Laminar Structure of Ptolemaic Graphs with Applications ⋆

Ryuhei Uehara [a] and Yushi Uno [b]

[a] *School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan.*

[b] *Department of Mathematics and Information Sciences, Gradate School of Science, Osaka Prefecture University, Osaka, Japan.*

Abstract

Ptolemaic graphs are those satisfying the Ptolemaic inequality for any four vertices. The graph class coincides with the intersection of chordal graphs and distance hereditary graphs. It can also be seen as a natural generalization of block graphs (and hence trees). In this paper, we first state a laminar structure of cliques, which leads to its canonical tree representation. This result is a translation of γ-acyclicity which appears in the context of relational database schemes. The tree representation gives a simple intersection model of ptolemaic graphs, and it is constructed in linear time from a perfect elimination ordering obtained by the lexicographic breadth first search. Hence the recognition and the graph isomorphism for ptolemaic graphs can be solved in linear time. Using the tree representation, we also give an efficient algorithm for the Hamiltonian cycle problem.

Key words: Algorithmic graph theory, data structures, γ-acyclicity, Hamiltonian cycle, intersection model, ptolemaic graphs.

1 Introduction

From an algorithmic point of view, a variety of graph classes have been proposed and studied so far [4,17]. Among them, the class of chordal graphs is

⋆ Preliminary version was presented at the 16th Annual International Symposium on Algorithms and Computation (ISAAC 2005)[32].

Email addresses: uehara@jaist.ac.jp (Ryuhei Uehara), uno@mi.s.osakafu-u.ac.jp (Yushi Uno).

classic and widely investigated. One of the reasons is that the class has a natural intersection model and hence has a concise tree representation; that is, a graph is chordal if and only if it is the intersection graph of subtrees of a tree. The tree representation can be constructed in linear time, and the tree is called a clique tree since each node of the tree corresponds to a maximal clique of the chordal graph (see [29]). Another reason is that the class is characterized by a vertex ordering, which is called a perfect elimination ordering. The ordering can also be computed in linear time, and a typical way to find it is called the lexicographic breadth first search (LBFS) introduced by Rose, Tarjan, and Lueker [28]. The LBFS is also intensively investigated as a tool for recognizing several graph classes (see a comprehensive survey by Corneil [9]). Using those characterizations, many efficient algorithms have been established for chordal graphs; to list a few of them, the maximum weighted clique problem, the maximum weighted independent set problem, the minimum coloring problem [16], the minimum maximal independent set problem [15], and so on. There are also parallel algorithms to solve some of these problems efficiently [24].

In algorithmic graph theory, distance in graphs is one of the most important topics. The class of distance hereditary graphs was characterized by Howorka to deal with the distance property called isometric [19]. Some characterizations of distance hereditary graphs are also given by Bandelt and Mulder [1], D'Atri and Moscarini [12], and Hammer and Maffray [18]. Especially, Bandelt and Mulder showed that any distance hereditary graph can be obtained from K_2 by a sequence of extensions called "adding a pendant vertex" and "splitting a vertex." Using these characterizations, many efficient algorithms have been found for distance hereditary graphs [3,6–8,22,23,27]. Among them, a few linear time algorithms are known for the recognition of a distance hereditary graph, however, they are too complicated and far from practical; Hammer and Maffray's algorithm [18] fails in some cases, and Damiand, Habib, and Paul's algorithm [11] requires to build a cotree in linear time (see [11, Chapter 4] for further details), where the cotree can be constructed in linear time by using a classic algorithm due to Corneil, Perl, and Stewart [10], or a recent algorithm based on multisweep LBFS approach by Bretscher, Corneil, Habib, and Paul [5]. Recently, Nakano, Uehara, and Uno also give another linear time algorithm for the recognition of distance hereditary graphs, which maintains two neighbor sets by two prefix trees [23]. However, their algorithm requires delicate implementation to achieve linear time.

In this paper, we focus on the class of ptolemaic graphs. Ptolemaic graphs are graphs that satisfy the Ptolemaic inequality $d(x,y)d(z,w) \leq d(x,z)d(y,w) + d(x,w)d(y,z)$ for any four vertices x, y, z, w. (The inequality is also known as "Ptolemy" inequality which seems to be more popular. However we use "Ptolemaic," which is stated by Howorka [20].) Howorka showed that the class of ptolemaic graphs coincides with the intersection of the class of chordal graphs

and the class of distance hereditary graphs [20]. Hence the results for chordal graphs and distance hereditary graphs can be applied to ptolemaic graphs. On the other hand, the class of ptolemaic graphs is a natural generalization of block graphs, and hence trees (see [33] for the relationships between related graph classes). However, there are relatively few known results specified to ptolemaic graphs.

We first state a tree representation of ptolemaic graphs which is based on the laminar structure of cliques of a ptolemaic graph. We here note that this result itself is not necessarily new. In [13], D'Atri and Moscarini showed that chordal graphs and its subclasses are strongly related to acyclicity concepts for hypergraphs which appeared in relational database theory. Especially, a ptolemaic graph corresponds to a γ-acyclic hypergraph, which is introduced by Fagin in [14]. The γ-acyclicity of hypergraph directly corresponds to the tree structure in our paper. The tree representation also gives a natural intersection model for ptolemaic graphs, which is defined over directed trees. We show a linear time algorithm that constructs the tree representation for a ptolemaic graph. The construction algorithm can also be modified to a recognition algorithm which runs in linear time. It is worth remarking that the algorithm is quite simple, especially, much simpler than the combination of two recognition algorithms for chordal graphs and distance hereditary graphs. In the tree construction and the recognition, the ordering of the vertices produced by the LBFS plays an important role. Therefore, our result adds the class of ptolemaic graphs to the list of graph classes that can be recognized efficiently by the LBFS. Moreover, the tree representation is canonical up to isomorphism. Hence, using the tree representation, we can solve the graph isomorphism problem for ptolemaic graphs in $O(\,|V|\,)$ time if a ptolemaic graph is given in the tree representation. We note that a clique tree of a chordal graph is not canonical and the graph isomorphism problem for chordal graphs is graph isomorphism complete.

The tree representation enables us to use the dynamic programming technique for some problems on ptolemaic graphs $G = (V, E)$. It is sure that the Hamiltonian cycle problem is one of most well known NP-hard problem, and it is still NP-hard even for a chordal graph, and that an $O(\,|V| + |E|\,)$ time algorithm is known for distance hereditary graphs [21,22]. In this paper, we show that the Hamiltonian cycle problem can be solved in $O(\,|V|\,)$ time by using the technique if a ptolemaic graph is given in the tree representation.

As we mentioned, the tree structure of a ptolemaic graph is known as γ-acyclicity for corresponding hypergraph in relational database theory, and it produces loop-free Bachman diagram (see [2,14] for more details). In a γ-acyclic hypergraph, hyperedges correspond to maximal cliques in ptolemaic graph. In [14], Fagin mentioned that the hypergraph can be recognized in polynomial time. Hence, we can say that our algorithm can also be used as a

more efficient substitution to recognize the hypergraphs in relational database theory.

2 Preliminaries

The *neighborhood* of a vertex v in a graph $G = (V, E)$ is the set $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$, and the *degree* of a vertex v is $|N_G(v)|$ and is denoted by $\deg_G(v)$. For a subset U of V, we denote by $N_G(U)$ the set $\{v \in V \mid v \in N(u) \text{ for some } u \in U\}$. If no confusion can arise we will omit the index G. Given a graph $G = (V, E)$ and a subset U of V, the *induced subgraph* by U, denoted by $G[U]$, is the graph (U, E'), where $E' = \{\{u, v\} \mid u, v \in U \text{ and } \{u, v\} \in E\}$. Given a graph $G = (V, E)$, its *complement*, denoted by $\bar{G} = (V, \bar{E})$, is defined by $\bar{E} = \{\{u, v\} \mid \{u, v\} \notin E\}$. A vertex set I is an *independent set* if $G[I]$ contains no edges, and then the graph $\bar{G}[I]$ is said to be a *clique*.

Given a graph $G = (V, E)$, a sequence of the distinct vertices $v_1, v_2, \ldots, v_l$ is a *path*, denoted by $(v_1, v_2, \ldots, v_l)$, if $\{v_j, v_{j+1}\} \in E$ for each $1 \leq j < l$. The *length* of a path is the number of edges in the path. For two vertices u and v, the *distance* of the vertices, denoted by $d(u, v)$, is the minimum length of the paths joining u and v. A *cycle* is a path beginning and ending at the same vertex. A cycle is said to be *Hamiltonian* if it visits every vertex in a graph exactly once.

