Laminar Structure of Ptolemaic Graphs and Its Applications

Ryuhei Uehara[1] and Yushi Uno[2]

[1] School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan. uehara@jaist.ac.jp

[2] Department of Mathematics and Information Sciences, Graduate School of Science, Osaka Prefecture University, Sakai, Japan. uno@mi.s.osakafu-u.ac.jp

Abstract. Ptolemaic graphs are graphs that satisfy the Ptolemaic inequality for any four vertices. The graph class coincides with the intersection of chordal graphs and distance hereditary graphs, and it is a natural generalization of block graphs (and hence trees). In this paper, a new characterization of ptolemaic graphs is presented. It is a laminar structure of cliques, and leads us to a canonical tree representation, which gives a simple intersection model for ptolemaic graphs. The tree representation 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 $O(n)$ time algorithm for the Hamiltonian cycle problem.

Keywords: algorithmic graph theory, data structure, Hamiltonian cycle, intersection model, ptolemaic graphs.

1 Introduction

Recently, many graph classes have been proposed and studied [2, 10]. Among them, the class of chordal graphs is classic and widely investigated. One of the reasons is that the class has a natural intersection model and hence a concise tree representation; 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 it is called a clique tree since each node of the tree corresponds to a maximal clique of the chordal graph (see [17]). Another reason is that the class is characterized by a vertex ordering 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 [16]. The LBFS is also widely investigated as a tool for recognizing several graph classes (see a comprehensive survey by Corneil [6]). Using those characterizations, many efficient algorithms have been established for chordal graphs (see, e.g., [9]).

Distance in graphs is one of the most important topics in algorithmic graph theory. The class of distance hereditary graphs was introduced by Howorka to deal with the distance property called isometric [12]. For the class, some characterizations are investigated [1, 8, 11], and many efficient algorithms have been proposed (see, e.g., [5, 4,

14]). However, the recognition of distance hereditary graphs in linear time is not simple; Hammer and Maffray's algorithm [11] fails in some cases, and Damiand, Habib, and Paul's algorithm [7] requires to build a cotree in linear time (see [7, Chapter 4] for further details), where the cotree can be constructed in linear time by using recent algorithm with multisweep LBFS approach by Bretscher, Corneil, Habib, and Paul [3].

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 [3]. 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 [13]. On the other hand, the class of ptolemaic graphs is a natural generalization of block graphs, and hence trees (see [19] for the relationships between related graph classes). However, there are relatively few known results specified to ptolemaic graphs. The reason seems that the ptolemaic graphs have no useful characterizations from the viewpoint of the algorithmic graph theory.

We propose a tree representation of ptolemaic graphs which is based on the laminar structure of cliques of a ptolemaic graph. The tree representation also gives a natural intersection model for ptolemaic graphs, which is defined over directed trees. The tree representation can be constructed in linear time. 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. 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 linear time.

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 [14]. Here, we show that the Hamiltonian cycle problem can be solved in $O(|V|)$ time using the technique if a ptolemaic graph is given in the tree representation.

Due to space limitation, some proofs are omitted, and can be found at `http://www.jaist.ac.jp/~uehara/pdf/ptolemaic2.pdf`.

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* is defined by $\bar{E} = \{\{u, v\} \mid \{u, v\} \notin E\}$, and is denoted by $\bar{G} = (V, \bar{E})$. 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*.

[3] The inequality is also known as "Ptolemy" inequality which seems to be more popular. We here use "Ptolemaic" stated by Howorka [13].

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 on 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 with 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 $N_{>i}(v_j)$. It is known that a graph is chordal iff it has a PEO (see [2]). A typical way of finding a PEO of a chordal graph in linear time is the lexicographic breadth first search (LBFS), which is introduced by Rose, Tarjan, and Lueker [16], and a comprehensive survey is presented by Corneil [6].

It is also known that a graph $G = (V, E)$ is chordal iff it is the intersection graph of subtrees of a tree T (see [2]). 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 iff T_v contains c on T. Such a tree T is called a *clique tree* of G. From a PEO of a chordal graph G, we can construct a clique tree of G in linear time [17].

