Artificial Intelligence 174 (2010) 442–448

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

A note on minimal d-separation trees for structural learning

Binghui Liu [a], Jianhua Guo [a,*], Bing-Yi Jing [b]

[a] Key Laboratory for Applied Statistics of MOE and School of Mathematics and Statistics, Northeast Normal University, Changchun 130024, Jilin Province, China
[b] Department of Mathematics, Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Received 2 April 2009
Received in revised form 19 January 2010
Accepted 23 January 2010
Available online 2 February 2010

Keywords:
Bayesian network
Clique tree
Minimal d-separation tree
Minimal triangulation
Separation tree
Structural learning

ABSTRACT

Structural learning of a Bayesian network is often decomposed into problems related to its subgraphs, although many approaches without decomposition were proposed. In 2006, Xie, Geng and Zhao proposed using a d-separation tree to improve the power of conditional independence tests and the efficiency of structural learning. In our research note, we study a minimal d-separation tree under a partial ordering, by which the maximal efficiency can be obtained. Our results demonstrate that a minimal d-separation tree of a directed acyclic graph (DAG) can be constructed by searching for the clique tree of a minimal triangulation of the moral graph for the DAG.

1. Introduction

Bayesian networks (BNs), also known as directed acyclic graphical models, are useful probabilistic graphical models that can be represented by DAGs. For a Bayesian network, two components are involved. First, the DAG is the qualitative component which represents dependence and independence relationships: the absence of some directed edges represents the existence of certain conditional independence relationships between variables, and the presence of edges may represent the existence of direct dependence relationships or causal relationships [10,17]. Second, the joint probability distribution is the quantitative component that expresses the strength of association between variables. We have special interest in the structure recovery of DAGs. Several methods for structural learning of DAGs were considered and there are mainly two basic approaches of structural learning [23]: constraint-based approach and search-and-score approach. We will introduce some more constraint-based algorithms. As mentioned in [23], in a constraint-based algorithm, search for d-separators of pairs of variables is a major problem for orientation of edges and for structure recovery of a DAG. Here a d-separator is a subset of variables by which the variable pairs are d-separated. Verma and Pearl [21] presented the IC algorithm and searched for a d-separator S from all possible variable subsets such that two variables u and v are conditionally independent on S. As the search for d-separators is often time-consuming, many algorithms were proposed to speed up the search. For example, the PC algorithm, an iterative algorithm, searched only for the d-separators contained in the variables that are adjacent to u and v in each step [19], where two variables are said to be adjacent if there is an edge between the two variables. Decomposition approaches [7,22] are also useful approaches to speed up the search. Using these approaches, the original learning question can be decomposed into some sub-questions with smaller dimensions.

For many constraint-based algorithms, there are two steps to recover DAG structures [23]. First, learn the moral graph of the target DAG applying Markov blanket learning algorithms, where the Markov blanket for a variable u is defined to

* Corresponding author.
E-mail addresses: liubh024@yahoo.com.cn (B. Liu), jhguo@nenu.edu.cn (J. Guo), majing@ust.hk (B.-Y. Jing).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.01.002

be the set of variables composed of u's parents, its children, and its children's other parents [16]. Second, perform further independence tests for edge orientation based on the moral graph learned in the first step. To speed up the second step, Xie, Geng and Zhao [22] proposed algorithms adopting a decomposition approach, which were supported by a useful and important definition of d-separation tree. They introduced this definition to depict separations and conditional independencies between sets of random variables, and we will describe the definition again in detail in the following section. Given a d-separation tree T, the problem of searching for d-separators in a large network can be localized to smaller subnetworks, that is, d-separators are only searched for in each node of the tree separately (see Theorem 2 in [22]). Then the number of the further independence tests mentioned in the second step can certainly be reduced.

Different d-separation trees often carry corresponding efficiencies. The efficiency can be defined from several different angles for different purposes. For example, if the aim is to search for the tree where the largest node has the minimum size, the efficiency may be defined inversely proportional to the number of vertices in the largest node of the tree. Our aim is slightly different from that one and here the efficiency is defined inversely proportional to the maximum number of possible tests, which will be described in detail in the following section. To describe an appropriate d-separation tree with high efficiency, we first introduce the definition of minimal d-separation trees for a DAG. Our paper focuses on the characterization and construction of a minimal d-separation tree. Xie, Geng and Zhao (2006) already proposed a sufficient condition for d-separation trees. Based on their work, we show that any minimal d-separation tree of a DAG is equivalent to a minimal separation tree of the moral graph for the DAG, which is in turn equivalent to a clique tree of a minimal triangulation of the moral graph. According to these equivalences, some useful algorithms of searching for minimal triangulations and clique trees are available for us to construct a minimal d-separation tree, which leads to a minimal number of the further independence tests mentioned above required for orientation of edges. The definitions, such as moral graph, clique tree and triangulation [10,12,13], will be introduced in the next section in detail.