An edge which joins two vertices of a cycle but is not itself an edge of the cycle is a *chord* of that cycle. A graph is *chordal* if each cycle of length at least 4 has a chord. Given a graph $G = (V, E)$, a vertex $v \in V$ is *simplicial* in G if $G[N(v)]$ is a clique in G. An ordering $v_1, \ldots, v_n$ of the vertices of V is a *perfect elimination ordering* (PEO) of G if the vertex v_i is simplicial in $G[\{v_i, v_{i+1}, \ldots, v_n\}]$ for all $i = 1, \ldots, n$. Once a vertex ordering is fixed, we denote $N(v_j) \cap \{v_{i+1}, \ldots, v_n\}$ by $N_{>i}(v_j)$. We also use the notations "min" and "max" to denote the first and the last vertices in an ordered set of vertices, respectively. It is known that a graph is chordal if and only if it has a PEO (see [4, Section 1.2] for further details). A typical way of finding a perfect elimination ordering of a chordal graph in linear time is the lexicographic breadth first search (LBFS), which is introduced by Rose, Tarjan, and Lueker [28], and a comprehensive survey is presented by Corneil [9].

It is also known that a graph $G = (V, E)$ is chordal if and only if it is the intersection graph of subtrees of a tree T (see [4, Section 1.2] for further details). Let T_v denote the subtree of T corresponding to the vertex v in G. Then we can assume that each node c in T corresponds to a maximal clique C of G such that C contains v on G if and only if T_v contains c on T. Such

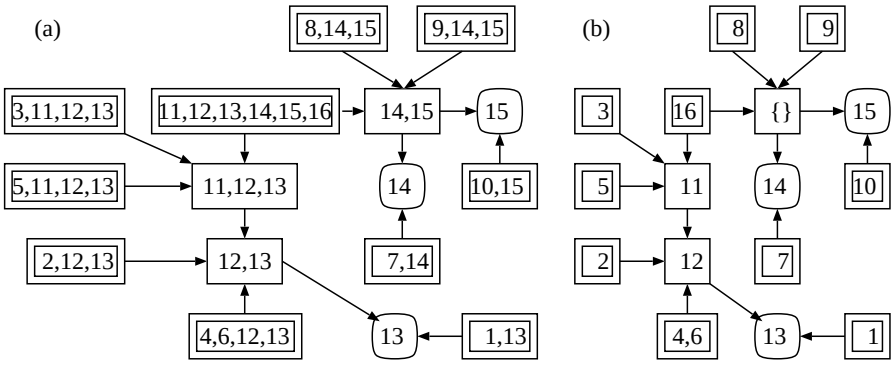


Fig. 1. (a) Laminar tree; (b) its labels

a tree T is called a *clique tree* of G. From a perfect elimination ordering of a chordal graph G, we can construct a clique tree of G in linear time [29]. We sometimes identify a node c of a clique tree T with a maximal clique (or a vertex set) C of G.

Given a graph $G = (V, E)$ and a subset U of V, an induced connected subgraph $G[U]$ is *isometric* if the distances in $G[U]$ are the same as in G. A graph G is *distance hereditary* if G is connected and every induced path in G is isometric.

A connected graph G is *ptolemaic* if for any four vertices u, v, w, x of G, $d(u, v)d(w, x) \leq d(u, w)d(v, x)+d(u, x)d(v, w)$. We will use the following characterization of ptolemaic graphs due to Howorka [20]:

Theorem 1 *The following conditions are equivalent: (1) G is ptolemaic; (2) G is distance hereditary and chordal; (3) for all distinct nondisjoint maximal cliques P and Q of G, $P \cap Q$ separates $P \setminus Q$ and $Q \setminus P$.*

Let V be a set of n vertices. Two sets X and Y are said to be *overlapping* if $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$, and $Y \setminus X \neq \emptyset$. A family $\mathcal{F} \subseteq 2^V \setminus \{\{\emptyset\}\}$ is said to be *laminar* if $\mathcal{F}$ contains no overlapping sets; that is, any pair of two distinct sets X and Y in $\mathcal{F}$ satisfies either $X \cap Y = \emptyset$, $X \subset Y$, or $Y \subset X$. Given a laminar family $\mathcal{F}$, we define *laminar digraph* $\overrightarrow{T}(\mathcal{F}) = (\mathcal{F}, \overrightarrow{E}_{\mathcal{F}})$ as follows; $\overrightarrow{E}_{\mathcal{F}}$ contains an arc (Y, X) if and only if $X \subset Y$ and there are no other subset Z of $\mathcal{F}$ such that $X \subset Z \subset Y$, for any sets X and Y. In this case, Y is said to be *parent* of X, and X is said to be *child* of Y. We denote the underlying graph of $\overrightarrow{T}(\mathcal{F})$ by $T(\mathcal{F}) = (\mathcal{F}, E_{\mathcal{F}})$. The following lemma for the laminar digraph is known (see, e.g., [26, Chapter 2.2]);

Lemma 2 *If a family $\mathcal{F} \subseteq 2^V \setminus \{\{\emptyset\}\}$ is laminar, (1) $T(\mathcal{F})$ is a forest, and (2) $|\mathcal{F}| \leq 2|V| - 1$.*

Hence, hereafter, we call $T(\mathcal{F})$ ($\overrightarrow{T}(\mathcal{F})$) a (directed) laminar forest. On a directed laminar forest $\overrightarrow{T}(\mathcal{F})$, we abuse some notations for trees as follows; we call a node *leaf* if it has outdegree 0, and *root* if it has indegree 0. Hence, a

leaf can have two or more parents if it has indegree two or more. To have a compact representation, we define a *label* of each node S_0 in $\mathcal{F}$ of $\overrightarrow{T}(\mathcal{F})$, denoted by $\ell(S_0)$, as follows: If S_0 is a leaf, $\ell(S_0) = S_0$. If S_0 is not a leaf and has children $S_1, S_2, \ldots, S_h$, $\ell(S_0) = S_0 \setminus (S_1 \cup S_2 \cup \cdots \cup S_h)$. That is, each vertex v in V appears in $\ell(S)$ where S is the minimal set containing v. Since $\mathcal{F}$ is laminar, each vertex in V appears exactly once in $\ell(S)$ for some $S \subseteq V$, and its corresponding node is uniquely determined. An example is shown in Fig. 1. In the figure, each double rectangle represents a root, and each rounded rectangle represents a leaf. In a directed laminar tree in Fig. 1(a), we remove redundant items, which can be found in its descendants, and obtain the compact tree in Fig. 1(b) that consists of nodes with its labels. We note that an internal node in $\overrightarrow{T}(\mathcal{F})$ has a label $\emptyset$ when it is partitioned completely by its subsets in $\mathcal{F}$ (the node corresponding to $\{14, 15\}$ has empty label in Fig. 1(b)).

3 A Tree Representation of Ptolemaic Graphs

In this section, we show that ptolemaic graphs have a canonical tree representation, and it can be constructed in linear time. Hereafter, we assume that $G = (V, E)$ is a given (ptolemaic) graph with $n = |V|$ vertices and $m = |E|$ edges.

3.1 A Tree Representation

For a ptolemaic graph $G = (V, E)$, let $\mathcal{M}(G)$ be the set of all maximal cliques, i.e.,

$$\mathcal{M}(G) := \{M \mid M \text{ is a maximal clique in } G\},$$

and $\mathcal{C}(G)$ be the set of nonempty vertex sets defined below:

$$\mathcal{C}(G) := \bigcup_{S \subseteq \mathcal{M}(G)} \{C \mid C = \cap_{M \in S} M, C \neq \emptyset\}.$$

Each vertex set $C \in \mathcal{C}(G)$ is a nonempty intersection of some maximal cliques. Hence, $\mathcal{C}(G)$ contains all maximal cliques, and each C in $\mathcal{C}(G)$ induces a clique. We also denote by $\mathcal{L}(G)$ the set $\mathcal{C}(G) \setminus \mathcal{M}(G)$. That is, each vertex set $L \in \mathcal{L}(G)$ is an intersection of two or more maximal cliques, and hence L is a non-maximal clique.

We note that the set $\mathcal{M}(G)$ of maximal cliques corresponds to the set of hyperedges in the hypergraph model discussed in [13,14]. Theorem 3, Lemma

4, and Theorem 5 are not necessarily new, which could be found in [14] in the context of database. However, we state here since we would like to make this paper be self-contained, and it is worth proving in the context of graph theory.

