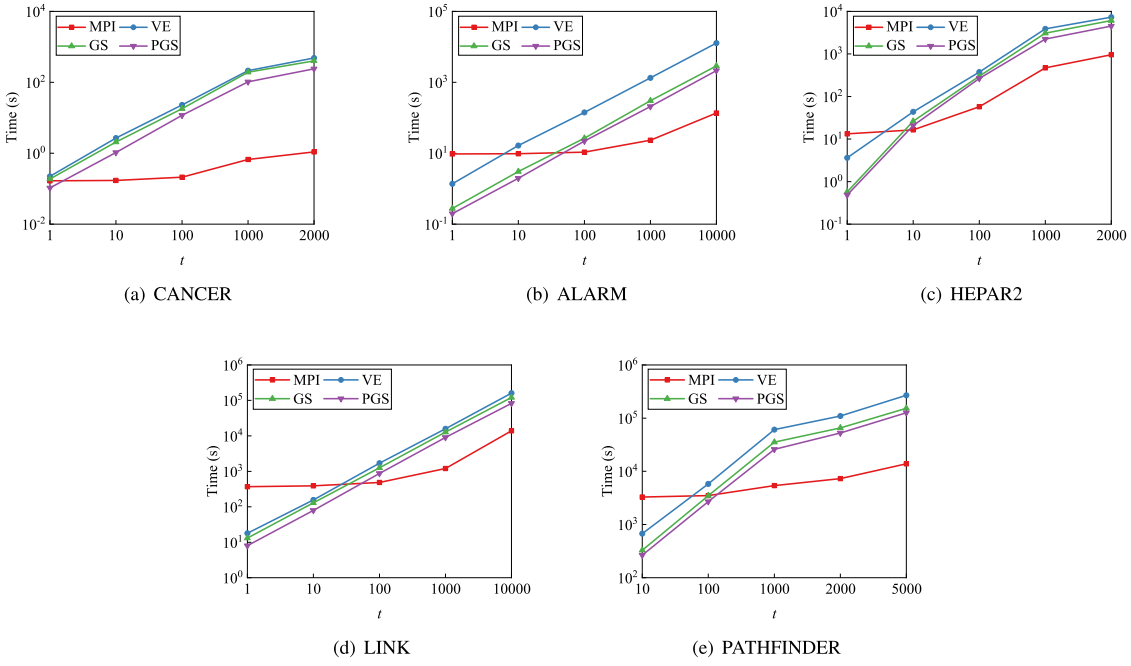


Fig. 4. Impacts of t on running time of MPI.

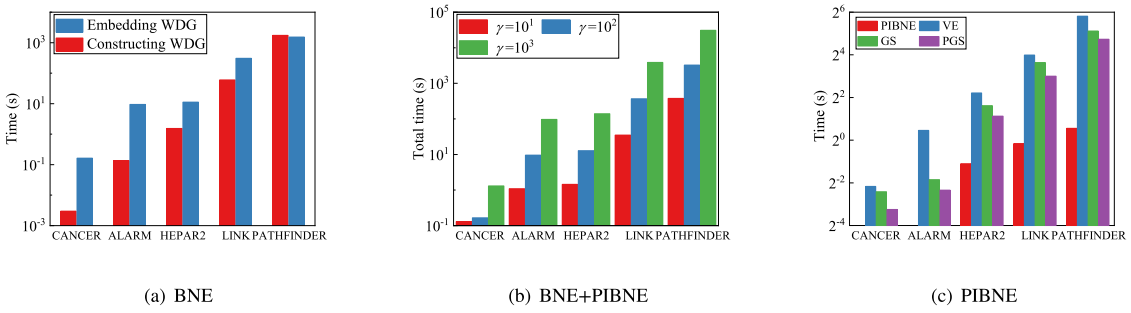


Fig. 5. Impacts of network size on efficiency of BNE and PIBNE.