In Section 2, we introduce the notation and definitions. As our paper is mainly inspired by Xie, Geng and Zhao, we mainly follow the terminology of [22]. Section 3 discusses the characterization and construction of a minimal d-separation tree of a DAG. In Section 4, we make a conclusion for our paper.

2. Notation and definitions

We follow the terminology from [13,22], unless noted otherwise.

2.1. Undirected graphs and separation trees

We consider simple and connected graphs. Let $G = (V, E)$ denote an undirected graph, where V is the vertex set and E is the set of undirected edges. An undirected edge between two vertices u and v is denoted by (u, v). $A \subseteq V$ *induces* a subgraph $G_A = (A, E_A)$, where $E_A = E \cap (A \times A)$. A subset of V is called *complete* if every pair of vertices in the subset is connected by an edge. A complete subset that is maximal w.r.t. inclusion is called a *clique*. A *path* p between two vertices u and v is a sequence of vertices $w_0 = u, w_1, \ldots, w_n = v$ with $(w_{i-1}, w_i) \in E$, for $1 \leqslant i \leqslant n$ $(n \geqslant 1)$ and $w_i \neq w_j$ for $1 \leqslant i, j \leqslant n$. A *cycle* is a path with $w_0 = w_n$. A *chord* of the cycle is a pair of vertices w_i, w_j such that $(w_i, w_j) \in E$ but $j \neq i-1$, $i+1$. Two subsets A, $B \subseteq V$ with $A \cap B = \emptyset$ are said to be *separated* by $C \subseteq V$ in G if all paths from A to B pass through C, i.e., for each vertex $u \in A$ and each vertex $v \in B$, all paths from u to v intersect C at some vertices. The partition (A, B, S) of V is said to be a *decomposition* of the undirected graph G, if (1) S separates A and B in G, and (2) S is complete in G. Then G can be *decomposed* into subgraphs $G_{A\cup S}$ and $G_{B\cup S}$.

Let $\mathcal{C} = \{C_1, \ldots, C_H\}$, such that $\bigcup_{h=1}^{H} C_h = V$. T denotes a tree whose node set is $\mathcal{C}$ and edge set is E_T. Inspired by the definition of d-separation tree, Definition 1 in [22], here we propose a similar definition of *separation tree* for undirected graphs.

Definition 1. A tree $T = (\mathcal{C}, E_T)$ is said to be a separation tree of an undirected graph $G = (V, E)$ if for each edge $(C_i, C_j) \in E_T$, $V_1 \backslash (C_i \cap C_j)$ and $V_2 \backslash (C_i \cap C_j)$ are separated by $C_i \cap C_j$ in G, where $V_k = \bigcup_{C\in\mathcal{C}_k} C$, $k = 1, 2$; $\mathcal{C}_1$ and $\mathcal{C}_2$ are node sets of T_1 and T_2, which are obtained by removing edge (C_i, C_j) from tree T.

If for each edge $(C_i, C_j) \in E_T$, $C_i \cap C_j$ is complete in G, we say that T is a *decomposition tree* of G.

A tree $T = (\mathcal{C}, E_T)$ is *reduced* if each vertex set in $\mathcal{C}$ is not contained in another one, that is, $red\{\mathcal{C}\} = \mathcal{C}$, where '*red*' stands for the operation of deleting the smaller of any two sets C_1, C_2 with $C_1 \subseteq C_2$.

2.2. Triangulated graphs and clique trees

A *triangulated graph*, also known as a *chordal graph*, is an undirected graph that contains no cycles of length $\geqslant 4$ without a chord. For any undirected graph $G = (V, E)$ edges can be added so that the resulting graph $G^t = (V, E \cup F)$, called a *triangulation* of the given graph G, is triangulated. A triangulation G^t of G is said to be *minimal* if $G' = (V, E \cup F')$ is not triangulated for every proper subset F' of F.

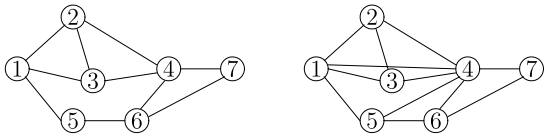

Fig. 1. An undirected graph $G = (V, E)$ (left) and a minimal triangulation G^t (right) of G.

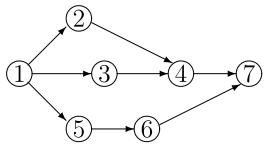

Fig. 2. A DAG G.