Theorem 3 *Let $G = (V, E)$ be a ptolemaic graph. Let $\mathcal{F}$ be a family of sets in $\mathcal{L}(G)$ such that $\cup_{L \in \mathcal{F}} L \subset M$ for some maximal clique $M \in \mathcal{M}(G)$. Then $\mathcal{F}$ is laminar.*

PROOF. To derive a contradiction, we assume that $\mathcal{F}$ is not laminar. Then we have two overlapping vertex sets L_1 and L_2 which are properly contained in the maximal clique M. Let v, v_1, v_2 be vertices in $L_1 \cap L_2$, $L_1 \setminus L_2$, and $L_2 \setminus L_1$, respectively. By definition, there are sets of maximal cliques $M_1^1, M_1^2, \ldots, M_1^a$, $M_2^1, M_2^2, \ldots, M_2^b$ such that $L_1 = M_1^1 \cap M_1^2 \cap \cdots \cap M_1^a$ and $L_2 = M_2^1 \cap M_2^2 \cap \cdots \cap M_2^b$. Here, if every M_1^i with $1 \leq i \leq a$ contains v_2, we have $v_2 \in L_1$. Thus, there is a maximal clique M_1^i with $v_2 \notin M_1^i$. Similarly, there is a maximal clique M_2^j with $v_1 \notin M_2^j$. Let L be $M_1^i \cap M_2^j$. Then we have $v_1, v_2 \notin L$ and $v \in L$ (hence $L \neq \emptyset$). Therefore $v_1 \in M_1^i \setminus L$ and $v_2 \in M_2^j \setminus L$. Moreover, since v_1, v_2 are in M, $\{v_1, v_2\} \in E$. Thus, $L = M_1^i \cap M_2^j$ does not separate $M_1^i \setminus L$ and $M_2^j \setminus L$, which contradicts Theorem 1(3). $\square$

Lemma 4 *Let C_1, C_2 be any overlapping sets in $\mathcal{C}(G)$ for a ptolemaic graph $G = (V, E)$. Then $C_1 \cap C_2$ separates $C_1 \setminus C_2$ and $C_2 \setminus C_1$.*

PROOF. Let $C := C_1 \cap C_2$. By definition of $\mathcal{C}(G)$, $C \in \mathcal{C}(G)$. Let C_i' be the sets in $\mathcal{C}(G)$ such that $C \subset C_i' \subset C_i$ and there is no other C' with $C \subset C' \subset C_i'$ for $i = 1, 2$. We first observe that C_1' and C_2' are overlapping: If $C_1' = C_2'$, we have $C_1' = C_2' \subseteq C_1 \cap C_2$ which contradicts that $C = C_1 \cap C_2$ and $C \subset C_1'$. On the other hand, if $C_1' \subset C_2'$, we have $C \subset C_1' \subset C_2'$ which contradicts the definition of C_2'.

We show that C separates C_1' and C_2'. Let M_i be maximal cliques that contains C_i' for $i = 1, 2$ such that M_1 is overlapping to C_2' and M_2 is overlapping to C_1. Let $C_c := M_1 \cap M_2$. By definition, $C \subseteq C_c$. It is sufficient to show that $C = C_c$. To derive contradictions, we assume that $v \in C_c \setminus C$. We first assume that $v \in C_1' \setminus C_2'$. In the case, since M_2 and C_1' are overlapping, $\mathcal{C}$ contains a set C'' with $(C_1' \cap C_2') \subset ((C_1' \cap C_2') \cup \{v\}) \subseteq C'' \subset C_1'$, which is a contradiction. Thus, we have $v \notin C_1'$ and $v \notin C_2'$.

By definition of C_1', there are maximal cliques $M_1^1, M_1^2, \ldots, M_1^k$ such that $C_1' = \cap_{i=1}^k M_1^i$. Since $v \notin C_1'$, there is at least one maximal clique M_1^i with $v \notin M_1^i$. Similarly, there is at least one maximal clique M_2^j with $C_2' \subseteq M_2^j$ and $v \notin M_2^j$. However, $C_1' \subseteq M_1^i$, $C_2' \subseteq M_2^j$, and $v \notin M_1^i \cup M_2^j$ imply that $M_1^i \setminus M_2^j$ and

$M_2^j \setminus M_1^i$ are connected by v. This is a contradiction to Theorem 1(3). Hence $M_1^i \cap M_2^j = M_1 \cap M_2 = C_1' \cap C_2' = C_1 \cap C_2$, and it is a separator. □

Now we define a directed graph $\overrightarrow{T}(\mathcal{C}(G)) = (\mathcal{C}(G), A(G))$ for a given ptolemaic graph $G = (V, E)$ as follows: two nodes $C_1, C_2 \in \mathcal{C}(G)$ are joined by an arc (C_1, C_2) if and only if $C_2 \subset C_1$ and there is no other C in $\mathcal{C}(G)$ such that $C_2 \subset C \subset C_1$. We denote by $T(\mathcal{C}(G))$ the underlying graph of $\overrightarrow{T}(\mathcal{C}(G))$.

Theorem 5 *A graph G is ptolemaic if and only if the graph $T(\mathcal{C}(G))$ is a tree.*

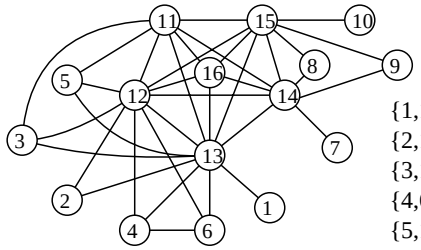
PROOF. We first assume that G is ptolemaic and show that $T(\mathcal{C}(G))$ is a tree. It is clear that $T(\mathcal{C}(G))$ is connected. Thus, to derive contradictions, we assume that $T(\mathcal{C}(G))$ contains a cycle $(C_1, C_2, \ldots, C_k, C_1)$, which is a minimal cycle without chords on $T(\mathcal{C}(G))$. Since $C_k \subset C_{k-1} \subset \cdots \subset C_1 \subset C_k$ (or vice versa) is impossible, there is a node C_a with $C_{a-1} \supset C_a \subset C_{a+1}$ for some a. Without loss of generality, we assume that $|C_a|$ is the smallest among such vertex sets on the cycle. Let C_x and C_y be the nodes on the cycle such that $C_{x-1} \subset C_x \supset C_{x+1} \supset \cdots \supset C_{a-1} \supset C_a \subset C_{a+1} \subset \cdots \subset C_{y-1} \subset C_y \supset C_{y+1}$. It is not difficult to see that C_{a-1} and C_{a+1}, and hence C_x and C_y are overlapping. Thus, by Lemma 4, C_a separates $C_x \setminus C_y$ and $C_y \setminus C_x$. Since C_a is a separator, we let G_x and G_y be the connected components that contain $C_x \setminus C_y$ and $C_y \setminus C_x$ on $G[V \setminus C_a]$, respectively.

Now we consider the path $\mathsf{P} = (C_x, C_{x-1}, C_{x-2}, \ldots, C_{y+2}, C_{y+1}, C_y)$ which does not contain C_a. However, since C_a is a separator, P contains at least one vertex set C_b in $\mathcal{C}$ with $C_a \cap C_b \neq \emptyset$. If $(C_x \cap C_b) \setminus C_a \neq \emptyset$ and $(C_y \cap C_b) \setminus C_a \neq \emptyset$, $C_x \setminus C_y$ and $C_y \setminus C_x$ are connected on $G[V \setminus C_a]$ since C_b is a clique. Hence each C_b with $C_a \cap C_b \neq \emptyset$ satisfies $(C_x \cap C_b) \setminus C_a = \emptyset$ or $(C_y \cap C_b) \setminus C_a = \emptyset$. Since P connects G_x and G_y through the separator C_a, we have at least two vertex sets C_b and C_b' such that $(C_y \cap C_b) \setminus C_a = \emptyset$ and $(C_x \cap C_b') \setminus C_a = \emptyset$. Moreover, since C_a separates G_x and G_y, we have $C_b \cap C_b' \subseteq C_a$. If $C_b \cap C_b' \subset C_a$, P contains smaller separator than C_a. Thus $C_b \cap C_b' = C_a$. Then P has to contain C_a between C_b and C_b', which contradicts the minimality of the cycle.

Therefore, $T(\mathcal{C}(G))$ is a tree.