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 [13]:

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, 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, for any pair of two distinct sets X and Y in $\mathcal{F}$ satisfy 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 (X, Y) iff $X \subset Y$ and there are no other subset Z such that $X \subset Z \subset Y$, for any sets X and Y. We denote the underlying graph of $\overrightarrow{T}(\mathcal{F})$ by $T(\mathcal{F}) = (\mathcal{F}, E_{\mathcal{F}})$. The following two lemmas for the laminar digraph are known (see, e.g., [15, Chapter 2.2]);

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

Hence, hereafter, we call $T(\mathcal{F})$ ($\overrightarrow{T}(\mathcal{F})$) a (directed) laminar forest. We regard each maximal (directed) tree in the laminar forest $T(\mathcal{F})$ ($\overrightarrow{T}(\mathcal{F})$) as a (directed) tree rooted at the maximal set, whose outdegree is 0 in $\overrightarrow{T}(\mathcal{F})$. We define a *label* of each node S_0 in $\overrightarrow{T}(\mathcal{F})$, 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 $S_1, S_2, \ldots, S_h$, $\ell(S_0) = S_0 \setminus (S_1 \cup S_2 \cup \cdots \cup S_h)$. 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.

3 A Tree Representation of Ptolemaic Graphs

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. The following properties are crucial.

Theorem 2. *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. Omitted. □

Lemma 2. *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. Omitted. □

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 $C_1 \subset C_2$ and there is no other C in $\mathcal{C}(G)$ such that $C_1 \subset C \subset C_2$. We denote by $T(\mathcal{C}(G))$ the underlying graph of $\overrightarrow{T}(\mathcal{C}(G))$.

Theorem 3. *A graph $G = (V, E)$ is ptolemaic if and only if the graph $T(\mathcal{C}(G))$ is a tree.*

Proof. Omitted. □

Hereafter, given a ptolemaic graph $G = (V, E)$, we call $T(\mathcal{C}(G))$ ($\overrightarrow{T}(\mathcal{C}(G))$) a *(directed) clique laminar tree* of G. We can naturally extend the label of a laminar forest to the directed clique laminar tree: Each node C_0 in $\mathcal{C}(G)$ has a label $\ell(C_0) := C_0 \setminus (C_1 \cup C_2 \cup \cdots \cup C_h)$, where (C_i, C_0) is an arc on $\overrightarrow{T}(\mathcal{C}(G))$ for $1 \leq i \leq h$. Intuitively, we additionally define the label of a maximal clique as follows; the label of a maximal clique is the set of vertices that are not contained in any other maximal cliques. We note that for each vertex in G its corresponding node in $T(\mathcal{C}(G))$ is uniquely determined by maximal cliques. Therefore, we can define the mapping from each vertex to a 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 $C_{\mathcal{M}}(v)$ or $C_{\mathcal{L}}(v)$. An example is given in Figure 2 (for the given ptolemaic graph (a), the clique laminar tree (b) is obtained after adding the vertices 16, 15, 14, 13, 12, 11, 10, 9, 8, and the clique laminar tree (c) is obtained after adding all vertices). In Figure 2, each single rectangle represents

a non-maximal clique, each double rectangle represents a maximal clique, and each rectangle contains its label.

We also note that from $\overrightarrow{T}(C(G))$ 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 $\overrightarrow{T}(C(G_1))$ is isomorphic to labeled $\overrightarrow{T}(C(G_2))$.

By Theorem 3, we obtain an intersection model for ptolemaic graphs as follows:

Corollary 1. *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 root C and all vertices reachable from 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 $\overrightarrow{T}(C(G))$ is the base directed graph of the intersection model. For each $v \in V$, we define the root C such that $v \in \ell(C)$. □

3.2 A Linear Time Construction of Clique Laminar Trees

The main theorem in this section is the following:

Theorem 4. *Given a ptolemaic graph $G = (V, E)$, the directed clique laminar tree $\overrightarrow{T}(C(G))$ can be constructed in $O(|V| + |E|)$ time.*

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

We first compute (and fix) a PEO $v_1, v_2, \ldots, v_n$ 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 [17]. 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 [17], there are two cases to consider: $N_{>i}(v_i) = C(v_j)$ or $N_{>i}(v_i) \subset C(v_j)$. The first case is sipmle; 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 have 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_L, C_M) and (C_L, C); 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 $\overrightarrow{T}(C(G))$, each node C stores $\ell(C)$. 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 $C(G) = \mathcal{M}(G) \cup \mathcal{L}(G)$. The detail of the algorithm is described as CliqueLaminarTree shown in Figure 1, and an example of the construction is depicted in Figure 2. In Figure 2, the left-hand graph gives a ptolemaic graph, and the right-hand tree is the clique laminar tree constructed according to the vertex ordering given in the figure. We show the correctness and a complexity analysis of the algorithm.