Example 1. We see that G^t in Fig. 1 is a triangulation of the undirected graph G, because it contains no cycles of length $\geqslant 4$ without a chord. And G^t is a minimal triangulation, as $G_1 = (V, E \cup \{(1,4)\})$ and $G_2 = (V, E \cup \{(4,5)\})$ are not triangulated. We continue to show a graph G_3 that is not triangulated. Let $G_3 = (V, E \cup \{(3,5),(3,6)\})$ and it looks very much like a triangulation. However, we see that there exists a cycle $(1,2,4,6,5,1)$ without a chord, which is contrary to the definition.

Let $\mathcal{K}_G$ denote the collection of all cliques in an undirected graph G [5]. For any triangulated graph G there exists a subset of the set of trees on $\mathcal{K}_G$ known as *clique trees* [5].

Definition 2. Assume that G is a triangulated graph. A tree T on $\mathcal{K}_G$ is said to be a clique tree of G if T satisfies the *clique-intersection* property: for every pair of distinct cliques K_i, $K_j \in \mathcal{K}_G$, the set $K_i \cap K_j$ is contained in every clique on the path connecting K_i and K_j in the tree.

Note that a clique tree of the triangulated graph G is also said to be a junction tree of cliques for G [6] and for a given triangulated graph G if T is a clique tree of G, T is a decomposition tree of G (see Theorem 4.6 in [6]).

There are many efficient algorithms for computing clique trees of a triangulated graph. The most commonly cited one is the MCS algorithm [20] with an $O(n+m)$ time bound, where n is the number of vertices and m is the number of edges of the given triangulated graph.

2.3. Directed graphs and d-separation trees

Let $G = (V, E)$ denote a DAG, where V is the finite vertex set and E is the set of directed edges. A directed edge from a vertex u to v is denoted as $\langle u, v\rangle$. If there is a directed edge from u to v, v is said to be a *child* of u and u is a *parent* of v. Denote the set of all parents of a vertex v as $pa(v)$ and denote the set of all children of a vertex u as $ch(u)$. We say that u is an *ancestor* of v and v is a *descendant* of u if there is a path between u and v in G and all edges in this path point in the direction of v. We denote the set of all ancestors of a vertex v as $an(v)$ and the set of all descendants of a vertex u as $de(u)$. For $A \subseteq V$, if $\forall v \in A$, $an(v) \subseteq A$, A is said to be an *ancestral set*. For $B \subseteq V$, we let $An(B)$ denote the smallest ancestral set containing B. The *moral graph* $G^m = (V, E^m)$ of a DAG G is defined to be an undirected graph whose edge set is constructed by marrying parents and dropping directions, that is, connecting vertices that have a common child and then making all edges in the graph undirected [13].

A path p is said to be *d-separated* by a set Z in a DAG G, if and only if (1) p contains a 'chain': $u \to v \to w$ or a 'fork': $u \leftarrow v \to w$ such that the middle vertex v is in Z, or (2) p contains a 'collider': $u \to v \leftarrow w$ such that the middle vertex v is not in Z and no descendant of v is in Z [17]. Two distinct sets X and Y of vertices are said to be d-separated by a set Z in G if Z d-separates every path from any vertex in X to any vertex in Y [17]. We also say that Z *d-separates* X and Y in G.

In fact, Z d-separates X, Y in G if and only if Z separates X and Y in the moral graph of $G_{An(X\cup Y\cup Z)}$, which is written as $(G_{An(X\cup Y\cup Z)})^m$ [13].

Example 2. We show an example of d-separation in Fig. 2. $A = \{2\}$ and $B = \{6\}$ are d-separated by $C = \{1\}$. In Fig. 2 there are four paths from $\{2\}$ to $\{6\}$: $(2,1,5,6)$, $(2,4,3,1,5,6)$, $(2,4,7,6)$ and $(2,1,3,4,7,6)$. We see that $(2,1,5,6)$ contains a fork $2 \leftarrow 1 \to 5$ and $(2,4,3,1,5,6)$ contains a fork $3 \leftarrow 1 \to 5$. $(2,4,7,6)$ and $(2,1,3,4,7,6)$ both contain a collider $4 \to 7 \leftarrow 6$, where the middle vertex 7 is not in C and no descendant of 7 is in C.

To depict d-separations and conditional independencies among multiple variable sets, a notion of the d-separation tree was first proposed in [22]. Replacing "an undirected graph" with "a DAG" and replacing "separated" with "d-separated" in Definition 1, we obtain the definition of d-separation tree of a DAG. We will show examples of d-separation trees of Fig. 2 in Fig. 3.