It is easy to see that G is ptolemaic if $T(\mathcal{C}(G))$ is a tree; for each pair of distinct nondisjoint maximal cliques M_1 and M_2, $(M_1 \cap M_2)$ separates $T(\mathcal{C}(G))$, and hence G. □

Hereafter, for a given ptolemaic graph $G = (V, E)$, we call $T(\mathcal{C}(G))$ ($\overrightarrow{T}(\mathcal{C}(G))$) a *(directed) clique laminar tree* of G. We note that for each vertex in G its



{1,13} {7,14}
{2,12,13} {8,14,15}
{3,11,12,13} {9,14,15}
{4,6,12,13} {10,15}
{5,11,12,13} {11,12,13,14,15,16}

Fig. 2. A ptolemaic graph G which produce the directed laminar tree in Fig. 1

corresponding node in $T(\mathcal{C}(G))$ is uniquely determined by maximal cliques. Therefore, we can define a mapping from each vertex to the vertex set in $\mathcal{C}$ in $T(\mathcal{C}(G))$: We denote by $C(v)$ the clique C with $v \in \ell(C)$. When we know whether $C(v)$ is in $\mathcal{M}$ or $\mathcal{L}$, we specify it by writing $C_M(v)$ or $C_L(v)$. An example is given in Fig. 2 (all maximal cliques are also depicted); for the graph G in the figure, $\overrightarrow{T}(\mathcal{C}(G))$ is given in Fig. 1. We also note that from $\overrightarrow{T}(\mathcal{C}(G))$ with labels, we can reconstruct the original ptolemaic graph uniquely up to isomorphism. That is, two ptolemaic graphs G_1 and G_2 are isomorphic if and only if labeled $\overrightarrow{T}(\mathcal{C}(G_1))$ is isomorphic to labeled $\overrightarrow{T}(\mathcal{C}(G_2))$.

Intuitively, a clique laminar tree subdivides a clique tree of a chordal graph. For a chordal graph, maximal cliques are joined in a looser way in the sense that a clique tree for a chordal graph is not always uniquely determined up to isomorphism. The clique laminar tree subdivides the relationships between maximal cliques by using their laminar structure.

We can easily see the following properties of $\overrightarrow{T}(\mathcal{C}(G))$, and they are useful from the algorithmic point of view:

Corollary 6 *If G is a ptolemaic graph, we have the following: (1) For each maximal clique M in $\mathcal{M}(G)$, $\ell(M)$ consists of simplicial vertices in M. (2) The vertices in a maximal clique M in $\mathcal{M}(G)$ induce a maximal directed subtree of $\overrightarrow{T}(\mathcal{C}(G))$ rooted at the node M. (3) Each leaf in $T(\mathcal{C}(G))$ corresponds to a maximal clique in $\mathcal{M}(G)$.*

It is well known that a graph is chordal if and only if it is the intersection graph of subtrees of a tree. By Theorem 5, we obtain an intersection model for ptolemaic graphs as follows:

Corollary 7 *Let $\overrightarrow{T}$ be any directed graph such that its underlying graph T is a tree. Let $\mathcal{T}$ be any set of subtrees $\overrightarrow{T_v}$ such that $\overrightarrow{T_v}$ consists of a node C and all vertices reachable to C in $\overrightarrow{T}$. Then the intersection graph over $\mathcal{T}$ is ptolemaic. On the other hand, for any ptolemaic graph, there exists such an intersection model.*

PROOF. The directed clique laminar tree $\overrightarrow{T}(\mathcal{C}(G))$ is the base directed graph of the intersection model. For each $v \in V$, we define the node C such that $v \in \ell(C)$. By definition, a clique C' contains v if and only if there is a directed path from the corresponding node C' to the node C on $\overrightarrow{T}(\mathcal{C}(G))$. □

3.2 A Linear Time Construction of Clique Laminar Trees

The main theorem in this section is the following:

Theorem 8 *Given a ptolemaic graph $G = (V, E)$, the directed clique laminar tree $\overrightarrow{T}(\mathcal{C}(G))$ can be constructed in $O(n + m)$ time.*

We will make the directed clique laminar tree $\overrightarrow{T}(\mathcal{C}(G))$ by separating the vertices in G into the vertex sets in $\mathcal{C}(G) = \mathcal{M}(G) \cup \mathcal{L}(G)$.

We first compute (and fix) a perfect elimination ordering $v_1, v_2, \ldots, v_n$ by the LBFS. The outline of our algorithm is similar to the algorithm for constructing a clique tree for a given chordal graph due to Spinrad in [29]. For each vertex $v_n, v_{n-1}, \ldots, v_2, v_1$, we add it into the tree and update the tree. For the current vertex v_i, let $v_j := \min\{N_{>i}(v_i)\}$. Then, in Spinrad's algorithm [29], there are two cases to be considered: $N_{>i}(v_i) = C(v_j)$ or $N_{>i}(v_i) \subset C(v_j)$. The first case is simple; just add v_i into $C(v_j)$. In the second case, Spinrad's algorithm adds a new maximal clique $C(v_i)$ that consists of $N_{>i}(v_i) \cup \{v_i\}$. However, in our algorithm, involved case analysis is required. For example, in the latter case, the algorithm has to handle three vertex sets; two maximal cliques $\{v_i\} \cup N_{>i}(v_i)$ and $C(v_j)$ together with one vertex set $N_{>i}(v_i)$ shared by them. In this case, intuitively, our algorithm makes three distinct sets C_M with $\ell(C_M) = \{v_i\}$, C_L with $\ell(C_L) = N_{>i}(v_i)$, and C with $\ell(C) = C(v_j) \setminus N_{>i}(v_i)$, and adds two arcs (C_M, C_L) and (C, C_L); this means that v_i is in $C_M = N_{>i}(v_i) \cup \{v_i\}$, C is a clique $C(v_j)$, and C_L is the vertex set shared by C_M and C. However, our algorithm has to handle more complicated cases since the set $C(v_j)$ (and hence $N_{>i}(v_i)$) can already be partitioned into some vertex sets.

In $\overrightarrow{T}(\mathcal{C}(G))$, each node C stores its label $\ell(C)$. Hence each vertex in G appears exactly once in the tree. To represent it, each vertex v has a pointer to the node $C(v)$ in $\mathcal{C}(G) = \mathcal{M}(G) \cup \mathcal{L}(G)$. The detail of the algorithm is described as CliqueLaminarTree shown in Fig. 3, and an example of the construction is depicted in Fig. 4. In Fig. 4, the left-hand graph gives a ptolemaic graph (as same as Fig. 2), and the right-hand trees are clique laminar trees constructed (a) after adding the vertices $16, 15, 14, 13, 12, 11$, (b) after adding the vertices $16, 15, 14, 13, 12, 11, 10$, (c) after adding the vertices $16, 15, 14, 13, 12, 11, 10, 9, 8$, and (d) after adding all the vertices. We show the correctness and a complexity analysis of the algorithm.

Fig. 3. Algorithm: CliqueLaminarTree

Input: A ptolemaic graph $G = (V, E)$ with a PEO $v_1, v_2, \ldots, v_n$ by the LBFS.
Output: A clique laminar tree T.

initialize T by the clique $C_M(v_n) := \{v_n\}$ and set the pointer from v_n to $C_M(v_n)$;
for $i := n - 1$ down to 1 **do**
  let $v_j := \min\{N_{>i}(v_i)\}$;
  switch condition of $N_{>i}(v_i)$ **do**
    case (1) $N_{>i}(v_i) = C_M(v_j)$
      update $\ell(C_M(v_j)) := \ell(C_M(v_j)) \cup \{v_i\}$ and $|C_M(v_j)| := |C_M(v_j)| + 1$;
      set $C_M(v_i) := C_M(v_j)$;
    case (2) $N_{>i}(v_i) = C_L(v_j)$
      make a new maximal clique $C_M(v_i)$ with $\ell(C_M(v_i)) := \{v_i\}$
        and $|C_M(v_i)| := |C_L(v_j)| + 1$;
      add an arc $(C_M(v_i), C_L(v_j))$;
    case (3) $N_{>i}(v_i) \subset C(v_j)$ and $|\ell(C(v_j))| = |C(v_j)|$
      update $\ell(C(v_j)) := \ell(C(v_j)) \setminus N_{>i}(v_i)$
        and $|\ell(C(v_j))| := |\ell(C(v_j))| - |N_{>i}(v_i)|$;
      make a new vertex set $L := N_{>i}(v_i)$ with $\ell(L) := N_{>i}(v_i)$
        and $|L| := |N_{>i}(v_i)|$;
      make a new maximal clique $C_M(v_i)$ with $\ell(C_M(v_i)) = \{v_i\}$
        and $|C_M(v_i)| := |L| + 1$;
      add arcs $(C(v_j), L)$ and $(C_M(v_i), L)$;
    case (4) $N_{>i}(v_i) \subset C(v_j)$ and $|\ell(C(v_j))| < |C(v_j)|$
      make a new vertex set $L := N_{>i}(v_i)$ with $\ell(L) := N_{>i}(v_i) \cap \ell(C(v_j))$
        and $|L| := |N_{>i}(v_i)|$;
      update $\ell(C(v_j)) := \ell(C(v_j)) \setminus L$ and $|\ell(C(v_j))| := |\ell(C(v_j))| - |L|$;
      make a new maximal clique $C_M(v_i)$ with $\ell(C_M(v_i)) = \{v_i\}$
        and $|C_M(v_i)| = |L| + 1$;
      remove the arc $(C(v_j), L')$ with $L' \subset L$ and add an arc (L, L');
      add arcs $(C(v_j), L)$ and $(C_M(v_i), L)$;
  set the pointer from v_i to $C(v_i)$;
return T.

