From Path Graphs to Directed Path Graphs

Steve Chaplick, Marisa Gutierrez, Benjamin Lévêque, Silvia Tondato

From Path Graphs to Directed Path Graphs

Steven Chaplick[1], Marisa Gutierrez[2],
Benjamin Lévêque[3], and Silvia B. Tondato[4]

[1] University of Toronto, Canada
chaplick@cs.toronto.edu
[2] CONICET, Universidad Nacional de La Plata, Argentina
marisa@mate.unlp.edu.ar
[3] CNRS, LIRMM, Montpellier, France
benjamin.leveque@lirmm.fr
[4] Universidad Nacional de La Plata, Argentina
tondato@mate.unlp.edu.ar

Abstract. We present a linear time algorithm to greedily orient the edges of a path graph model to obtain a directed path graph model (when possible). Moreover we extend this algorithm to find an odd sun when the method fails. This algorithm has several interesting consequences concerning the relationship between path graphs and directed path graphs. One is that for a directed path graph, path graph models and directed path graph models are the same. Another consequence concerns the difference between path graphs and directed path graphs in terms of forbidden induced subgraphs. This can be used to deduce the forbidden induced subgraph characterization of directed path graphs from the forbidden induced subgraph characterization of path graphs. The last consequence is algorithmic and shows that the recognition of directed path graphs is not more difficult than the recognition of path graphs.

1 Introduction

A *hole* is a chordless cycle of length at least four. A graph is a *chordal graph* if it contains no hole as an induced subgraph. Gavril [3] proved that a graph is chordal if and only if it is the intersection graph of a family of subtrees of a tree. In this paper, whenever we talk about the intersection of subgraphs of a graph we mean that the *vertex sets* of the subgraphs intersect. A graph is an *interval graph* if it is the intersection graph of a family of intervals on the real line; or equivalently, the intersection graph of a family of subpaths of a path. The class of path graphs lies between interval graphs and chordal graphs. A graph is a *path graph* if it is the intersection graph of a family of subpaths of a tree. Two variants of path graphs have been defined when the tree is a directed graph. A *directed tree* is a directed graph whose underlying undirected graph is a tree. A *directed subpath* of a directed tree is a subpath whose edges are all oriented in the same way. A graph is a *directed path graph* if it is the intersection graph of a family of directed subpaths of a directed tree. A *rooted tree* is a directed tree in which the

D.M. Thilikos (Ed.): WG 2010, LNCS 6410, pp. 256–265, 2010.

path from a particular vertex r to every other vertex is a directed path; vertex r is called the *root*. A graph is a *rooted path graph* if it is the intersection graph of a family of directed subpaths of a rooted tree.

The following inclusions hold by definition:

$$\text{interval} \subset \text{rooted path} \subset \text{directed path} \subset \text{path} \subset \text{chordal}$$

and these inclusions are strict.

In Section 4, we present a method to greedily orient the edges of a tree T that is a path graph model to obtain a directed path graph model (when possible). The idea is very simple: Pick any non oriented edge e of T and orient it arbitrarily. Orient every edge of T that is forced by e. Repeat the process until all edges of T are oriented. In fact, to ensure that the algorithm runs in linear time the formal description of the algorithm is more complex and uses a particular order obtained by an algorithm presented in Section 3. Moreover, we extend this method to find an odd sun when the greedy path forcing fails. A *sun* is a graph with vertices $C = \{c_0, \ldots, c_r\}$, and $S = \{s_0, \ldots, s_r\}$, $r \geq 2$, where C is a clique, S is a stable set and for $0 \leq i \leq r$, $N(s_i) \cap C = \{c_{i-1}, c_i\}$ (subscripts are modulo $r+1$). An *odd sun* is a sun where $|S|$ is odd. Finding an odd sun is interesting as it certifies that the input graph is not a path graph.

This algorithm has several interesting consequences concerning the relationship between path graphs and directed path graphs presented in Section 5. One is that for a directed path graph, every path graph model has a corresponding directed path graph model. Another consequence concerns the difference between path graphs and directed path graphs in terms of forbidden induced subgraphs. This can be used to deduce the forbidden induced subgraph characterization of directed path graphs from the forbidden induced subgraph characterization of path graphs. The last consequence is algorithmic and shows that the recognition of directed path graph is not more difficult than the recognition of path graphs.