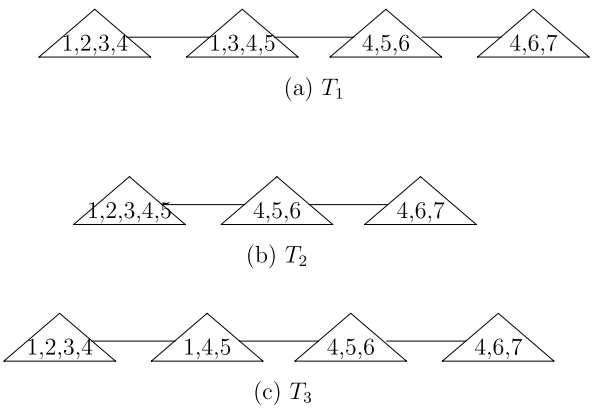

Fig. 3. T_1 and T_2 are two d-separation trees of G in Fig. 2, but neither of them is a minimal d-separation tree; T_3 is a minimal d-separation tree of G.

The following theorem, Theorem 1 in [22], gives a valuable and important property of the d-separation tree, which is central to the decomposing approach to structural learning.

Theorem 1. *(See Xie et al. [22].) Let T be a d-separation tree of a DAG G. Vertices u and v are d-separated by some $S \subseteq V \setminus \{u, v\}$ in G if and only if* (1) *u and v are not contained together in any node of T, or* (2) *there exists a node C that contains both u and v such that a subset S' of C d-separates u and v.*

Theorem 1 implies that the problem of searching for d-separators and constructing the skeleton of a DAG is divided into some smaller problems in nodes of T so that the efficiency can be vastly improved. By using different d-separation trees we often obtain different efficiencies. To depict an appropriate d-separation tree with maximal efficiency, we present the *minimal d-separation tree* of a DAG.

For two collections of subsets $\mathcal{C}$, $\mathcal{C}'$, let $\mathcal{C}' \wedge \mathcal{C} = red\{C \cap C' \colon\ C \in \mathcal{C},\ C' \in \mathcal{C}'\}$. Note that if $\mathcal{C}' \wedge \mathcal{C} = \mathcal{C}'$, for each $C' \in \mathcal{C}'$ there must exist a $C \in \mathcal{C}$ such that $C' \subseteq C$. We proceed to define a partial order '$\prec$' for two trees $T = (\mathcal{C}, E_T)$ and $T' = (\mathcal{C}', E_{T'})$, where $T' \prec T$ if and only if $\mathcal{C}' \wedge \mathcal{C} = \mathcal{C}'$ and $\mathcal{C}' \neq \mathcal{C}$. For any tree $T = (\mathcal{C}, E_T)$, we use $less(T)$ to denote the following collection of trees:

$$\left\{T' = \left(\mathcal{C}', E_{T'}\right) \colon\ T' \text{ is reduced, } \bigcup_{C' \in \mathcal{C}'} C' = V, T' \prec T\right\}.$$

Definition 3. A reduced d-separation tree $T = (\mathcal{C}, E_T)$ of a DAG $G = (V, E)$ is said to be a minimal d-separation tree of G if any tree $T' = (\mathcal{C}', E_{T'}) \in less(T)$ is not a d-separation tree of G.

As mentioned above, given a d-separation tree $T = (\{C_1, \ldots, C_H\}, E_T)$, the efficiency of structural learning can be improved. To construct the global skeleton of a DAG we need to execute conditional independence tests only for $\sum_{j=1}^{H} C_{|C_j|}^2 \cdot 2^{|C_j|-2}$ times at most, which is obviously smaller than $C_{|V|}^2 \cdot 2^{|V|-2}$. Define a cost function $f(\cdot)$ on d-separation trees in this way:

$$f(T) = \sum_{j=1}^{H} C_{|C_j|}^2 \cdot 2^{|C_j|-2}.$$

And define the efficiency function $eff(\cdot)$ as $1/f(\cdot)$. For two reduced d-separation trees T_1 and T_2 of a DAG $G = (V, E)$, if $T_1 \prec T_2$, we have $f(T_1) < f(T_2)$, which will be rigorously proved in Lemma 3 in Section 3. Therefore the d-separation tree T_1 is better than T_2 in terms of efficiency improvement.

Computing a d-separation tree with the smallest cost seems to be the best choice for the following searching work. However, since the problem of computing a d-separation tree with the minimum cost is often with great computational complexity, the related polynomially solvable problem of computing a d-separation tree with the minimal cost becomes more interesting.