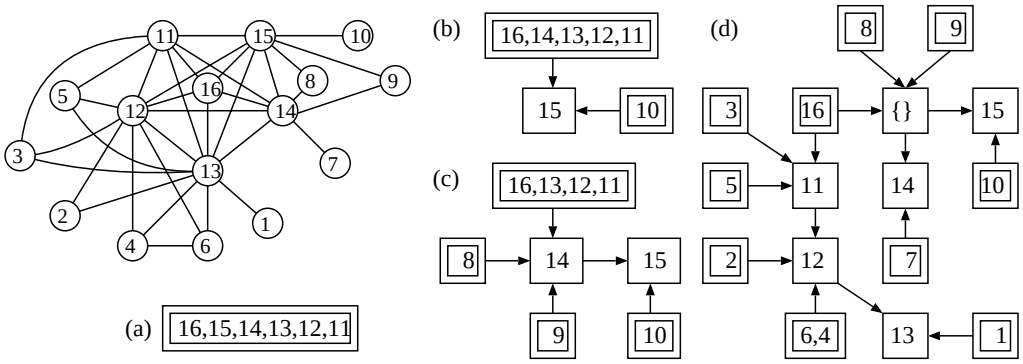


Fig. 4. A ptolemaic graph and its clique laminar tree.

We will use the following property of a PEO found by the LBFS of a chordal graph:

Lemma 9 ([9, Theorem 1]) *Let $v_1, v_2, \ldots, v_n$ be a PEO found by the LBFS. Then $i < j$ implies $\max\{N(v_i)\} \leq \max\{N(v_j)\}$.*

This property is one of basic properties of LBFS, and hence the proof is omitted here. See Theorem 1 in [9], for example.

We assume that Algorithm CliqueLaminarTree is going to add v_i, and let $v_j := \min\{N_{>i}(v_i)\}$. We will show that all possible cases are listed, and in each case, CliqueLaminarTree correctly manages the nodes in $\mathcal{C}(G)$ and their labels in $O(\deg(v_i))$ time. The following lemma drastically decreases the number of possible cases and simplifies the algorithm.

Lemma 10 *Let v_k be $\max\{N_{>i}(v_i)\}$. In addition, assume that the set $N_{>i}(v_i)$ has already been divided into some distinct vertex sets $L_1, L_2, \ldots, L_h$. Then, there is an ordering of the sets such that $v_k \in L_1 \subset L_2 \subset \cdots \subset L_h$.*

PROOF. We first observe that $G[\{v_i, v_{i+1}, \ldots, v_n\}]$ is ptolemaic if G is ptolemaic since any vertex induced subgraph of a chordal graph is chordal, and any vertex induced subgraph of a distance hereditary graph is distance hereditary.

We assume that there is a vertex set $L \subset N_{>i}(v_i)$ such that L does not contain v_k. Then, there is a vertex $v_{i'}$ with $i' > i$ that makes the vertex set L before v_i. Since $\{v_{i'}, v_k\} \notin E$, by Lemma 9, $v_{i'}$ has another neighbor $v_{k'}$ with $k' > k$. By the property of the LBFS, it is easy to see that $G[\{v_k, \ldots, v_n\}]$ is connected. Let M_i be a maximal clique $\{v_i\} \cup N_{>i}(v_i)$, and $M_{i'}$ be a maximal clique that contains $\{v_{i'}\} \cup L$. Then, $M_i \cap M_{i'} = L$ which contains no vertex in $G[\{v_k, \ldots, v_n\}]$. On the other hand, we have $\{v_i, v_k\}, \{v_{i'}, v_{k'}\} \in E$. Hence, $M_i \cap M_{i'}$ does not separate $M_i \setminus M_{i'}$ and $M_{i'} \setminus M_i$. Therefore $G[\{v_i, v_{i+1}, \ldots, v_n\}]$ is not ptolemaic by Theorem 1(3), which is a contradiction. Thus we have $v_k \in L$, and hence, all the vertex sets $L_1, L_2, \ldots, L_h$ contain v_k. The vertex set $N_{>i}(v_i)$ is contained in a maximal clique in the ptolemaic graph $G[\{v_i, v_{i+1}, \ldots, v_n\}]$. Hence by Theorem 3, $L_1, L_2, \ldots, L_h$ are laminar. Therefore, we have $v_k \in L_1 \subset L_2 \subset \cdots \subset L_h$ for some appropriate ordering. □

Proof of Theorem 8. Since the graph G is chordal and the vertices are ordered in a perfect elimination ordering, $N_{>i}(v_i)$ induces a clique. By Lemma 10, we have three possible cases; (a) $N_{>i}(v_i) = C(v_j)$, (b) $N_{>i}(v_i) \subset C(v_j)$ and there are no vertex sets in $N_{>i}(v_i)$, and (c) $N_{>i}(v_i) \subset C(v_j)$ and there are vertex sets $L_1 \subset L_2 \subset \cdots \subset L_h \subset N_{>i}(v_i)$. In the last case, we note that $L_h \neq N_{>i}(v_i)$; otherwise, we have $v_j \in L_h$, or consequently, $L_h = C(v_j) = N_{>i}(v_i)$, which is case (a).

(a) $N_{>i}(v_i) = C(v_j)$**:** We have two subcases; $C(v_j)$ is a maximal clique (i.e. $N_{>i}(v_i) = C_M(v_j)$) or $C(v_j)$ is a non-maximal clique (i.e. $N_{>i}(v_i) = C_L(v_j)$). In the former case, we just update $C_M(v_j)$ by $C_M(v_j) \cup \{v_i\}$. This is case (1) in CliqueLaminarTree. In the latter case, there is another vertex set that contains $C_L(v_j)$ as a subset. Thus we add a new maximal clique $C_L(v_j) \cup \{v_i\}$. More precisely, we add a new node $C_M(v_i)$ with $\ell(C_M(v_i)) = \{v_i\}$ and $|C_M(v_i)| = |C_L(v_j)| + 1$, and a new arc $(C_M(v_i), C_L(v_j))$. This is done in case (2) of CliqueLaminarTree. We can check if $N_{>i}(v_i) = C(v_j)$ by checking if $|N_{>i}(v_i)| = |C(v_j)|$ in $O(1)$ time. Thus the time complexity is $O(1)$ in both cases.

(b) $N_{>i}(v_i) \subset C(v_j)$ **and there are no vertex sets in** $N_{>i}(v_i)$**:** We remove $N_{>i}(v_i)$ from $C(v_j)$ and make a new vertex set $N_{>i}(v_i)$ shared by $C(v_j)$ and $C_M(v_i) = \{v_i\} \cup N_{>i}(v_j)$. We can observe that $N_{>i}(v_i) \subset C(v_j)$ and there are no vertex sets in $N_{>i}(v_i)$ if and only if $|N_{>i}(v_i)| < |C(v_j)|$ and $|\ell(C(v_j))| = |C(v_j)|$. Thus, CliqueLaminarTree recognizes this case in $O(1)$ time, and handles it in case (3). It is easy to see that case (3) can be done in $O(\,|N_{>i}(v_i)|\,) = O(\deg(v_i))$ time. We note that, in this case, we do not mind if $C(v_j)$ is maximal or not. In any case, the property does not change for $C(v_j)$.