2 Definitions and Background

In a graph G, a *clique* is a set of pairwise adjacent vertices. Let $\mathcal{C}(G)$ be the set of all (inclusionwise) maximal cliques of G. For any vertex $v \in V$, let $\mathcal{C}_v(G) = \{C \in \mathcal{C}(G) : v \in C\}$. When there is no ambiguity we write $\mathcal{C}$ and $\mathcal{C}_v$ instead of $\mathcal{C}(G)$ and $\mathcal{C}_v(G)$. Given a set X of vertices, let $G[X]$ denote the subgraph of G induced by the vertices of X.

A *clique tree* T of a graph G is a tree whose vertices are the members of $\mathcal{C}$ and, for each vertex v of G, the induced subgraph $T[\mathcal{C}_v]$ is a tree. A classical result of Gavril [3] states that a graph is chordal if and only if it has a clique tree. A *clique path tree* T of G is a clique tree of G such that, for each vertex v of G, the subtree $T[\mathcal{C}_v]$ is a path. Gavril [4] proved that a graph is a path graph if and only if it has a clique path tree. A *clique directed path tree* T of G is a clique path tree of G such that edges of the tree T are directed and for each vertex v of G, the subpath $T[\mathcal{C}_v]$ is a directed path. A *clique rooted path tree* T of G is a clique directed path tree of G such that T is a rooted tree. Monma and Wei [9]

proved that a graph is a directed path graph if and only if it has a clique directed path tree, and that a graph is a rooted path graph if and only if it has a clique rooted path tree. These results allow us to restrict our attention to intersection models that are clique trees when studying the properties of these graph classes.

For more information about clique trees and chordal graphs, see [5,8].

3 Maximum Cardinality Clique Search

First, we need an algorithm (see Algorithm 1) that provides the vertex order used to accomplish the greedy path forcing algorithm (see Algorithm 2). In particular, we order the vertices of a given graph G starting from an arbitrary vertex and selecting the next vertex v_i such that the number of maximal cliques v_i shares with $v_0, ..., v_{i-1}$ is as large as possible.

Algorithm 1. *Maximum Cardinality Clique Search*

Input: A graph G and the sets $\mathcal{C}_v$ for every vertex v.
Output: An ordering σ on the vertices of G such that for every vertex v the cardinality of $\mathcal{C}_v \cap (\cup_{\sigma(u)<\sigma(v)} \mathcal{C}_u)$ is maximum.

```
1  All vertices and maximal cliques of G are non-marked.
2  Let label(v) = 0 for every vertex v.
3  for i = 1 to n do
4  |  Choose a non-marked vertex v with maximum label.
5  |  Mark v and let σ(v) = i.
6  |  foreach non-marked clique C ∈ C_v do
7  |  |  Mark C.
8  |  |  foreach u ∈ C \ {v} do
9  |  |  |  label(u) = label(u) + 1.
10 return σ.
```

Notice that the above algorithm runs in linear time with respect to its input, i.e. $O(\Sigma_{v \in V} |C_v|)$. In particular, this is the same as the number of ones in the vertex to maximal clique incidence matrix of the input graph G. Furthermore, we have the following result by Fulkerson and Gross [6]:

Theorem 1 ([6]). *For a chordal graph, the number of non-zero entries in the vertex to maximal clique incidence matrix is $O(n + m)$.*

Therefore Algorithm 1 runs in time $O(n+m)$ for a chordal graph with n vertices and m edges.

4 Greedy Path Forcing

We now provide a linear time algorithm that for any path graph G and any clique path tree T of G, returns either an orientation of T that is a clique directed path tree of G or an induced subgraph of G that is an odd-sun.

Algorithm 2 considers all the vertices of G one by one, using the order obtained by Algorithm 1, and orients their corresponding subpath in T to form a directed path without modifying already oriented edges. If the method fails, there is a subpath that cannot be oriented. This subpath cannot be oriented because it has two consecutive edges e, f which are already oriented in opposite directions. By following the sequence of vertices that leads from the orientation of e to f, it is possible to find an odd sun whose central clique corresponds to the common extremity of e and f.