Example 3. Consider the DAG G in Fig. 2. Three d-separation trees, $T_i = (\mathcal{C}_i, E_{T_i})$, $i = 1, 2, 3$, are shown in Fig. 3, where

$$\mathcal{C}_1 = \big\{\{1, 2, 3, 4\}, \{1, 3, 4, 5\}, \{4, 5, 6\}, \{4, 6, 7\}\big\},$$
$$\mathcal{C}_2 = \big\{\{1, 2, 3, 4, 5\}, \{4, 5, 6\}, \{4, 6, 7\}\big\},$$
$$\mathcal{C}_3 = \big\{\{1, 2, 3, 4\}, \{1, 4, 5\}, \{4, 5, 6\}, \{4, 6, 7\}\big\}.$$

Taking T_3 for example, $\{2,3\}$, $\{5,6,7\}$ are d-separated by $\{1,4\}$; $\{1,2,3\}$, $\{6,7\}$ are d-separated by $\{4,5\}$; and $\{1,2,3,5\}$, $\{7\}$ are d-separated by $\{4,6\}$. $\mathcal{C}_3 = \mathcal{C}_1 \wedge \mathcal{C}_3 = \mathcal{C}_2 \wedge \mathcal{C}_3$, so $T_3 \prec T_1$ and $T_3 \prec T_2$. Then, $f(T_3) < f(T_1)$ and $f(T_3) < f(T_2)$. In fact, T_3 is a minimal d-separation tree of DAG G.

Similarly, we can give the definition of a *minimal separation tree* of an undirected graph by replacing "d-separation" with "separation" and replacing "a DAG" with "an undirected graph" in Definition 3.

Now that we have given the necessary background and definitions for minimal d-separation trees, we move on to the more important topic: constructing minimal d-separation trees.

3. Construction of minimal d-separation trees

In this section, we focus on the relationships among minimal d-separation trees, minimal separation trees and clique trees of minimal triangulations. Afterwards, we construct minimal d-separation trees taking advantage of algorithms for minimal triangulation, which were widely studied.

Lemma 1. *In a DAG $G = (V, E)$, the following three statements are equivalent*:

(1) *Tree $T = (\mathcal{C}, E_T)$ is a d-separation tree of G*;
(2) *T is a separation tree of G^m*;
(3) *T is a decomposition tree of some triangulation G^t of G^m.*

Proof. (3) $\Rightarrow$ (1): Considering each $(C_i, C_j) \in E_T$, let $K = C_i \cap C_j$. Then $V_1 \backslash K$ and $V_2 \backslash K$ are separated by K in G^t. Since G^m has fewer edges than G^t has, we know that $V_1 \backslash K$ and $V_2 \backslash K$ are separated by K in G^m. Therefore, T is a d-separation tree of G.

(1) $\Rightarrow$ (2): For each $(C_i, C_j) \in E_T$, $V_1 \backslash K$ and $V_2 \backslash K$ are d-separated by K in DAG G, so $V_1 \backslash K$ and $V_2 \backslash K$ are separated by K in $(G_{An((V_1 \backslash K) \cup (V_2 \backslash K) \cup K)})^m = G^m$. Then T is also a separation tree of G^m.

(2) $\Rightarrow$ (3): Let $G^t = (V, \bigcup_{C \in \mathcal{C}} E_C)$ where $E_C = \{(a,b)\colon a, b \in C \text{ and } a \neq b\}$. Considering each $(C_i, C_j) \in E_T$ and let $K = C_i \cap C_j$. For each $x \in V_1 \backslash K$ and each $y \in V_2 \backslash K$, (x, y) is not in any $C \in \mathcal{C}$. In fact, if there exists a subset $C_1 \in \mathcal{C}$, such that $\{x, y\} \subseteq C_1$, we let C_1 be a node of T_1 without loss of generality. Then, $y \in C_1 \backslash K \subseteq V_1 \backslash K$, which is contrary to the fact that $y \in V_2 \backslash K$ and $V_1 \backslash K$, $V_2 \backslash K$ are separated by K in G^m. By the construction of G^t, we know that (x, y) is not an edge of G^t. Then, $V_1 \backslash K$ and $V_2 \backslash K$ are separated by K in G^t and T is a decomposition tree of G^t. G^t can be decomposed step by step into complete subgraphs, therefore, G^t is a triangulated graph.

For each $(x, y) \in E^m$, there is a $C \in \mathcal{C}$ such that $x, y \in C$. In fact, if x, y are not in any set in $\mathcal{C}$, then $d(x, y) > 0$ (see Definition A.1 in [22]). Let $d(x, y) = d(C_3, C_4)$, where $x \in C_3 \in \mathcal{C}$ and $y \in C_4 \in \mathcal{C}$. Let C_5 be the first node on the path of T from C_3 to C_4 and $K = C_3 \cap C_5$. Then, $C_3 \backslash K$ and $C_4 \backslash K$ are separated by K in G^m. Since x and y are not contained in K, we get the result that x and y are separated by K in G^m, which is contrary to the fact that $(x, y) \in E^m$. Therefore G^t is a triangulation of G^m and T is a decomposition tree of G^t. □

Lemma 2. *In a DAG $G = (V, E)$, the following three statements are equivalent*:

(1) *Tree $T = (\mathcal{C}, E_T)$ is a reduced d-separation tree of G*;
(2) *T is a reduced separation tree of G^m*;
(3) *T is a clique tree of some triangulation G^t of G^m.*

Proof. (3) $\Rightarrow$ (1): This point was proved in Theorem 2 of [22].