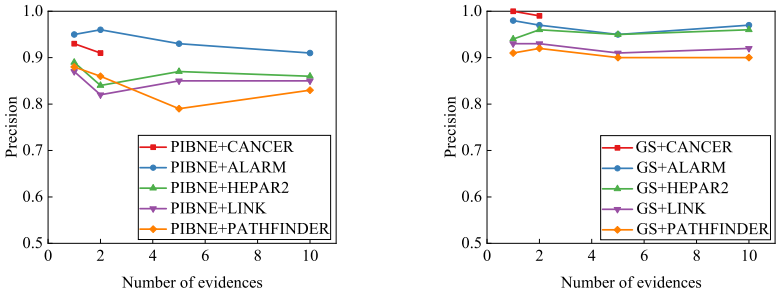
improvement of MPI increases with the increase of t. Indeed, on the largest BN PATHFINDER, MPI is almost 10 times faster than the second-fastest method PGS when the number t is larger than 5000. This is because MPI transforms the original BN into low dimensional embedding and takes PIBNE to do each inference on the embedding, which could reuse some states of variables of the BN and improve the efficiency compared with other methods. Thus, the larger the value of t, the more states of variables could be reused, which means that our method is insensitive to the number of inferences. This verifies the efficiency of our proposed method MPI.

Exp-2: Impacts of network size. To evaluate the impacts of network size, we record the running time of BNE and PIBNE on 5 BNs with different sizes. Since BNE consists of WDG construction and embedding, we report the running time of these two parts and the total execution time of BNE in Fig. 5(a) and Fig. 5(b), respectively. Moreover, we compare our inference method PIBNE with VE, GS and PGS by choosing targets and evidences randomly. The running time of these methods is reported in Fig. 5(c).

The results tell us that: (a) our method PIBNE outperforms other methods on all datasets by large margins, especially for the large and complex BNs. In fact, PIBNE is almost 10 times faster than the second-fastest method PGS and 20 times faster than the exact inference method VE; (b) the running time of WDG construction and that of embedding increase w.r.t. the increased network size. Indeed, the embedding takes most of the time when the number of original conditional probabilities is small while the generation of WDG takes most of the time when the number of original conditional probabilities is large; (c) the running time of BNE increases linearly w.r.t. the increased network size, which means that our method BNE can scale to large BNs. This verifies the efficiency of our proposed method.

5.2.2. Effectiveness

In this set of tests, we adopt the probabilities inferred by the exact inference method VE as the ground truth and evaluate the effectiveness of our proposed methods by comparing PIBNE with



(a) PIBNE (b) GS

Fig. 6. Comparison of precision on various datasets.

Table 3
$P(\tau|\varsigma)$ by VE, GS and PIBNE.

Probability	VE	GS	PIBNE	Probability	VE	GS	PIBNE	Probability	VE	GS	PIBNE
$P(\tau_{11}\|\varsigma_1)$	0.014	0.170	0.183	$P(\tau_{41}\|\varsigma_4)$	0.012	0.080	0.048	$P(\tau_{71}\|\varsigma_7)$	0.100	0.080	0.091
$P(\tau_{12}\|\varsigma_1)$	0.987	0.730	0.817	$P(\tau_{42}\|\varsigma_4)$	0.988	0.920	0.952	$P(\tau_{72}\|\varsigma_7)$	0.900	0.920	0.909
$P(\tau_{21}\|\varsigma_2)$	0.010	0.120	0.010	$P(\tau_{51}\|\varsigma_5)$	0.097	0.092	0.200	$P(\tau_{81}\|\varsigma_8)$	0.040	0.350	0.038
$P(\tau_{22}\|\varsigma_2)$	0.990	0.880	0.990	$P(\tau_{52}\|\varsigma_5)$	0.903	0.908	0.800	$P(\tau_{82}\|\varsigma_2)$	0.960	0.650	0.962
$P(\tau_{31}\|\varsigma_3)$	0.050	0.135	0.019	$P(\tau_{61}\|\varsigma_6)$	0.180	0.200	0.151	$P(\tau_{91}\|\varsigma_9)$	0.000	0.000	0.020
$P(\tau_{32}\|\varsigma_3)$	0.893	0.731	0.019	$P(\tau_{62}\|\varsigma_6)$	0.062	0.100	0.305	$P(\tau_{92}\|\varsigma_9)$	0.010	0.000	0.316
$P(\tau_{33}\|\varsigma_3)$	0.058	0.134	0.623	$P(\tau_{63}\|\varsigma_6)$	0.758	0.700	0.545	$P(\tau_{93}\|\varsigma_9)$	0.990	1.000	0.664