The following theorem shows the correctness of Algorithm 2:

Theorem 2. *For any path graph G and any clique path tree T of G, Algorithm 2 returns in time $\mathcal{O}(n+m)$, either an orientation of T that is a clique directed path tree of G or an induced subgraph of G that is an odd-sun.*

Algorithm 2. *Greedy Path Forcing*

Input: A path graph G and a clique path tree T of G
Output: An orientation of T that is a clique directed path tree of G or an induced subgraph of G that is an odd-sun.

Extract the sets $\mathcal{C}_v$ from the clique path tree T.
Let $\sigma(v_i) = i$, $1 \leq i \leq n$, obtained by applying Algorithm 1 on G and sets $\mathcal{C}_v$.
for $i = 1$ to n **do**
  if $T[\mathcal{C}_{v_i} \cap (\cup_{\sigma(u)<i} \mathcal{C}_u)]$ is a directed path (maybe empty) **then**
    Orient the edges of $T[\mathcal{C}_{v_i}]$ that are not already oriented such that $T[\mathcal{C}_{v_i}]$ forms a directed path.
  else */* find an odd sun */*
    Let $C_{start}, C_{centre}, C_{stop}$ be three consecutive cliques of $T[\mathcal{C}_{v_i} \cap (\cup_{\sigma(u)<i} \mathcal{C}_u)]$ such that $T[C_{start}, C_{centre}, C_{stop}]$ is not a directed path.
    Mark all vertices v with $\sigma(v) \geq i$ (all other vertices are non-marked).
    Mark C_{start} (all other cliques are non-marked).
    Let $f(C_{start}) = v_i$ and $g(v_i)$ be a vertex of $C_{start} \setminus C_{centre}$.
    while C_{stop} is not marked **do**
      Choose a non marked vertex v of a marked clique C_{parent}.
      Mark v.
      Let $h(v) = f(C_{parent})$.
      if there exists a non marked clique $C \in \mathcal{C}_v \cap N(C_{centre})$ **then**
        Mark C.
        Let $f(C) = v$ and $g(v)$ be a vertex of $C \setminus C_{centre}$.
    Let $u_0 = f(C_{stop})$ and $u_j = h^j(u_0)$, $1 \leq j \leq r$, with $u_r = v_i$.
    return $G[u_0, \ldots, u_r, g(u_0), \ldots, g(u_r)]$.
Orient all not already oriented edges of T with an arbitrary direction.
return T with its orientation.

Proof. To distinguish between marked elements of Algorithm 1 and 2, we say that a clique or a vertex is marked 1 if it corresponds to the marking of Algorithm 1 and marked 2 if it corresponds to the marking of Algorithm 2. Similarly we distinguish between lines of the two algorithms by using 1.x and 2.x.

Let G be any path graph and T be any clique path tree of G. We prove that Algorithm 2 applied on G and T, returns a clique directed path tree when every test of line 2.4 is satisfied, and returns an odd sun when one such test is false (i.e., the algorithm executes lines 2.6 to 2.18 and returns an odd sun at line 2.19).

Case 1: Every test of line 2.4 is true. Every vertex is considered one by one and its corresponding subpath in T is oriented to form a directed path without modifying already oriented edges. At the end, there may still be some non-oriented edges (when G is disconnected), that are oriented arbitrarily at line 2.20. Clearly, at line 2.21, the algorithm returns an orientation of T that is a clique directed path tree of G.

In this case, the complexity of the algorithm is $O(\Sigma_{v \in V} |C_v|)$ and thus $O(n+m)$ by Theorem 1.

Case 2: At least one test of line 2.4 is false. Let i be the first time such that the test of line 2.4 is false for v_i. Also, consider the point when Algorithm 2 enters the else due to vertex v_i. In fact, there is only one time that this test can be false since the algorithm will return (at line 2.19) before leaving the scope of this else block. Let $U = \{v \in V \text{ such that } \sigma(v) < i\}$. The subgraph $T[\mathcal{C}_{v_i} \cap (\cup_{\sigma(u)<i} \mathcal{C}_u)]$ is connected by the choice of σ and thus the three cliques $C_{start}, C_{centre}, C_{stop}$ exists at line 2.7. The set $C_{start} \cap C_{stop} \cap U = \emptyset$ as $T[C_{start}, C_{centre}, C_{stop}]$ is not a directed path (line 2.7). First, we prove that the clique C_{stop} will be marked 2 during the while loop at line 2.11 (i.e., the algorithm ends).