(1) $\Rightarrow$ (2): T is reduced, so T is also a reduced separation tree of G^m from Lemma 1.

(2) $\Rightarrow$ (3): Let $G^t = (V, \bigcup_{C \in \mathcal{C}} E^t_C)$ where $E^t_C = \{(a,b)\colon a, b \in C \text{ and } a \neq b\}$. By the proof of Lemma 1, G^t is a triangulation of G^m and T is a decomposition tree of G^t, then $\mathcal{C} = \mathcal{K}_{G^t}$. In fact, for each $C \in \mathcal{C}$ there exists $K \in \mathcal{K}_{G^t}$ such that $C \subseteq K$, since C is complete in G^t. Furthermore, for each clique $K \in \mathcal{K}_{G^t}$, if G^t can be decomposed into two subgraphs, K is certainly contained in one. We continue to decompose the subgraph step by step and K is finally contained in some $C \in \mathcal{C}$.

Now we show that T is a clique tree of G^t. Let C and C' be two arbitrary distinct subsets in $\mathcal{C}$. For each node C_i on the path of T between C and C', let C_j be the first node from C_i toward C' on the path. For edge (C_i, C_j), let $K = C_i \cap C_j$. Then, $V_1 \backslash K$ and $V_2 \backslash K$ are separated by K in G^t. Therefore, $C \cap C' \subseteq K \subseteq C_i$. In fact, if there exists a vertex $x \in C \cap C'$ such that x is not in K, we can get the paradoxical result that x and x are separated by K in G^t. By the definition of the clique tree, we know that T is a clique tree of G^t. □

Lemma 3. *Assume that T_1 and T_2 are two reduced d-separation trees of a DAG $G = (V, E)$. If $T_1 \prec T_2$, then $f(T_1) < f(T_2)$.*

Proof. From Lemma 2, there exist two triangulations G^{t1}, G^{t2} of G^m such that T_i is a clique tree of G^{ti} for $i = 1, 2$. From Lemma 2.21 in [13], there is an increasing sequence $G^{t1} = G_0, G_1, \ldots, G_k = G^{t2}$ of triangulated graphs that differ by

exactly one edge, where k is the difference between the numbers of edges in G^{t1} and G^{t2}. Let $G_j = (V, E_j)$ for $0 \leqslant j \leqslant k$. Suppose that $E_1 \backslash E_0 = \{(\alpha, \beta)\}$, and from Lemma 2.19 in [13], in graph G_1, (α, β) is a member of only one clique C^*. Let $\mathcal{K}_{G_1} = \{C_1, \ldots, C_n\}$, $C_q = C^*$, $C_{q1} = C^* \backslash \{\alpha\}$ and $C_{q2} = C^* \backslash \{\beta\}$, where $1 \leqslant q \leqslant n$. Let $\mathcal{K} = \{C_1, \ldots, C_{q-1}, C_{q+1}, \ldots, C_n, C_{q1}, C_{q2}\}$ and then $\mathcal{K}_{G_0} = red(\mathcal{K})$. We see that

$$\begin{aligned}\sum_{C \in \mathcal{K}_{G_0}} C_{|C|}^2 \cdot 2^{|C|} - \sum_{C \in \mathcal{K}_{G_1}} C_{|C|}^2 \cdot 2^{|C|} &\leqslant \sum_{C \in \mathcal{K}} C_{|C|}^2 \cdot 2^{|C|} - \sum_{C \in \mathcal{K}_{G_1}} C_{|C|}^2 \cdot 2^{|C|} \\ &= C_{|C_{q1}|}^2 \cdot 2^{|C_{q1}|} + C_{|C_{q2}|}^2 \cdot 2^{|C_{q2}|} - C_{|C_q|}^2 \cdot 2^{|C_q|} \\ &= C_{|C_{q1}|}^2 \cdot 2^{|C_{q1}|} + C_{|C_{q2}|}^2 \cdot 2^{|C_{q2}|} - 2C_{|C_q|}^2 \cdot 2^{|C_q|-1} < 0.\end{aligned}$$

Similarly, we can prove that $\sum_{C \in \mathcal{K}_{G_{j-1}}} C_{|C|}^2 \cdot 2^{|C|} - \sum_{C \in \mathcal{K}_{G_j}} C_{|C|}^2 \cdot 2^{|C|} < 0$ for each $1 \leqslant j \leqslant k$, which implies that $\sum_{C \in \mathcal{K}_{G_0}} C_{|C|}^2 \cdot 2^{|C|} - \sum_{C \in \mathcal{K}_{G_k}} C_{|C|}^2 \cdot 2^{|C|} < 0$, that is, $f(T_1) < f(T_2)$. □