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

Algorithm 1: CliqueLaminarTree

Input : A ptolemaic graph $G = (V, E)$ with a PEO $v_1, v_2, \ldots, v_n$ obtained 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 $\left|C_M(v_j)\right| := \left|C_M(v_j)\right| + 1$;
      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 $|C_M(v_i)| := \left|C_L(v_j)\right| + 1$;
      add an arc $(C_L(v_j), C_M(v_i))$;
    case *(3)* $N_{>i}(v_i) \subset C(v_j)$ *and* $\left|\ell(C(v_j))\right| = \left|C(v_j)\right|$
      update $\ell(C(v_j)) := \ell(C(v_j)) \setminus N_{>i}(v_i)$ and $\left|\ell(C(v_j))\right| := \left|\ell(C(v_j))\right| - |N_{>i}(v_i)|$;
      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 $(L, C(v_j))$ and $(L, C_M(v_i))$;
    case *(4)* $N_{>i}(v_i) \subset C(v_j)$ *and* $\left|\ell(C(v_j))\right| < \left|C(v_j)\right|$
      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 $\left|\ell(C(v_j))\right| := \left|\ell(C(v_j))\right| - |L|$;
      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 $(L', C(v_j))$ with $L' \subset L$ and add an arc (L', L);
      add arcs $(L, C(v_j))$ and $(L, C_M(v_i))$;
    end
  end
  set the pointer from v_i to $C(v_i)$;
end
return T.

Fig. 1. A linear time algorithm for the clique laminar tree T of a ptolemaic graph $G = (V, E)$.

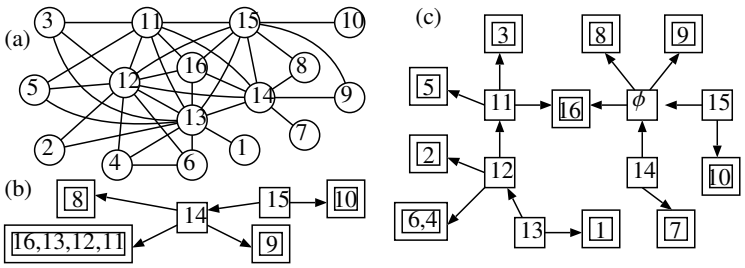

Fig. 2. A ptolemaic graph and its clique laminar tree.

Lemma 3. *[6, Theorem 1] Let $v_1, v_2, \ldots, v_n$ be a PEO found by the LBFS. Then $i < j$ implies* $\max\{N(v_i)\} \leq \max\{N(v_j)\}$.

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 $C(G)$ and their labels in $O(\deg(v_i))$ time. The following lemma drastically decreases the number of possible cases, and simplifies the algorithm.

Lemma 4. *Let v_k be* $\max\{N_{>i}(v_i)\}$. *We moreover 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. Omitted. □

We here describe the outline of the proof of Theorem 4, and the details can be found in Appendix. Since the graph G is chordal and the vertices are ordered in a PEO, $N_{>i}(v_i)$ induces a clique. By Lemma 4, 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). In case (a), 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 each case, careful analysis implies that the maintenance of the clique laminar tree can be done in $O(\deg(v_i))$ time for each v_i. Therefore the time complexity of CliqueLaminarTree is $O(n + m)$, which completes the proof of Theorem 4.

4 Applications of Clique Laminar Trees

Theorem 5. *The recognition problem for ptolemaic graphs can be solved in linear time.*

Proof. Omitted. □

Theorem 6. *The graph isomorphism problem for ptolemaic graphs can be solved in linear time.*

Proof. Omitted. □

We note that Theorem 5 is not new. Since a graph is ptolemaic iff it is chordal and distance-hereditary [13], we have Theorem 5 by combining the results in [16, 11, 7, 3]. We dare to state Theorem 5 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.)

Theorem 7. *The Hamiltonian cycle problem for ptolemaic graphs can be solved in $O(n)$ time.*

Due to space limitation, we describe the outline of the proof of Theorem 7.

We remind that $\overrightarrow{T}(C(G))$ takes $O(n)$ space. We first observe that if $\overrightarrow{T}(C(G))$ contains a vertex set C with $|C| = 1$, the vertex in C is a cutpoint of G, and hence G does not have a Hamiltonian cycle. This condition can be checked in $O(n)$ time over $\overrightarrow{T}(C(G))$. Hence, hereafter, we assume that G has no cutpoint, or equivalently, any vertex set C in C satisfies $|C| > 1$.