Claim. While C_{stop} is not marked 2, there exists a non marked 2 vertex v of an already marked 2 clique and a non marked 2 clique $C \in \mathcal{C}_v \cap N(C_{centre})$ (corresponding to lines 2.12 and 2.15).

Proof. Suppose on the contrary that at one point of the loop of line 2.11 these v and C do not exist. Let $\mathcal{M}$ be the set of already marked 2 cliques. Note that $\mathcal{M} \subseteq N(C_{centre})$ as all marked 2 cliques (at line 2.9 or 2.16) are adjacent to C_{centre}. Let $\mathcal{N} = N(C_{centre}) \setminus \mathcal{M}$. Note that $C_{start} \in \mathcal{M}$ and $\mathcal{C}_{stop} \in \mathcal{N}$. Let $A = U \cap C_{centre} \cap (\cup_{C \in \mathcal{M}} C)$ and $B = U \cap C_{centre} \cap (\cup_{C \in \mathcal{N}} C)$. The set $A \cap B$ is empty, otherwise there exists $v \in A \cap B$ and $C \in \mathcal{C}_v \cap \mathcal{N}$ that can play the role of v and C as in the claim. Edges $C_{start}C_{centre}$ and $C_{stop}C_{centre}$ are already oriented, so the sets $C_{start} \cap C_{centre} \cap U$ and $C_{stop} \cap C_{centre} \cap U$ are non empty. So A and B are non empty. Let x (resp. y) be the minimum vertex for σ in A (resp. in B). We have $x \notin B$ and $y \notin A$. Let C_x be a clique of $\mathcal{C}_x \cap \mathcal{M}$. Let $t_0 = f(C_x)$ and $t_j = h^j(t_0)$, $1 \leq j \leq s$, with $t_s = v_i$. We distinguish two cases corresponding to the relation between x and y for the order σ obtained by applying Algorithm 1 at line 2.2.

Case $\sigma(x) < \sigma(y)$. Consider the point of Algorithm 1 when y is chosen at line 1.4. Note that x is already marked 1, so C_{centre} and C_x are already marked 1. Vertex y is the first vertex of B chosen by Algorithm 1. So, when y is chosen, the only marked 1 clique in $\mathcal{C}_y$ is C_{centre} and so $label(y) = 1$ (otherwise, $y \in A \cap B$).

We claim that $\sigma(t_j) < \sigma(y)$ for $0 \leq j \leq s$. Suppose the contrary and let k be minimal such that $\sigma(t_k) > \sigma(y)$. If $k = 0$, then C_x and C_{centre} are already marked 1 cliques of $\mathcal{C}_{t_0}$, so $label(t_0) \geq 2$, a contradiction to the choice of y. If $1 \leq k \leq s$, then t_{k-1} is already marked 1, so $f^{-1}(t_k)$ is already marked 1. Then $f^{-1}(t_k)$ and C_{centre} are already marked 1 cliques of $\mathcal{C}_{t_k}$, so $label(t_k) \geq 2$, a contradiction to the choice of y. Thus $\sigma(v_i) = \sigma(t_s) < \sigma(y)$, contradicting $y \in U$.

Case $\sigma(y) < \sigma(x)$. Consider the point of Algorithm 1 when x is chosen at line 1.4. Note that y is already marked 1, so C_{centre} is already marked 1. Vertex x is the first chosen vertex of A in Algorithm 1. So when it is chosen, the only marked 1 clique in $\mathcal{C}_x$ is C_{centre} and so $label(x) = 1$ (otherwise, $x \in A \cap B$).