Table 4
Comparison of MAE on various BNs.

BN	MPI	GS	PGS
CANCER	0.1229	0.1002	0.0821
ALARM	0.0587	0.0309	0.0359
HEPAR2	0.0805	0.0673	0.0627
LINK	0.0857	0.0755	0.0701
PATHFINDER	0.1555	0.1009	0.0871

Table 5
Comparison of MAE on LINK with different t.

t	MPI	GS	PGS
10	0.0831	0.0672	0.0681
20	0.0741	0.0725	0.0729
50	0.0869	0.0715	0.0753
100	0.0810	0.0701	0.0762

Table 6
MAE of MPI on various BNs with different t.

t	CANCER	ALARM	HEPAR2	LINK	PATHFINDER
10	0.1293	0.0490	0.0813	0.0895	0.1470
20	0.1388	0.0683	0.0723	0.0953	0.1414
50	0.1166	0.0567	0.0781	0.1102	0.1756
100	0.1070	0.0607	0.0902	0.0876	0.1578

GS on DoA and precision, and comparing MPI with GS and PGS on MAE.

Exp-3: PIBNE versus GS. To test the effectiveness of our probabilistic inference method PIBNE, we randomly generate the targets and evidences for PIBNE, VE and GS, and calculate the probability of $P(\tau|\varsigma)$ with different sizes of ς. A random set of inferred $P(\tau|\varsigma)$ from the tests is reported in Table 3, where the jth row represents the probability $P(\tau_{ij}|\varsigma_i)$ for the ith probabilistic inference $P(\tau_i|\varsigma_i)$.

The results tell us that: (a) the probabilities inferred by PIBNE are close to those of VE and GS, and the DoA of PIBNE is 1.1, which means that PIBNE performs as well as GS in the probabilistic inference task; (b) most of the probability orders are also preserved by PIBNE. This verifies the effectiveness of our method.

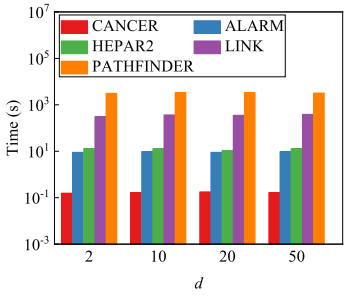
To further evaluate our method, we also test PIBNE compared with GS on precision by using the value of the target variable with the maximum probability of VE as the ground truth. The results of precision are compared and shown in Fig. 6. The results tell us that: (a) PIBNE achieves a high precision (greater than 75%). In fact, the wrong inferences of PIBNE are caused by the information loss during BNE; (b) the precision of PIBNE stays stable with the increase of the number of evidence variables, which verifies that our method is insensitive to the number of evidence variables; (c) the precision of PIBNE is close to that of GS, which means that our method is similar to GS and reasonable to probabilistic inferences.

Exp-4: MPI versus GS and PGS. To evaluate the accuracy of our multiple probabilistic inference method MPI, we compared MPI with GS and PGS by choosing the targets τ and evidences ς from the original set of values of variables randomly. The results by varying the number of inferences on all 5 different BNs are shown in Table 4 and the results on LINK BN are shown in Table 5. The results tell us that: (a) MPI performs stably with the increase of the number of inferences; (b) MPI works as well as GS and PGS on all datasets. This verifies the effectiveness of our proposed method.