Let L be a vertex set in $C(G)$. Each vertex set L' with $(L, L') \in A(G)$ is said to be a *child* of L, and each vertex set L'' with $(L'', L) \in A(G)$ is said to be a *parent* of L. That is, a child L' and a parent L'' of L satisfy $L'' \subset L \subset L'$. We define ancestors and descendants for L as in ordinary trees. Note here that any node L in $\overrightarrow{T}(C(G))$ is 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 $\overrightarrow{T}(C(G))$, respectively. Hence $c(M) = 0$ for each maximal clique M, and $p(L) = 0$ for each minimal vertex set L.

We first consider a minimal vertex set L with $p(L) = 0$. By Lemma 2, each L in $\mathcal{L}(G)$ is a separator of G. It is easy to see that if we remove L from G, we have $c(L)$ connected components. Hence, if $|L| < c(L)$, G cannot have a Hamiltonian cycle. On the other hand, when $|L| = c(L)$, any Hamiltonian cycle uses all vertices in L to connect each connected component. This fact can be seen as follows; 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 say that we *assign* each edge to distinct child of L. If $|L| > c(L)$, we can construct a Hamiltonian cycle with $|L| - c(L)$ edges in $G[L]$. In this case, $c(L)$ edges in L are assigned to construct a cycle, and $|L| - c(L)$ edges are left, which can be assigned in some other descendants. We then define the *margin* $m(L)$ by $|L| - c(L) = |\ell(L)| - c(L)$. That is, if $m(L) < 0$, G has no Hamiltonian cycle, and if $m(L) > 0$, we have $m(L)$ edges in L which can be assigned in some descendants. We note that a margin can be inherited only from an ancestor to an descendant.

We next define a *distribution* $\delta((C_i, C_j))$ of the margin, which is a function assigned to each arc (C_i, C_j) in $\overrightarrow{T}(C(G))$. Let $C_1, \ldots, C_k$ be the children of L. Then for $i = 1, 2, \ldots, k$ each arc (L, C_i) has a distribution $\delta((L, C_i))$ with $\sum_{i=1}^{k} \delta((L, C_i)) = m(L)$. That is, each child C_i inherits $\delta((L, C_i))$ margins from L, and some descendants of C_i will consume $\delta((L, C_i))$ margins from L. The way to compute the distribution will be discussed later.

We then consider a vertex set C with $p(C) > 0$ and $c(C) \geq 0$, that is, C is a vertex set which is not minimal. Let $P_1, P_2, \ldots, P_h$ be parents of C and $C_1, C_2, \ldots, C_k$ children of C. That is, we have $P_i \subset C \subset C_j$ for each i and j with $1 \leq i \leq h = p(C)$ and $1 \leq j \leq k = c(C)$ ($k = c(C) = 0$ when C is maximal clique). We assume that $\delta((P_i, C))$ are already defined for each P_i. In the case, we have to assign k edges in C to the children of C. Each child will replace it by the path through all vertices in the child. We also have one assigned edge from each parent, and some additional vertices from parents P_i if $\delta((P_i, C)) > 0$. Hence the margin $m(C)$ is defined by $|\ell(C)| + h + \sum_{i=1}^{h} \delta((P_i, C)) - k = |\ell(C)| + \sum_{i=1}^{h} (\delta((P_i, C)) + 1) - k$. The distribution $\delta((C, C_i))$ of the margin $m(C)$ is defined by a function with $\sum_{i=1}^{k} \delta((C, C_i)) = m(C)$.

Above discussion leads us to the following theorem:

Theorem 8. *Let $G = (V, E)$ be a ptolemaic graph. Then G has a Hamiltonian cycle if and only if there exist feasible distributions of margins such that each vertex set C in $\mathcal{C}$ satisfies $m(C) \geq 0$.*

Our linear time algorithm, say $\mathcal{A}$, runs on $T(\mathcal{C}(G))$; $\mathcal{A}$ collects the leaves in $T(\mathcal{C}(G))$, computes the margins, and repeats this process by computing the margin of C such that all neighbors of C have been processed except exactly one neighbor. The outline of the procedure for each vertex set C with parents $P_1, P_2, \ldots, P_h$ and children $C_1, C_2, \ldots, C_k$ is described as follows:
(1) When the vertex set C is a leaf of $T(\mathcal{C}(G))$, C is a maximal clique in G, and hence $\delta((P, C))$ is set to 0, where P is the unique parent of C.
(2) When C is not a leaf of $T(\mathcal{C}(G))$, let X be the only neighbor which is not processed. Without loss of generality, we assume that either $X = P_h$ or $X = C_k$. To simplify the notation, we define $h' = h - 1$ and $k' = k$ if $X = P_h$, and $h' = h$ and $k' = k - 1$ if $X = C_k$. We have three subcases, but the most complicated case is described below.