Suppose there exists $z \in B$ with $\sigma(x) < \sigma(z)$ and let z be minimal with this property. Clearly $label(z) \leq label(x)$ when x is chosen, so $label(z) = 1$ as $C_{centre} \in \mathcal{C}_z$. Vertex z is the first vertex of B chosen after x, so its label remains the same until it is marked. Consider (temporarily) the point of Algorithm 1 when z is chosen at line 1.4. Note that x is already marked 1, so C_x is already marked 1. We claim that all t_j, $0 \leq j \leq s$, satisfy $\sigma(t_j) < \sigma(z)$. Suppose the contrary and let k minimal such that $\sigma(t_k) > \sigma(z)$. If $k = 0$, then C_x and C_{centre} are already marked 1 cliques of $\mathcal{C}_{t_0}$, so $label(t_0) \geq 2$, a contradiction to the choice of z. If $1 \leq k \leq s$, then t_{k-1} is already marked 1, so $f^{-1}(t_k)$ is already marked 1. Then $f^{-1}(t_k)$ and C_{centre} are already marked 1 cliques of $\mathcal{C}_{t_k}$, so $label(t_k) \geq 2$, a contradiction to the choice of z. Thus $\sigma(v_i) = \sigma(t_s) < \sigma(z)$, contradicting $z \in U$. So there are no vertices in B with $\sigma(x) < \sigma(z)$.

Let z be a vertex of $C_{stop} \cap C_{centre} \cap U$, thus $z \in B$. By the preceding paragraph $\sigma(z) < \sigma(x)$. We consider again the point of Algorithm 1 when x is chosen at line 1.4 with $label(x) = 1$. Cliques C_{stop} and C_{centre} are already marked 1 cliques of $\mathcal{C}_{v_i}$, so $label(v_i) \geq 2$, a contradiction to the choice of x. □

By the claim, a new clique will always be marked 2 at line 2.15 until C_{stop} is marked. So the while loop of line 2.11 ends and so the algorithm ends. Let u_j, $0 \leq j \leq r$, be as defined at line 2.18. For $0 \leq j \leq r$, let $z_j = g(u_j)$. At line 2.19, the graph G' induced by $u_0, \ldots, u_r, z_0, \ldots, z_r$ is returned. We now prove that G' is an odd sun.

For $0 \leq j \leq r$, let $C_j = f^{-1}(u_j)$. Note that $C_0 = C_{stop}$ and $C_r = C_{start}$. All of the cliques C_i are distinct and adjacent to C_{centre} so the tree $T[C_{centre}, C_0, \ldots, C_r]$ is a star centred at C_{centre}. Thus $u_0, \ldots, u_r$ is a clique Q and $z_0, \ldots, z_r$ is a stable set with $N(z_j) \cap Q = \{u_{j-1}, u_j\}$, for $0 \leq j \leq r$ and subscripts modulo $r+1$. So G' is a sun. The tree $T[C_{centre}, C_0, \ldots, C_r]$ is already oriented and $T[C_0, C_{centre}, C_r]$ is not directed by the choice of C_{start}, C_{centre}, C_{stop} of line 2.7. So C_0C_{centre} and $C_{centre}C_r$ are not directed in the same way. Suppose, by symmetry, that $C_0 \to C_{centre}$ and $C_r \to C_{centre}$ (where $a \to b$ means there is a edge oriented from a to b). Vertices C_0, C_{centre}, C_1 appear in this order along $T[\mathcal{C}_{u_0}]$ and $C_0 \to C_{centre}$, so $C_{centre} \to C_1$. Vertices C_1, C_{centre}, C_2 appear in this order along $T[\mathcal{C}_{u_1}]$ and $C_{centre} \to C_1$, so $C_2 \to C_{centre}$. Propagating this forward, for $2 \leq j < r$, vertices C_j, C_{centre}, C_{j+1} appear in this order along $T[\mathcal{C}_{u_j}]$, so $C_{j+1} \to C_{centre}$ when j is odd and $C_{centre} \to C_{j+1}$ when j is even. Thus r is even as $C_r \to C_{centre}$ and G' is an odd sun.

Vertices and maximal cliques are marked at most once in the while loop of line 2.11, so the complexity of the else part is $O(|V| + |\mathcal{C}|)$. Thus the total complexity of the algorithm is $O(n + m)$. □

5 Consequences

Theorem 2 has several consequences concerning the relationship between path graphs and directed path graphs.

First, we need the following lemma that is part of the work of Panda [10]. We give a short proof of this lemma here. One consequence of this lemma, Theorem 2, and [7] is a new proof of the main result of [10] (i.e., the forbidden induced subgraph characterization of directed path graphs).