(c) $N_{>i}(v_i) \subset C(v_j)$ **and there are vertex sets** $L_1 \subset L_2 \subset \cdots \subset L_h \subset N_{>i}(v_i)$**:** We first observe that the nodes $L_h, L_{h-1}, \ldots, L_2, L_1$ with $C(v_j)$ form a directed path $(C(v_j), L_h, \ldots, L_1)$ in $\overrightarrow{T}$ in the case. (Hence we can recognize this case in $O(\,|N_{>i}(v_i)|\,) = O(\deg(v_i))$ time, which will be used in Theorem 11.) Thus we make a new vertex set $L := N_{>i}(v_i)$ with $\ell(L) = N_{>i}(v_i) \setminus L_h$. The set $N_{>i}(v_i) \setminus L_h$ is given by $N_{>i}(v_i) \cap \ell(C(v_j))$. Then we update $\ell(C(v_j))$ by $\ell(C(v_j)) \setminus N_{>i}(v_i)$. It is easy to add a maximal clique $C_M(v_i) = \{v_i\} \cup N_{>i}(v_i)$. Next, we have to update arcs around $C(v_j)$. By Lemma 10, this process is simple; we can find L_h in $O(\deg(v_i))$ time, and there is no other vertex set L' that has an arc $(C(v_j), L')$ which has to be updated. We note that there can be some vertex set L' with an arc $(C(v_j), L')$. But L' is independent from L in this case, and hence we do not have to mind it. Finally, we change the arc $(C(v_j), L_h)$ to (L, L_h), and add the arcs $(C(v_j), L)$ and $(C_M(v_i), L)$. Therefore the time complexity in the last case is $O(\deg(v_i))$ time.

By the above case analysis, Theorem 8 is settled. □

4 Applications of Clique Laminar Trees

4.1 The Recognition Problem

Theorem 11 *The recognition problem for ptolemaic graphs can be solved in linear time.*

PROOF. Using the LBFS, we can obtain a perfect elimination ordering of G in linear time if G is chordal (and reject it if G is not chordal). For a chordal graph, we run modified CliqueLaminarTree. It is not difficult to modify CliqueLaminarTree to reject it if G is not distance hereditary. The key fact is that, if G is ptolemaic, $N_{>i}(v_i)$ corresponds to a maximal directed path in $\overrightarrow{T}(\mathcal{C}(G))$ as follows; suppose that we have vertex sets $L_1 \subset L_2 \subset \cdots \subset L_h \subset N_{>i}(v_i) \subset C(v_j)$ in case (c) in the proof of Theorem 8. In this case, (1) the nodes form a connected directed path $(C(v_j), L_h, \ldots, L_2, L_1)$ in $\overrightarrow{T}(\mathcal{C}(G))$, (2) there are no other set L with $L \subset L_1$, (3) all vertices in L_h (and hence $L_1 \cup L_2 \cup \cdots \cup L_h$) belong to $N_{>i}(v_i)$, and (4) some vertices in $C(v_j)$ may not be in $N_{>i}(v_i)$. Checking them can be done in $O(\,|N_{>i}(v_i)|\,) = O(\deg(v_i))$ time for each i, and otherwise, the vertex sets in the tree are not laminar, and hence they would be rejected. Cases (a) and (b) can be seen as special cases of case (c). Therefore, the total running time of the modified CliqueLaminarTree is still $O(n+m)$. □

We note that the result in Theorem 11 is not necessarily new. Since a graph is ptolemaic if and only if it is chordal and distance-hereditary [20], distance hereditary graphs are recognized in linear time [5,10,11,18], and chordal graphs are also recognized in linear time [28,31], we have Theorem 11 by combining them. We dare to state Theorem 11 to show that we can recognize if a graph is ptolemaic and then construct its clique laminar tree at the same time in linear time, and the algorithm is much simpler and more straightforward than the combination of known algorithms. (As noted in Introduction, the linear time algorithm for recognition of distance hereditary graphs is not so simple.)

4.2 The Graph Isomorphism Problem

Theorem 12 *The graph isomorphism problem for ptolemaic graphs can be solved in linear time.*

PROOF. Given a ptolemaic graph $G = (V, E)$, the labeled clique laminar tree $\overrightarrow{T}(\mathcal{C}(G))$ is uniquely determined up to isomorphism: Maximal cliques are uniquely determined, and so are $\mathcal{M}(G)$ and $\mathcal{C}(G)$. By Theorem 3, they form a laminar structure, and hence $\overrightarrow{T}(\mathcal{C}(G))$ is the unique tree structure for given ptolemaic graph G by Lemma 2. Each vertex in V appears once in $\overrightarrow{T}(\mathcal{C}(G))$, and the number of nodes in $\overrightarrow{T}(\mathcal{C}(G))$ is at most $2|V| - 1$ by Lemma 2(2). Thus the representation of $\overrightarrow{T}(\mathcal{C}(G))$ requires $O(|V|)$ space. The graph isomorphism problem for labeled trees can be done in linear time (see, e.g., [25]), which completes the proof. □

It is worth mentioning that the graph isomorphism problem can be solved in $O(n)$ time if a ptolemaic graph is given in the tree representation.

4.3 The Hamiltonian Cycle Problem

We assume that a ptolemaic graph $G = (V, E)$ is given by a directed clique laminar tree $\overrightarrow{T}(\mathcal{C}(G)) = (\mathcal{C}(G), A(G))$. Then the main theorem in this section is the following:

Theorem 13 *The Hamiltonian cycle problem for ptolemaic graphs can be solved in $O(n)$ time.*

We remind that $\overrightarrow{T}(\mathcal{C}(G))$ takes $O(n)$ space. We then remind that each clique C in $\mathcal{C}$ is a separator of G; removing C makes G disconnected. Hence, if $\overrightarrow{T}(\mathcal{C}(G))$ contains a vertex set C with $|C| = 1$, G does not have a Hamiltonian cycle. This condition can be checked in $O(n)$ time over $\overrightarrow{T}(\mathcal{C}(G))$. Therefore, hereafter, we assume that any vertex set C in $\mathcal{C}$ satisfies $|C| > 1$.

Let L be a vertex set in $\mathcal{C}(G)$. We remind that the notions in ordinary trees are slightly abused on $\overrightarrow{T}(\mathcal{C}(G))$: A root has indegree 0, and a leaf has outdegree 0. That is, each maximal clique corresponds to a root. Note that a vertex set can have two or more parents when it is shared by some maximal cliques. We define *ancestors* and *descendants* as in ordinary trees. We regard any node L as an ancestor and descendant of itself. We denote by $c(L)$ and $p(L)$ the number of children of L and the number of parents of L in $\overrightarrow{T}(\mathcal{C}(G))$, respectively. Hence $p(M) = 0$ for each maximal clique M, and $c(L) = 0$ for each minimal vertex set L.

The basic idea to construct a Hamiltonian cycle is as follows. By Lemma 4, each clique L in $\overrightarrow{T}(\mathcal{C}(G))$ is a separator of its parents $P_1, \ldots, P_k$. Hence, to visit all vertices, at least k edges in L has to be used to join the parents. More precisely, each edge will replaced by a path which visits all vertices in a parent

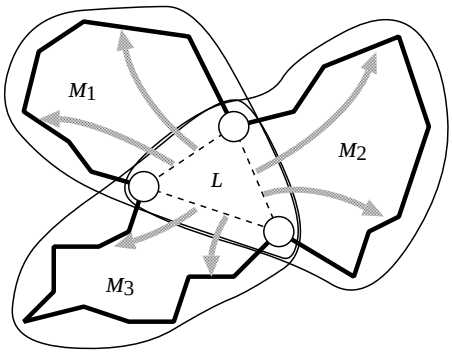


Fig. 5. Assignment of an edge to a path.

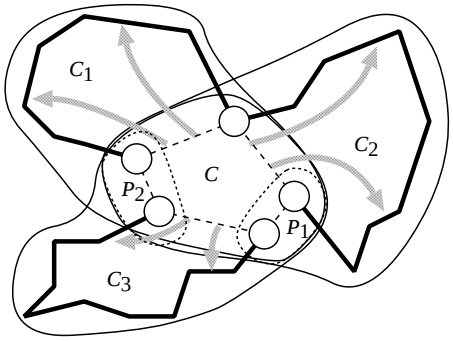


Fig. 6. Connection of children and parents.

and its ancestors. If L has enough vertices (and edges) in $\ell(L)$, they can be used. However, if $\ell(L)$ does not have enough resources, L has to use some edges which are in $\ell(C)$ where C is a child or descendants of L. This observation also means that some parents may require two or more edges from $\ell(L)$ to join their ancestors, and hence L and its descendants may have to provide more than k edges. Thus, we will say *margin* of L which is the number of edges in L that ancestors can use. For an arc (P, L), a *distribution* is that the number of edges L provides to P and its ancestors. In other words, we will consider *assignment* of a path in L to each parent. We will say that L has *feasible* distribution if the total distribution of arcs to L is less than or equal to the margin of L; that is, L and its descendants have enough edges to join their parents and ancestors. We give more detailed discussion below.