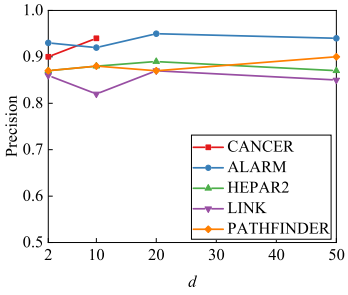
5.2.3. Impacts of parameters

Finally, we evaluated the impacts of parameters on the running time, precision and MAE of our method.

Exp-5: Impacts of t. To evaluate the impacts of t on the accuracy of MPI, we vary t from 10 to 100 and fix other parameters on all BNs. The MAE results are shown in Table 6. The results tell us that: (a) the MAE of MPI is stable with the increase of t, which means that our method MPI is insensitive to the number of inferences; (b) the MAE among different BNs is close to each other, which shows that MPI is robust to BNs with different sizes. Thus, we fix t to 20 on other tests.

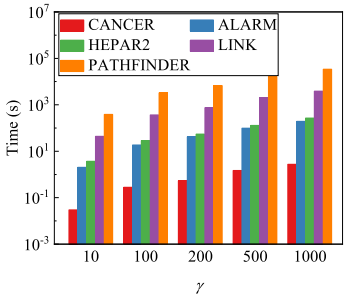


(a) Running time of BNE

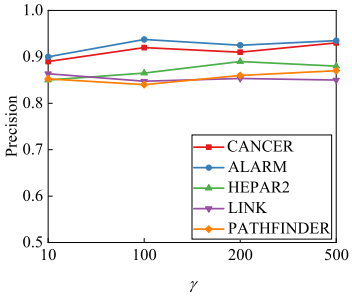


(b) Precision of PIBNE

Fig. 7. Impacts of the dimension.



(a) Running time of BNE



(b) Precision of PIBNE

Fig. 8. Impacts of the number of walks.

Exp-6: Impacts of d. To evaluate the impacts of the dimension of the BN embedding, we vary the dimension from 2 to 50 and fix other parameters. The running time of BNE and precision of PIBNE are shown in Fig. 7(a) and Fig. 7(b), respectively. The results tell us that: (a) the running time of BNE is almost the same no matter what dimensions of BN embedding are. This denotes that the dimension has little impact on the efficiency of BNE; (b) the precision of PIBNE fluctuates with the d increases, precision grows when d increases from 2 to 20 and the precision declines when d becomes greater than 20 in most cases. PIBNE has a higher precision when d is 20. The precision cannot be improved obviously by increasing d. Thus, we fix the dimension to 20 for other tests.

Exp-7: Impacts of γ. To evaluate the impacts of the number of walks, we vary the number of walks from 10 to 1000 and fix other parameters. The running time of BNE and precision of PIBNE are shown in Fig. 8(a) and Fig. 8(b), respectively. The results tell us that: (a) the running time of BNE increases w.r.t. the increase random walks. Indeed, the running time of BNE is proportional to the number of walks; (b) the precision of PIBNE increases slightly when the number of walks increases, since the larger number of walks could generate more powerful embedding for inferences. Thus, we fix the number of walks to 100 for other tests.

5.2.4. Analysis of limitations

In order to help users understand our method more clearly, we discuss the limitations of our method: (a) the precision may not be as good as GS in some situations, and (b) our methods cannot predict as accurately as GS on MAE. Fig. 6 shows that the precision of GS is higher than that of PIBNE in most cases, while the DoA of Table 3 shows that the accuracy of PIBNE is higher than GS in some cases. Tables 4 and 5 shows that MAE of our method is greater than GS and PGS. The results tell us that our method performs a bit worse than GS on effectiveness.

We next explain the reasons. Although the WDG preserves both structure and parameters of BN completely, the sampling of random walks and the embedding method cause the loss of information. Besides, there is another loss due to the approximation of probabilities calculated by our method. Thus, the precision and MAE of our methods perform worse than GS. To address these limitations of probabilistic methods, it deserves full treatment in the future.