Lemma 1 ([10]). *Odd suns are minimally not directed path graphs.*

Proof. Let G be an odd sun. Let $C = \{c_0, \ldots, c_{2k}\}$, $S = \{s_0, \ldots, s_{2k}\}$, $k \geq 1$, be the vertices of G where C is a clique, S is a stable set and for $0 \leq i \leq 2k$, $N(s_i) \cap C = \{c_{i-1}, c_i\}$ (subscripts are modulo $2k + 1$).

Suppose that G is a directed path graph and let T be a clique directed path tree of G. The maximal cliques of G are C and $C_i = \{s_i, c_{i-1}, c_i\}$ for $0 \leq i \leq 2k$. For $0 \leq i \leq 2k$, the clique C is between C_i and C_{i+1} in T as otherwise c_{i-1} is adjacent to s_{i+1} or c_{i+1} is adjacent to s_i. Thus the subtree $T[\mathcal{C}_{c_i}]$ is the path C_i, C, C_{i+1}. Suppose, by symmetry, that $C_0 \to C$. Vertices C_0, C, C_1 appear in this order along $T[\mathcal{C}_{c_0}]$ and $C_0 \to C$, so $C \to C_1$. Vertices C_1, C, C_2 appear in this order along $T[\mathcal{C}_{c_1}]$ and $C \to C_1$, so $C_2 \to C$. Propagating this forward, for $2 \leq i \leq 2k$, vertices C_i, C, C_{i+1} appear in this order along $T[\mathcal{C}_{c_i}]$, where $C_{i+1} \to C$ when i is odd and $C \to C_{i+1}$ when i is even. So for $i = 2k$, we have $C \to C_{2k+1}$ and $C_{2k+1} = C_0$, contradicting $C_0 \to C$. So G is not a directed path graph.

We now prove that G is minimally not directed path graph, that is for any vertex w of G, the graph $G \setminus w$ is a directed path graph. If $w \in S$, then we can assume that $w = s_0$. Thus the tree on vertices C, C_i, $1 \leq i \leq 2k$ and edges $C_i \to C$ when i is odd and $C \to C_i$ when i is even is a clique directed path tree of $G \setminus w$. If $w \in C$, then we can assume that $w = c_{2k}$. Thus the tree on vertices $C' = C \setminus \{w\}$, $C'_0 = C_0 \setminus \{w\}$, $C'_{2k} = C_{2k} \setminus \{w\}$, $C'_i = C_i$, $1 \leq i \leq 2k - 1$ and edges $C'_i \to C'$ when i is odd and $C' \to C'_i$ when i is even is a clique directed path tree of $G \setminus w$. □

A consequence of Theorem 2 and Lemma 1 is that, for a directed path graph, clique path trees and clique directed path trees are the same. More precisely:

Theorem 3. *For any directed path graph G and any clique path tree T of G, the edges of T can be oriented to obtain a clique directed path tree of G.*

Note that there is no analogous of Theorem 3 for rooted path graph. In fact, there exists rooted path graphs having clique (directed) path tree that cannot be oriented to obtain a clique rooted path tree (see Figure 1 for an example).

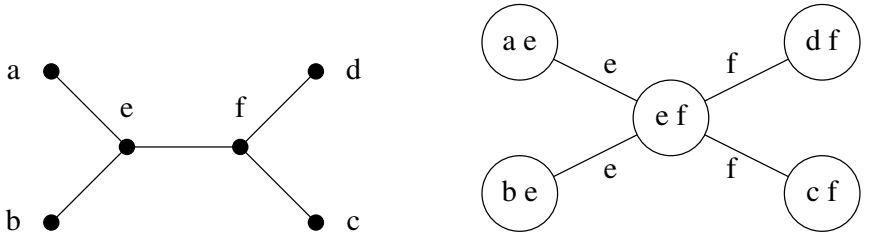

Fig. 1. A rooted path graph with a clique path tree that cannot be rooted to obtain a clique rooted path tree

Thus, there is no algorithm that can return a clique rooted path tree of a rooted path graph G by simply orienting the edges of any clique path tree of G.