(i) We first consider a minimal vertex set L with $c(L) = 0$. By Lemma 4, each L in $\mathcal{L}(G)$ is a separator of G. We can see that if we remove L from G, we have $p(L)$ connected components. Hence, if $|L| < p(L)$, G cannot have a Hamiltonian cycle. On the other hand, when $|L| = p(L)$, any Hamiltonian cycle uses all vertices in L to connect each connected components. This fact can be seen as follows (Figure 5); we first make a cycle of length $|L|$ in L, and next replace each edge by a path through the vertices in one vertex set corresponding to a child of the node L. We then *assign* each edge to distinct parent of L. (When $|L| = 2$, we temporarily assign two (multi)edges.) If $|L| > p(L)$, we can construct a Hamiltonian cycle that uses $|L| - p(L)$ edges in $G[L]$. In this case, we need to assign $p(L)$ edges in L to construct a cycle, and we also have $|L| - p(L)$ edges which can be assigned in some other ancestors. We then define the *margin* $m(L)$ by $|L| - p(L) = |\ell(L)| - p(L)$. That is, if $m(L) < 0$, G has no Hamiltonian cycle, and if $m(L) > 0$, we have extra $m(L)$ edges in L which can be assigned in some ancestors. We note that a margin can be inherited only from a descendant to an ancestor.

We here define a *distribution* $\delta((C_j, C_i))$ of the margin, which is a function assigned to each arc $(C_j, C_i) \in \overrightarrow{T}(\mathcal{C}(G))$. Let $P_1, \ldots, P_{p(L)}$ be the parents of L. Then for $i = 1, 2, \ldots, p(L)$ each arc (P_i, L) has a distribution $\delta((P_i, L))$ with

$\sum_{i=1}^{p(L)} \delta((P_i, L)) = m(L)$. That is, each parent P_i inherits $\delta((P_i, L))$ margins from L, and some ancestors of P_i will consume $\delta((P_i, L))$ margins from L. The way to compute the distribution will be discussed later.

(ii) We next consider a vertex set L with $c(L) > 0$ and $p(L) \geq 0$, that is, L is a vertex set which is not minimal. Let $C_1, C_2, \ldots, C_h$ be children of L and $P_1, P_2, \ldots, P_k$ parents of L in $\overrightarrow{T}(\mathcal{C}(G))$. That is, we have $C_i \subset L \subset P_j$ for each i and j with $1 \leq i \leq h = c(L)$ and $1 \leq j \leq k = p(L)$ ($k = p(L) = 0$ when L is a maximal clique). We assume that $m(C_i)$ and $\delta((L, C_i))$ are already defined for each C_i, and $m(C_i) \geq 0$ (otherwise G does not have any Hamiltonian cycle). As in the first case, we have to assign $p(L)$ edges in L. In this case, each child C_i can be used as a single vertex if $\delta((L, C_i)) = 0$ (Figure 6); we first cut (remove) the assigned edge in C_i for L, and replace it by the path through all vertices in L and its parents. If $\delta((L, C_i)) > 0$ for some C_i, we can use the additional vertices to connect parents P_j. Hence the *margin* $m(L)$ is defined by $|\ell(L)| + h + \sum_{i=1}^{h} \delta((L, C_i)) - k = |\ell(L)| + \sum_{i=1}^{h} (\delta((L, C_i)) + 1) - k$. The distribution of the margin is defined as the same as in the first case; $\delta((P_i, L))$ is a function such that $\sum_{i=1}^{k} \delta((P_i, L)) = m(L)$.

The above discussion leads us to the following theorem:

Theorem 14 *Let $G = (V, E)$ be a ptolemaic graph. Then G has a Hamiltonian cycle if and only if there exist feasible distributions of margins, that is, each vertex set L in $\mathcal{C}$ satisfies $m(L) \geq 0$.*

We can see that the margin $m(M)$ for any maximal clique M is positive in case (2) since $k = 0$. In other words, every maximal clique M does not require any distribution of margins from its parents.

Our linear time algorithm, say $\mathcal{A}$, runs on $T(G)$; $\mathcal{A}$ collects the leaves in $T(G)$, computes the margins, and repeats this process by computing the margin of L such that all neighbors of L have been processed except exactly one neighbor. The precise procedure for each vertex set L is described as follows:

(1) When the vertex set L is a leaf of $T(G)$, L is a maximal clique in G, and hence $\delta((L, C))$ is set to 0, where C is the unique child of L.

(2) When L is not a leaf of $T(G)$, let $C_1, C_2, \ldots, C_h$ be children of L in $\overrightarrow{T}(\mathcal{C}(G))$, $P_1, P_2, \ldots, P_k$ parents of L in $\overrightarrow{T}(\mathcal{C}(G))$, and X be the only neighbor which is not processed. Without loss of generality, we assume that either $X = C_h$ or $X = P_k$. To simplify the notation, we define $h' = h - 1$ and $k' = k$ if $X = C_h$, and $h' = h$ and $k' = k - 1$ if $X = P_k$. We have three subcases.

(a) If L is a maximal clique in G, or $k = 0$, L requires no distribution of margins. Hence, $\mathcal{A}$ assigns $\delta((L, X)) = 0$ (since $X \subset L$).

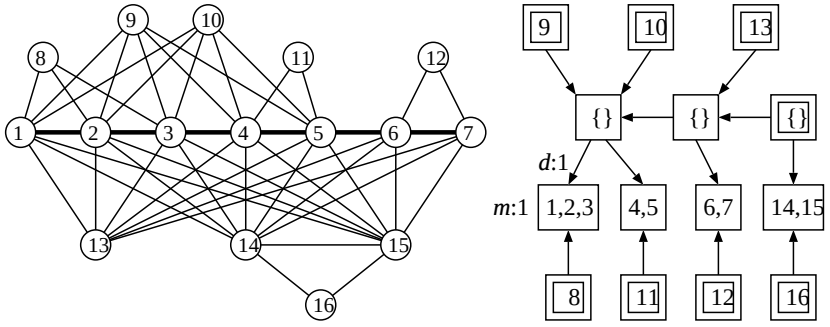


Fig. 7. Definition of margins.

(b) If L is a minimal vertex set with $k > 0$, $h = 0$, we have $X = P_k$. Then $\mathcal{A}$ first computes $m(L) = |\ell(L)| - k'$. Then, for each i with $i = 1, 2, \ldots, k'$, each parent P_i has been processed, and it requires distribution $\delta((P_i, L))$ to L. Hence $\mathcal{A}$ computes $\delta'((X, L)) = m(L) - \sum_{i=1}^{k'} \delta((P_i, L)) = |\ell(L)| - \sum_{i=1}^{k'}(\delta((P_i, L)) + 1)$. If $\delta'((X, L)) < 0$, G has no Hamiltonian cycles. Otherwise, $\mathcal{A}$ sets $\delta((X, L)) := \delta'((X, L))$.

(c) When L is not a maximal clique with $k > 0$ and $h > 0$, $\mathcal{A}$ first computes the margin $m(L) = |\ell(L)| + \sum_{i=1}^{h'}(\delta((L, C_i)) + 1) - k'$. Next, $\mathcal{A}$ distributes the margin $m(L)$ to the parents $P_1, \ldots, P_{k'}$ by computing $\delta' := m(L) - \sum_{i=1}^{k'} \delta((P_i, L)) = |\ell(L)| + \sum_{i=1}^{h'}(\delta((L, C_i)) + 1) - \sum_{i=1}^{k'}(\delta((P_i, L)) + 1)$. The value δ' indicates the margin that will be exchanged between L and X.

If $X = P_k$, that is, (X, L) is the arc in $\overrightarrow{T}(\mathcal{C}(G))$, $\mathcal{A}$ distributes all margins δ' to X, or sets $\delta((X, L)) = \delta'$. The margin can be inherited from a child to a parent. Thus, in this case, if $\delta' < 0$, G has no Hamiltonian cycles. When $\delta' \geq 0$, $\mathcal{A}$ will use the margin δ' when it processes the vertex set X.