5.2.5. Summary

From these experimental results on several BNs from different domains, we find the following:

- Our method PIBNE outperforms VE, GS and PGS on efficiency when the BNE is available. For the multiple probabilistic inferences, our method MPI is 10 times faster than the second-fastest method PGS and could scale to large and complex BNs by reusing some results of common calculations in the process of inferences.
- Our methods PIBNE and MPI are effective to the probabilistic inference task and work as well as traditional approximate inference method GS. In fact, PIBNE obtains over 75% precision with the MAE less than 0.2 for different probabilistic inferences on all BNs.

6. Conclusions and future work

We propose BNE, a novel framework for BN embedding based on Deepwalk and PIBNE, a probabilistic inference method with BNE based on distances of vectors. Experiments on BNs with different sizes show the efficiency and effectiveness of our methods. PIBNE is several times faster than its competitors and has

almost the same precision and error for different BNs. The more the tasks of multiple probabilistic inferences, the more efficient of our method for probabilistic inference compared with the competitors. The execution time of BNE mainly depends on the size of BN and the number of random walks, while the execution time of PIBNE mainly depends on the dimensions of embedding and number of walks. Our proposed methods establish the framework for efficient evaluation of multiple probabilistic inferences and overcome the efficiency bottleneck of classical inference algorithms.

As the initial exploration of BNE, the upper bound of information loss w.r.t. the original BN should be further studied to guarantee the soundness theoretically. The precision of PIBNE has the space to improve and the high MAE still occurs sometimes due to the information loss during BNE. Quantifying the uncertainty of the inference results might be useful when our proposed methods is used. We are now considering these issues. In the future, we will make optimization on the findings achieved in this paper and focus on BNE with other deep learning based methods. Meanwhile, it is worthwhile to consider the embedding based on joint knowledge inferences of BN and other knowledge frameworks, such as first-order logical rules and knowledge graphs.

CRediT authorship contribution statement

Jiahui Wang: Conceptualization, Methodology, Experiment, Writing – original draft. **Kun Yue:** Supervision, Methodology, Writing – review & editing. **Liang Duan:** Supervision, Methodology, Writing – review & editing. **Zhiwei Qi:** Data curation, Resources, Writing – review. **Shaojie Qiao:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

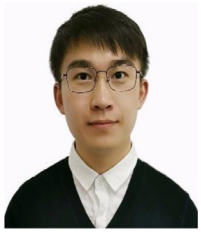
Acknowledgments

This work was supported by the National Natural Science Foundation of China [grant numbers U1802271, 61802035, 62002311]; the Science Foundation for Distinguished Young Scholars of Yunnan Province, China [grant number 2019FJ011]; the China Postdoctoral Science Foundation, China [grant number 2020M673310]; the Major Project of Science and Technology of Yunnan Province, China [grant number 202002AD080002-1-B]; the Cultivation Project of Donglu Scholar of Yunnan University, China.