Another consequence of Theorem 2 and Lemma 1 concerns the difference between path graphs and directed path graphs in terms of forbidden induced subgraphs:

Theorem 4. *A path graph is a directed path graph if and only if it does not contain an odd sun as an induced subgraph.*

Theorem 4 can be used to deduce the forbidden induced subgraph characterization of directed path graphs from the forbidden induced subgraph characterization of path graphs. The forbidden induced subgraph characterization of path graphs was obtained by Lévêque, Maffray and Preissmann [7] (see Figure 2) (an independent proof has been obtained by Tondato [12]):

Theorem 5 ([7]). *A graph is a path graph if and only if it does not contain any of* $F_0(n)_{n\geq 4}$, F_1, F_2, F_3, F_4, $F_5(n)_{n\geq 7}$, F_6, F_7, F_8, F_9, $F_{10}(n)_{n\geq 8}$, $F_{11}(4k)_{k\geq 2}$, $F_{12}(4k)_{k\geq 2}$, $F_{13}(4k+1)_{k\geq 2}$, $F_{14}(4k+1)_{k\geq 2}$, $F_{15}(4k+2)_{k\geq 2}$, $F_{16}(4k+3)_{k\geq 2}$ *as an induced subgraph.*

Form the list of minimal forbidden induced subgraphs of path graphs of Theorem 5, if we remove every graph that contains an odd sun, namely F_2, F_8, $F_{11}(4k)_{k\geq 2}$, $F_{12}(4k)_{k\geq 2}$, $F_{14}(4k+1)_{k\geq 2}$, and add the odd suns to the list, we obtain the list of minimal forbidden induced subgraphs of directed path graphs (see Figure 3):

Theorem 6 ([10]). *A graph is a directed path graph if and only if it contains no* $F_0(n)_{n\geq 4}$, F_1, F_3, F_4, $F_5(n)_{n\geq 7}$, F_6, F_7, F_9, $F_{10}(n)_{n\geq 8}$, $F_{13}(4k+1)_{k\geq 2}$, $F_{15}(4k+2)_{k\geq 2}$, $F_{16}(4k+3)_{k\geq 2}$, $F_{17}(4k+2)_{k\geq 1}$ *as an induced subgraph.*

Theorem 6 has already been proven by Panda [10] with the use of the Separator Theorem of Monma and Wei [9] and a technical case analysis. Here we obtain an independent proof using [7]. This new proof does not rely on the Separator Theorem as it is not used in the proof of Theorem 5 in [7].

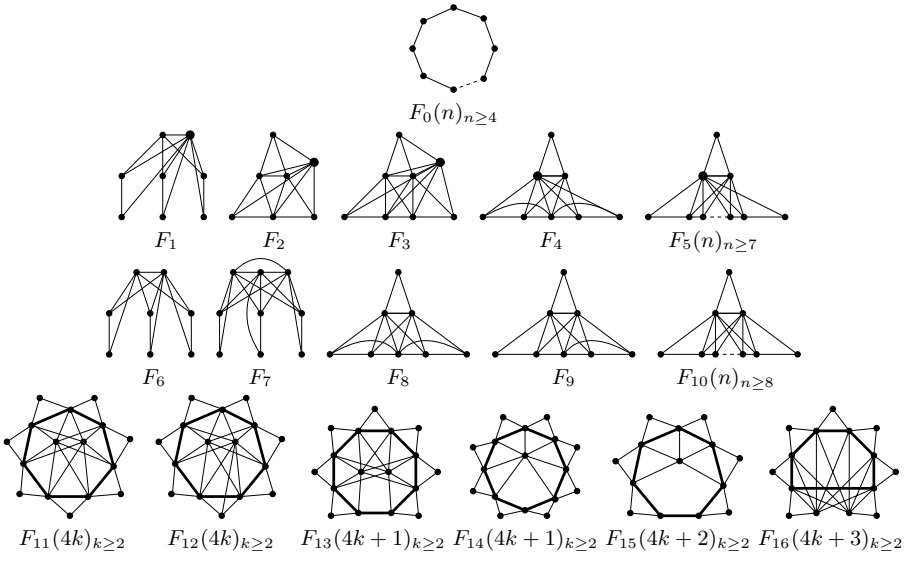

Fig. 2. Minimal forbidden induced subgraphs for path graphs (the vertices in the cycle marked by bold edges form a clique)