On the other hand, if $X = C_h$, that is, (L, X) is the arc in $\overrightarrow{T}(\mathcal{C}(G))$, the margin will be distributed from X to L. Hence, if $\delta' < 0$, the vertex L borrows margin δ' from X which will be adjusted when the vertex X is chosen by $\mathcal{A}$. Thus $\mathcal{A}$ sets $\delta((L, X)) = -\delta'$ in this case. If $\delta' \geq 0$, the margin is useless since the child X only counts the number of its parents L, and does not use their margins. Therefore, $\delta((L, X))$ will not be used, and hence $\mathcal{A}$ does nothing.

(3) When L is the last node of the process; that is, every value of $\delta((L, L'))$ for each neighbor L' of L has been computed. Let $C_1, C_2, \ldots, C_h$ be children of L in $\overrightarrow{T}(\mathcal{C}(G))$, and $P_1, P_2, \ldots, P_k$ be parents of L in $\overrightarrow{T}(\mathcal{C}(G))$. In this case, $\mathcal{A}$ computes $m(L) = |\ell(L)| + \sum_{i=1}^{h}(\delta((L, C_i)) + 1) - \sum_{i=1}^{k}(\delta((P_i, L)) + 1)$. If $m(L) < 0$, L does not have enough margin. Hence G has no Hamiltonian cycle. Otherwise, every node has enough margin, and hence G has a Hamiltonian cycle.

A simple example is depicted in Figure 7, where $\{1, 2, \ldots, 7\}$ induces a clique; the node L with $\ell(L) = \{1, 2, 3\}$ has margin 1, and the arc from L' to L with

$L' = \{1, 2, 3, 4, 5\}$ has distribution 1. The other nodes have margin 0, and the other arcs have distribution 0. Hence the graph in Figure 7 has a Hamiltonian cycle, e.g., $(1, 8, 2, 9, 3, 10, 4, 11, 5, 14, 16, 15, 7, 12, 6, 13, 1)$.

The correctness of $\mathcal{A}$ can be proved by a simple induction for the number of nodes in $\overrightarrow{T}(\mathcal{C}(G))$ with Theorem 14. On the other hand, since $T(G)$ contains $O(n)$ nodes, the algorithm runs in $O(n)$ time and space, which completes the proof of Theorem 13. We note that the construction of a Hamiltonian cycle can be done simultaneously in $O(n)$ time and space.

5 Concluding Remarks

In this paper, we present a new tree representation (data structure) for ptolemaic graphs. The result enables us to use the dynamic programming technique to solve some basic problems on this graph class. We presented a linear time algorithm for the Hamiltonian cycle problem, as one of such typical examples. To develop such efficient algorithms based on the dynamic programming for other problems are future works.

We note that, recently, one of the authors and his colleagues extend the algorithm for the Hamiltonian cycle problem, and obtain a polynomial time algorithm for finding a longest cycle and path in a ptolemaic graph [30].

Acknowledgment

The authors are grateful to anonymous referees, who give numerous suggestions, and tell us the relationship between our results and those in relational database theory.

References

[1] H.-J. Bandelt and H.M. Mulder. Distance-Hereditary Graphs. *Journal of Combinatorial Theory, Series B*, 41:182–208, 1986.

[2] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, 1983.

[3] A. Brandstädt and F.F. Dragan. A Linear-Time Algorithm for Connected r-Domination and Steiner Tree on Distance-Hereditary Graphs. *Networks*, 31:177–182, 1998.

[4] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Math. and Appl., Vol. 3, 1999.

[5] A. Bretscher, D. Corneil, M. Habib, and C. Paul. A Simple Linear Time LexBFS Cograph Recognition Algorithm. In *Graph-Theoretic Concepts in Computer Science (WG 2003)*, pages 119–130. Lecture Notes in Computer Science Vol. 2880, Springer-Verlag, 2003.

[6] H.J. Broersma, E. Dahlhaus, and T. Kloks. A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Applied Mathematics*, 99:367–400, 2000.

[7] M.-S. Chang, S.-Y. Hsieh, and G.-H. Chen. Dynamic Programming on Distance-Hereditary Graphs. In *Proceedings of 8th International Symposium on Algorithms and Computation (ISAAC '97)*, pages 344–353. Lecture Notes in Computer Science Vol. 1350, Springer-Verlag, 1997.

[8] M.-S. Chang, S.-C. Wu, G.J. Chang, and H.-G. Yeh. Domination in distance-hereditary graphs. *Discrete Applied Mathematics*, 116:103–113, 2002.

[9] D.G. Corneil. Lexicographic Breadth First Search — A Survey. In *Graph-Theoretic Concepts in Computer Science (WG 2004)*, pages 1–19. Lecture Notes in Computer Science Vol. 3353, Springer-Verlag, 2004.

[10] D.G. Corneil, Y. Perl, and L.K. Stewart. A Linear Recognition Algorithm for Cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[11] G. Damiand, M. Habib, and C. Paul. A Simple Paradigm for Graph Recognition: Application to Cographs and Distance Hereditary Graphs. *Theoretical Computer Science*, 263:99–111, 2001.

[12] A. D'Atri and M. Moscarini. Distance-Hereditary Graphs, Steiner Trees, and Connected Domination. *SIAM Journal on Computing*, 17(3):521–538, 1988.

[13] A. D'Atri and M. Moscarini. On Hypergraph Acyclicity and Graph Chordality. *Information Processing Letters*, 29:271–274, 1988.

[14] R. Fagin. Degrees of Acyclicity for Hypergraphs and Relational Database Schemes. *Journal of the ACM*, 30(3):514–550, 1983.

[15] M. Farber. Independent Domination in Chordal Graphs. *Operations Research Letters*, 1(4):134–138, 1982.

[16] F. Gavril. Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

[17] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics 57. Elsevier, 2nd edition, 2004.

[18] P.L. Hammer and F. Maffray. Completely Separable Graphs. *Discrete Applied Mathematics*, 27:85–99, 1990.

[19] E. Howorka. A Characterization of Distance-Hereditary Graphs. *Quart. J. Math. Oxford (2)*, 28:417–420, 1977.

[20] E. Howorka. A Characterization of Ptolemaic Graphs. *Journal of Graph Theory*, 5:323–331, 1981.

[21] S.-Y. Hsieh, C.-W. Ho, T.-S. Hsu, and M.-T. Ko. The Hamiltonian Problem on distance-hereditary graphs. *Discrete Applied Mathematics*, 154:508–524, 2006.

[22] R.-W. Hung and M.-S. Chang. Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs. *Theoretical Computer Science*, 341:411–440, 2005.

[23] S. i. Nakano, R. Uehara, and T. Uno. A New Approach to Graph Recognition and Applications to Distance Hereditary Graphs. In *4th Annual Conference on Theory and Applications of Models of Computation (TAMC 07)*, pages 115–127. Lecture Notes in Computer Science Vol. 4484, Springer-Verlag, 2007.

[24] P.N. Klein. Efficient Parallel Algorithms for Chordal Graphs. *SIAM Journal on Computing*, 25(4):797–827, 1996.

[25] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, 1993.

[26] B. Korte and J. Vygen. *Combinatorial Optimization*, volume 21 of *Algorithms and Combinatorics*. Springer, 2000.

[27] F. Nicolai and T. Szymczak. Homogeneous Sets and Domination: A Linear Time Algorithm for Distance-Hereditary Graphs. *Networks*, 37(3):117–128, 2001.

[28] D.J. Rose, R.E. Tarjan, and G.S. Lueker. Algorithmic Aspects of Vertex Elimination on Graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.

[29] J.P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, 2003.

[30] Y. Takahara, S. Teramoto, and R. Uehara. Longest Path Problems on Ptolemaic Graphs. *IEICE Transactions*, E91-D(2):170–177, 2008.

[31] R.E. Tarjan and M. Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.

[32] R. Uehara and Y. Uno. Laminar Structure of Ptolemaic Graphs and Its Applications. In *16th Annual International Symposium on Algorithms and Computation (ISAAC 2005)*, pages 186–195. Lecture Notes in Computer Science Vol. 3827, Springer-Verlag, 2005.

[33] H.-G. Yeh and G.J. Chang. Centers and medians of distance-hereditary graphs. *Discrete Mathematics*, 265:297–310, 2003.