References

[1] Judea Pearl, Fusion, propagation, and structuring in belief networks, Artificial Intelligence 29 (3) (1986) 241–288.
[2] Kung-Jeng Wang, Jyun-Lin Chen, Kung-Min Wang, Medical expenditure estimation by Bayesian network for lung cancer patients at different severity stages, Comput. Biol. Med. 106 (2019) 97–105.
[3] Wanke Yu, Chunhui Zhao, Online fault diagnosis for industrial processes with Bayesian network-based probabilistic ensemble learning strategy, IEEE Trans. Autom. Sci. Eng. 16 (4) (2019) 1922–1932.
[4] Madjid Tavana, Amir-Reza Abtahi, Debora Di Caprio, et al., An artificial neural network and Bayesian network model for liquidity risk assessment in banking, Neurocomputing 275 (2018) 2525–2554.
[5] Sin Yi Lim, Mohd Saberi Mohamad, Lian En Chai, et al., Investigation of the effects of imputation methods for gene regulatory networks modelling using dynamic bayesian networks, in: Proceedings of the 13th International Conference on Distributed Computing and Artificial Intelligence, Springer, 2016, pp. 413–421.
[6] Gregory F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artificial Intelligence 42 (2–3) (1990) 393–405.
[7] Paul Dagum, Michael Luby, Approximating probabilistic inference in Bayesian belief networks is NP-hard, Artificial Intelligence 60 (1) (1993) 141–153.
[8] Cory J. Butz, Jhonatan S. Oliveira, Anders L. Madsen, BayesIan network inference using marginal trees, Internat. J. Approx. Reason. 68 (2016) 127–152.
[9] Alexander Terenin, Daniel Simpson, David Draper, Asynchronous gibbs sampling, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 144–154.
[10] Hongyun Cai, Vincent W. Zheng, Kevin Chen-Chuan Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Trans. Knowl. Data Eng. 30 (9) (2018) 1616–1637.
[11] Palash Goyal, Emilio Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowl.-Based Syst. 151 (2018) 78–94.
[12] Zhiwei Qi, Kun Yue, Liang Duan, Jiahui Wang, Shaojie Qiao, Xiaodong Fu, Matrix factorization based Bayesian network embedding for efficient probabilistic inferences, Expert Syst. Appl. 169 (2021) 114294.
[13] Peng Cui, Xiao Wang, Jian Pei, Wenwu Zhu, A survey on network embedding, IEEE Trans. Knowl. Data Eng. 31 (5) (2018) 833–852.
[14] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, Wenwu Zhu, Asymmetric transitivity preserving graph embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1105–1114.
[15] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Graph embedding: A general framework for dimensionality reduction, in: Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, IEEE, 2005, pp. 830–837.
[16] Daixin Wang, Peng Cui, Wenwu Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1225–1234.
[17] Xiaokai Wei, Linchuan Xu, Bokai Cao, Philip S. Yu, Cross view link prediction by learning noise-resilient representation consensus, in: Proceedings of the 26th International Conference on World Wide Web, ACM, 2017, pp. 1611–1619.
[18] Bryan Perozzi, Rami Al-Rfou, Steven Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 701–710.
[19] Zhiwei Qi, Jiahui Wang, Kun Yue, Shaojie Qiao, Jin Li, Methods and applications of graph embedding:A survey, Acta Electron. Sin. 48 (4) (2020) 808–818.
[20] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, ACM, 2015, pp. 1067–1077.
[21] Cheng Yang, Maosong Sun, Zhiyuan Liu, Cunchao Tu, Fast network embedding enhancement via high order proximity approximation, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, ACM, 2017, pp. 3894–3900.
[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, Jeff Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.
[23] Masakazu Ishihata, Shan Gao, Shin-ichi Minato, Fast message passing algorithm using ZDD-based local structure compilation, in: Advanced Methodologies for Bayesian Networks, 2017, pp. 117–128.
[24] Hamza Agli, Philippe Bonnard, Christophe Gonzales, Pierre-Henri Wuillemin, Incremental junction tree inference, in: Proceedings of International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, 2016, pp. 326–337.
[25] Chao Li, Maomi Ueno, An extended depth-first search algorithm for optimal triangulation of Bayesian networks, Internat. J. Approx. Reason. 80 (2017) 294–312.
[26] Vibhav Gogate, Abhay Jha, Deepak Venugopal, Advances in lifted importance sampling, in: Proceedings of the 26th AAAI Conference on Artificial Intelligence, ACM, 2012.
[27] Md Vasimuddin, Sriram P. Chockalingam, Srinivas Aluru, A parallel algorithm for Bayesian network inference using arithmetic circuits, in: Proceedings of 2018 IEEE International Parallel and Distributed Processing Symposium, IEEE, 2018, pp. 34–43.
[28] Giso H. Dal, Alfons W. Laarman, Peter J.F. Lucas, Parallel probabilistic inference by weighted model counting, in: Proceedings of International Conference on Probabilistic Graphical Models, 2018, pp. 97–108.
[29] Lu Zheng, Ole Mengshoel, Optimizing parallel belief propagation in junction treesusing regression, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 757–765.