Theorem 2. *In a DAG $G = (V, E)$, the following three statements are equivalent*:

(1) *Tree $T = (\mathcal{C}, E_T)$ is a minimal d-separation tree of G*;
(2) *T is a minimal separation tree of G^m*;
(3) *T is a clique tree of some minimal triangulation G^t of G^m.*

Proof. (3) ⇒ (1): T is a clique tree of G^t and then T is a reduced d-separation tree of G. Suppose that T is not a minimal d-separation tree of G. There exists a tree T' in $less(T)$ such that T' is a reduced d-separation tree of G. By Lemma 2, we know that there is a triangulation G' of G^m such that T' is a clique tree of G', and furthermore, $G' = (V, E')$, where $E' = \bigcup_{C \in \mathcal{C}'} E_C^t$, $E_C^t = \{(a, b)\colon\ a, b \in C \text{ and } a \neq b\}$. T is a clique tree of G^t. We thus know that $G^t = (V, E^t)$ where $E^t = \bigcup_{C \in \mathcal{C}} E_C^t$. Because $\mathcal{C} \wedge \mathcal{C}' = \mathcal{C}'$ and $\mathcal{C} \neq \mathcal{C}'$, G' has fewer edges than G^t has. G^t is therefore not a minimal triangulation of G^m, which is contrary to the fact above.

(1) ⇒ (2): Assume that T is not a minimal separation tree of G^m, then there exists a tree $T' \in less(T)$ such that T' is a reduced separation tree of G^m. Thus T' is a reduced d-separation tree of G from Lemma 2, which is contrary to the fact that T is a minimal d-separation tree of G.

(2) ⇒ (3): Since T is a minimal separation tree of G^m, we know that T is a clique tree of some triangulation G^t of G^m and $G^t = (V, E^t)$ where $E^t = \bigcup_{C \in \mathcal{C}} E_C^t$. Suppose that G^t is not a minimal triangulation of G^m, i.e., there exists another triangulation G' of G^m that has fewer edges than G^t has. Construct a clique tree $T' = (\mathcal{K}_{G'}, E_{T'})$ of G'. T' is a reduced separation tree of G^m. For each $C' \in \mathcal{K}_{G'}$, there is some $C \in \mathcal{C}$ such that $C' \subseteq C$ and then $\mathcal{C} \wedge \mathcal{K}_{G'} = \mathcal{K}_{G'}$. It is obvious that $\mathcal{C} \neq \mathcal{K}_{G'}$, so T is not a minimal separation tree of G^m, which is contrary to the fact above. Thus, G^t is a minimal triangulation of G^m, and finally we know that T is a clique tree of the minimal triangulation G^t of G^m. □

Example 4. To construct a minimal d-separation tree of the DAG in Fig. 2, we need to compute a minimal triangulation (Fig. 1 (right)) of the moral graph (Fig. 1 (left)), and construct a clique tree of the minimal triangulation. We see that T_3 in Fig. 3(c) is just a clique tree of the minimal triangulation.

According to Theorem 2, a minimal d-separation tree is equivalent to the clique tree of a minimal triangulation of the moral graph. Then efficient algorithms for finding minimal triangulations and clique trees are available for us to construct a minimal d-separation tree. We therefore continue to introduce some algorithms for minimal triangulation.

Several characterizations and algorithms of minimal triangulations on general undirected graphs have been presented since 1976. As Heggernes pointed out [9], there are two approaches to compute and characterize minimal triangulations: one through vertex elimination and the other through minimal separators of the given graph.

The first characterizations of minimal triangulations were given simultaneously by Ohtsuki [14], Ohtsuki et al. [15], and Rose et al. [18] in 1976. These characterizations are strongly connected to vertex elimination. Based on these characterizations, some algorithms, such as the algorithm of Ohtsuki [14], the LEX M algorithm [18] and the MCS-M algorithm [3], were proposed with time bound $O(nm)$, where n is the number of vertices and m is the number of edges of the given graph. For more than 20 years, $O(nm) = O(n^3)$ remained the best known time bound for minimal triangulations. A breakthrough was made in 2004 when a new implementation of LEX M was described by Kratsch and Spinrad [11], which runs in time $O(n^{2.69})$.

There are also some useful algorithms for minimal triangulations based on characterizations strongly connected to minimal separators, such as the LB-Triang algorithm by Berry et al. [1,4], the vertex incremental minimal triangulation algorithm by Berry et al. [2] and the Fast Minimal Triangulation algorithm by Heggernes et al. [8]. The first two algorithms have running times of $O(nm)$ and the third one runs in just $o(n^{2.376})$, which was a simultaneous breakthrough to the LEX M implementation made in 2004.