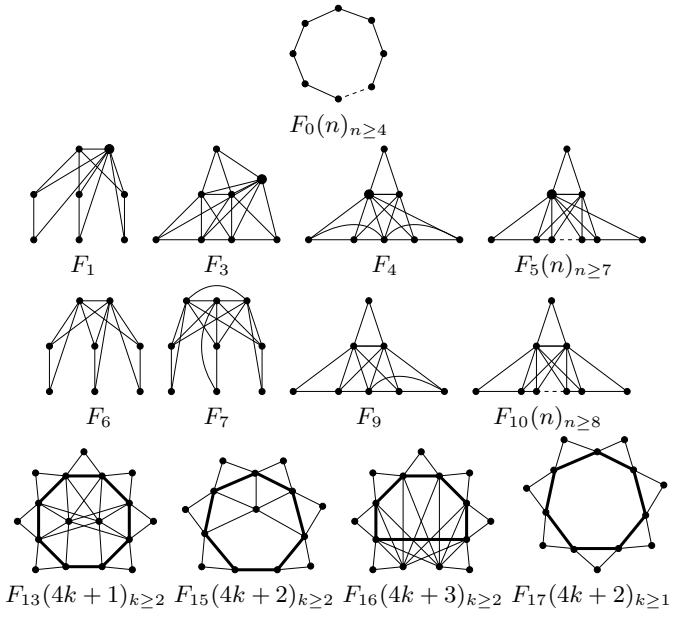

Fig. 3. Minimal forbidden induced subgraphs for directed path graphs (the vertices in the cycle marked by bold edges form a clique)

The algorithmic consequence of Theorem 2 and Lemma 1 is the following:

Theorem 7. *If there exists a polynomial algorithm that tests if a graph G is a path graph and returns a clique path tree of G when the answer is "yes." Then, there exists an algorithm with the same complexity to test if a graph is a directed path graph.*

Some efficient recognition algorithms for path graphs were given by Gavril [4], Schäffer [11] and Chaplick [1], whose complexity is respectively $O(n^4)$, $O(nm)$ and $O(nm)$ for graphs with n vertices and m edges. Another algorithm was proposed in [2] and claimed to run in $O(n + m)$ time, but it has only appeared as an extended abstract and is not considered to be complete or correct (see comments in [1, Section 2.1.4]). All of these algorithms can be extended to recognize directed path graphs with the use of Algorithm 2 without increasing the time complexity. Thus, the fastest recognition algorithm of directed path graphs with this method has complexity $O(nm)$. This is the fastest known algorithm to recognize directed path graphs. Previously, the fastest known algorithm to recognize directed path graphs was from Monma and Wei [9] and has complexity $O(n^2m)$.

References

1. Chaplick, S.: PQR-trees and undirected path graphs, M.Sc. Thesis, Dept. of Computer Science, University of Toronto, Canada (2008)
2. Dahlhaus, E., Bailey, G.: Recognition of path graphs in linear time. In: 5th Italian Conference on Theoretical Computer Science (Revello, 1995), pp. 201–210. World Sci., Publishing, River Edge (1996)
3. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. J. Combin. Theory B 16, 47–56 (1974)
4. Gavril, F.: A recognition algorithm for the intersection graphs of paths in trees. Discrete Math. 23, 211–227 (1978)
5. Golumbic, M.C.: Algorithmic graph theory and perfect graphs. Annals Disc. Math. 57 (2004)
6. Fulkerson, D.R., Gross, O.A.: Incidence matrices and interval graphs. Pacific J. Math. 15, 835–855 (1965)
7. Lévêque, B., Maffray, F., Preissmann, M.: Characterizing path graphs by forbidden induced subgraphs. Journal of Graph Theory 62, 369–384 (2009)
8. McKee, T.A., McMorris, F.R.: Topics in intersection graph theory. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (1999)
9. Monma, C.L., Wei, V.K.: Intersection graphs of paths in a tree. Journal of Combinatorial Theory B 41, 141–181 (1986)
10. Panda, B.S.: The forbidden subgraph characterization of directed vertex graphs. Discrete Mathematics 196, 239–256 (1999)
11. Schäffer, A.A.: A faster algorithm to recognize undirected path graphs. Discrete Appl. Math. 43, 261–295 (1993)
12. Tondato, S.B.: Grafos Cordales: Árboles clique y Representaciones canónicas, Doctoral Thesis, Universidad Nacional de La Plata, Argentina (2009) (in spanish)