[30] Robert Nishihara, Iain Murray, Ryan P. Adams, Parallel MCMC with generalized elliptical slice sampling, J. Mach. Learn. Res. 15 (1) (2014) 2087–2112.
[31] Joseph Gonzalez, Yucheng Low, Arthur Gretton, Carlos Guestrin, Parallel gibbs sampling: From colored fields to thin junction trees, in: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, ACM, 2011, pp. 324–332.
[32] Aditya Grover, Jure Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855–864.
[33] Hanjun Dai, Bo Dai, Le Song, Discriminative embeddings of latent variable models for structured data, in: Proceedings of the 33rd International Conference on Machine Learning, ACM, 2016, pp. 2702–2711.
[34] Qianjin Zhang, Ronggui Wang, Juan Yang, Lixia Xue, Knowledge graph embedding by translating in time domain space for link prediction, Knowl.-Based Syst. (2020) 106564.
[35] Yuxiao Dong, Nitesh V. Chawla, Ananthram Swami, Metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 135–144.
[36] Guolei Sun, Xiangliang Zhang, A novel framework for node/edge attributed graph embedding, in: Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2019, pp. 169–182.
[37] Zhilin Yang, William Cohen, Ruslan Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: Proceedings of the 33rd International Conference on Machine Learning, PMLR, 2016, pp. 40–48.
[38] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, Qiaozhu Mei, Deepcas: An end-to-end predictor of information cascades, in: Proceedings of the 26th International Conference on World Wide Web, ACM, 2017, pp. 577–586.
[39] Jörg Schlötterer, Martin Wehking, Fatemeh Salehi Rizi, Michael Granitzer, Investigating extensions to random walk based graph embedding, in: Proceedings of 2019 IEEE International Conference on Cognitive Computing, IEEE, 2019, pp. 81–89.
[40] Darwiche Adan, A differential approach to inference in Bayesian networks, J. ACM 50 (2003) 280–305.



Jiahui Wang received a B.S. degree in computer science from Yunnan University in 2017. He is currently a Ph.D. candidate at the School of Information Science and Engineering, Yunnan University. His research interests include knowledge engineering, big data analysis and applications.

Kun Yue received the B.S., M.S. and Ph.D. degrees in computer science from Yunnan University, Fudan University and Yunnan University in 2001, 2004 and 2009, respectively. He is currently a professor and Ph.D. supervisor at the School of Information Science and Engineering, Yunnan University, Kunming, China. He has authored more than 80 papers in peer-reviewed journals and conferences, such as IEEE TCYB, IEEE TSC, INS, KBS, DMKD and DASFAA. His current research interests include massive data analysis and uncertainty in artificial intelligence.

Liang Duan received the B.S., M.S. and Ph.D. degrees in computer science from Beihang University, Yunnan University and Beihang University in 2009, 2014 and 2019, respectively. He has been a postdoctor since 2020 at the School of Information Science and Engineering, Yunnan University, Kunming, China. He has published more than 10 peer-reviewed journal and conference papers, such as IEEE TKDE, KAIS, ICDM, WSDM and DASFAA. His current research interests include databases and social data analysis.

Zhiwei Qi received the B.S. and M.S. degrees in educational technology from Shanxi Datong University, Yunnan University in 2010, 2013 respectively. He is currently a Ph.D. candidate and lecturer at the School of Information Science and Engineering, Yunnan University. His research interests include massive data analysis and uncertainty in artificial intelligence.

Shaojie Qiao received the B.S. and Ph.D. degrees from Sichuan University, Chengdu, China, in 2004 and 2009 respectively. From 2007 to 2008, He worked as a visiting scholar in the School of Computing at the National University of Singapore. He is currently a professor with the School of Software Engineering, Chengdu University of Information Technology, Chengdu, China. He has led several research projects in the areas of databases and data mining. He has authored more than 60 high quality papers, and co-authored more than 90 papers. His research interests include location-based social networks and trajectory data mining.