4. Conclusions

We proposed a partial ordering for d-separation trees, where the fact that one tree is smaller than another implies that the smaller one is more efficient. Under this ordering, a definition of minimal d-separation tree was introduced. We obtained the equivalence between minimal d-separation trees of a DAG G and clique trees of the minimal triangulations of the moral graph of G in Theorem 2. Therefore many efficient algorithms for minimal triangulations and clique trees can be directly used for us to construct minimal d-separation trees of DAGs. Accordingly, the work of performing the further independence tests in the second steps of many constraint-based algorithms of structural learning can be carried out with maximal efficiency.

However, generally speaking, the constructed minimal d-separation tree is not a minimum d-separation tree with the largest efficiency. Our minimal d-separation tree just exhibits a tradeoff between efficiency and computation time.

Acknowledgements

The authors are very grateful to the editor and three reviewers for their insightful comments and helpful suggestions, which greatly improved the quality and presentation of the paper. This research was partially supported by the National Natural Science Foundation of China (Grant Numbers 10701022, 10871038, 10826110 and 10926186), National 973 Key Project of China (2007CB311002) and HK RGC Grant HKUST6117/02.

References

[1] A. Berry, A wide-range efficient algorithm for minimal triangulation, in: Proceedings of the Tenth Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'99), 1999, pp. S860–S861.
[2] A. Berry, P. Heggernes, Y. Villanger, A vertex incremental approach for dynamically maintaining chordal graphs, in: Algorithms and Computation, ISAAC 2003, in: Lecture Notes in Comput. Sci., vol. 2906, Springer, Berlin, 2003, pp. 47–57.
[3] A. Berry, J.R.S. Blair, P. Heggernes, B.W. Peyton, Maximum cardinality search for computing minimal triangulations of graphs, Algorithmica 39 (4) (2004) 287–298.
[4] A. Berry, J.P. Bordat, P. Heggernes, G. Simonet, Y. Villanger, A wide-range algorithm for minimal triangulation from an arbitrary ordering, J. Algorithms 58 (1) (2006) 33–66.
[5] J.R.S. Blair, B. Peyton, An introduction to chordal graphs and clique trees, in: A. George, J.R. Gilbert, J. Liu (Eds.), Graph Theory and Sparse Matrix Computation, Springer-Verlag, New York, 1993.
[6] R.G. Cowell, A.P. David, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer Publications, New York, 1999.
[7] Z. Geng, C. Wang, Q. Zhao, Decomposition of search for V-structures in DAGs, J. Multivariate Anal. 170 (2005) 422–439.
[8] P. Heggernes, J.A. Telle, Y. Villanger, Computing minimal triangulations in time $O(n^{\alpha}\log n) = o(n^{2.376})$, SIAM J. Discrete Math. 19 (4) (2005) 900–913.
[9] P. Heggernes, Minimal triangulations of graphs: A survey, Discrete Math. 306 (2006) 297–317.
[10] F.V. Jensen, Bayesian Networks and Decision Graphs, Springer-Verlag, 2001.
[11] D. Kratsch, J. Spinrad, Minimal fill in $O(n^{2.69})$ time, Discrete Math. 306 (2006) 366–371.
[12] H.G. Leimer, Optimal decomposition by clique separators, Discrete Math. 113 (1993) 99–123.
[13] S.L. Lauritzen, Graphical Models, Clarendon Press, Oxford, 1996.
[14] T. Ohtsuki, A fast algorithm for finding an optimal ordering in the vertex elimination on a graph, SIAM J. Comput. 5 (1976) 133–145.
[15] T. Ohtsuki, L.K. Cheung, T. Fujisawa, Minimal triangulation of a graph and optimal pivoting ordering in a sparse matrix, J. Math. Anal. Appl. 54 (1976) 622–633.
[16] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Mateo, CA, 1988.
[17] J. Pearl, Causality: Models, Reasoning and Inference, Cambridge University Press, Cambridge, 2000.
[18] D. Rose, R.E. Tarjan, G. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM J. Comput. 5 (1976) 146–160.
[19] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction, and Search, 2nd edition, MIT Press, 2000.
[20] R.E. Tarjan, M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, SIAM J. Comput. 13 (1984) 566–579.
[21] T. Verma, J. Pearl, Equivalence and synthesis of causal models, in: Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence, Elsevier, Amsterdam, 1990, pp. 255–268.
[22] X. Xie, Z. Geng, Q. Zhao, Decomposition of structural learning about directed acyclic graphs, Artificial Intelligence 170 (2006) 422–439.
[23] X. Xie, Z. Geng, A recursive method for structural learning of directed acyclic graphs, J. Mach. Learn. Res. 9 (2008) 459–483.