When C is not a maximal clique with $k > 0$ and $h > 0$, $\mathcal{A}$ first computes the margin $m(C) = |\ell(C)| + \sum_{i=1}^{h'}(\delta((P_i, C)) + 1) - k'$. Next, $\mathcal{A}$ distributes the margin $m(C)$ to the children $C_1, \ldots, C_{k'}$ by computing $\delta' := m(C) - \sum_{i=1}^{k'} \delta((C, C_i)) = |\ell(C)| + \sum_{i=1}^{h'}(\delta((P_i, C)) + 1) - \sum_{i=1}^{k'}(\delta((C, C_i)) + 1)$. The value δ' indicates the margin that can be consumed by X.

If X is a child C_k, $\mathcal{A}$ distributes all margins δ' to X, or sets $\delta((C, X)) = \delta'$. 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 is a parent P_h, the margin will be distributed from X to C. Hence, if $\delta' < 0$, the vertex C borrows margin δ' from X which will be adjusted when the vertex X is chosen by $\mathcal{A}$. Thus $\mathcal{A}$ sets $\delta((X, C)) = -\delta'$ in this case. If $\delta' \geq 0$, the margin is useless since the parent X only counts the number of its children C, and does not use their margins. Therefore, $\mathcal{A}$ does nothing.
(3) When C is the last node of the process; that is, every value of $\delta((C, C'))$ or $\delta((C', C))$ for each neighbor C' of C has been computed. In the case, $\mathcal{A}$ computes $m(C) = |\ell(C)| + \sum_{i=1}^{h}(\delta((P_i, C)) + 1) - \sum_{i=1}^{k}(\delta((C, C_i)) + 1)$. If $m(C) < 0$, C does not have enough margin. Hence G has no Hamiltonian cycle. Otherwise, every node has enough margin, and hence G has a Hamiltonian cycle.

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 8. On the other hand, since $T(\mathcal{C}(G))$ contains $O(n)$ nodes, the algorithm runs in $O(n)$ time and space, which completes the proof of Theorem 7. 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 presented new tree representations (data structures) 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.

Acknowledgment

The authors thank Professor Hiro Ito, who pointed out a flaw in our polynomial time algorithm for finding a longest path in a ptolemaic graph stated in [18]. That motivated us to investigate the problems in this paper.

References

1. H.-J. Bandelt and H.M. Mulder. Distance-Hereditary Graphs. *J. of Combinatorial Theory, Series B*, 41:182–208, 1986.
2. A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999.
3. 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. LNCS Vol. 2880, Springer-Verlag, 2003.
4. 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.
5. 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. LNCS Vol. 1350, Springer-Verlag, 1997.
6. D.G. Corneil. Lexicographic Breadth First Search — A Survey. In *Graph-Theoretic Concepts in Computer Science (WG 2004)*, pages 1–19. LNCS Vol. 3353, Springer-Verlag, 2004.
7. 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.
8. A. D'Atri and M. Moscarini. Distance-Hereditary Graphs, Steiner Trees, and Connected Domination. *SIAM J. on Computing*, 17(3):521–538, 1988.
9. F. Gavril. Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph. *SIAM J. on Computing*, 1(2):180–187, 1972.
10. M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics 57. Elsevier, 2nd edition, 2004.
11. P.L. Hammer and F. Maffray. Completely Separable Graphs. *Discrete Applied Mathematics*, 27:85–99, 1990.
12. E. Howorka. A Characterization of Distance-Hereditary Graphs. *Quart. J. Math. Oxford (2)*, 28:417–420, 1977.
13. E. Howorka. A Characterization of Ptolemaic Graphs. *J. of Graph Theory*, 5:323–331, 1981.
14. 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.
15. B. Korte and J. Vygen. *Combinatorial Optimization*, volume 21 of *Algorithms and Combinatorics*. Springer, 2000.
16. D.J. Rose, R.E. Tarjan, and G.S. Lueker. Algorithmic Aspects of Vertex Elimination on Graphs. *SIAM J. on Computing*, 5(2):266–283, 1976.
17. J.P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, 2003.
18. R. Uehara and Y. Uno. Efficient Algorithms for the Longest Path Problem. In *15th Annual International Symposium on Algorithms and Computation (ISAAC 2004)*, pages 871–883. LNCS Vol.3341, Springer-Verlag, 2004.
19. H.-G. Yeh and G.J. Chang. Centers and medians of distance-hereditary graphs. *Discrete Mathematics*, 265:297–310, 2003.