Discrete Mathematics and Theoretical Computer Science ....
.... **vol. 27:2 #9 (2025)**

Simpler and Unified Recognition Algorithm for Path Graphs and Directed Path Graphs

Lorenzo Balzotti

Department of Statistical Sciences, Sapienza University of Rome, Rome, Italy.

revisions 6th Apr. 2024, 15th Nov. 2024; accepted 12th Mar. 2025.

A path graph is the intersection graph of paths in a tree. A directed path graph is the intersection graph of paths in a directed tree. Even if path graphs and directed path graphs are characterized very similarly, their recognition algorithms differ widely. We further unify these two graph classes by presenting the first recognition algorithm for both path graphs and directed path graphs. We deeply use a recent characterization of path graphs, and we extend it to directed path graphs. Our algorithm does not require complex data structures and has an easy and intuitive implementation, simplifying recognition algorithms for both graph classes.

Keywords: path graphs, directed path graphs, intersection graphs, recognition algorithms

1 Introduction

A *path graph* is the intersection graph of paths in a tree. A *directed path graph* is the intersection graph of paths in a directed tree. In this article we present a recognition algorithm for both path graphs and directed path graphs.

Path graphs are introduced by Renz (1970), who also gives a combinatorial, non-algorithmic characterization. The second characterization is due to Gavril (1978), and it leads him to present a first recognition algorithm with $O(n^4)$ time complexity (in this paper, the input graph has n vertices and m edges). The third characterization is due to Monma and Wei (1986), and it is used by Schäffer to build a faster recognition algorithm Schäffer (1993), that has $O(p(m+n))$ time complexity (where p is the number of *cliques*, namely, maximal induced complete subgraphs). Later, Chaplick (2008) gives a recognition algorithm with the same time complexity that uses PQR-trees. Lévêque, Maffray and Preissmann present the first characterization by forbidden subgraphs Lévêque et al. (2009), while, recently, Apollonio and Balzotti give another characterization Apollonio and Balzotti (2023), that builds on Monma and Wei (1986). Another algorithm is proposed in Dahlhaus and Bailey (1996) and claimed to run in $O(m+n)$ time, but it has only appeared as an extended abstract and is not considered to be complete or correct (see comments in [Chaplick (2008), Section 2.1.4]).

Directed path graphs are characterized by Gavril (1975), in the same article he also gives the first recognition algorithm that has $O(n^4)$ time complexity. In the article cited above, Monma and Wei (1986) give the second characterization of directed path graphs, that yields a recognition algorithm with $O(n^2m)$ time complexity. Chaplick et al. (2010) present a linear time algorithm able to establish if a path graph is a directed path graph (actually, their algorithm requires the *clique path tree* of the input graph, we refer to Section 2 for further details). This implies that the algorithms in Chaplick (2008); Schäffer (1993) can be used to obtain a recognition algorithm for directed path graphs with the same time complexity. At the state of art, this technique leads to the fastest algorithms.

Path graphs and directed path graphs are classes of graphs between *interval graphs* and *chordal graphs*. A *hole* is is a chordless cycle of length at least four. A graph is a chordal graph if it does not contain a *hole* as an induced subgraph. Gavril (1974) proves that a graph is chordal if and only if it is the intersection graph of subtrees of a tree. We can recognize chordal graphs in $O(m+n)$ time Rose et al. (1976); Tarjan and Yannakakis (1984).

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. Lekkeikerker and Boland (1962) characterize interval

ISSN 1365–8050

graphs as chordal graphs with no asteroidal triples, where an asteroidal triple is a stable set of three vertices such that each pair is connected by a path avoiding the neighborhood of the third vertex. Interval graphs can be recognized in linear time by several algorithms Booth and Lueker (1976); Corneil et al. (1998); Habib et al. (2000); Hsu (1992); Hsu and McConnell (2003); Korte and Möhring (1989); McConnell and Spinrad (1999).

We now introduce a last class of intersection graphs. A *rooted path graph* is the intersection graph of directed paths in a rooted tree. Rooted path graphs can be recognized in linear time by using the algorithm by Dietz (1984). All inclusions among the introduced classes of graphs are summarized in the following chain of strict inclusion relations:

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

Our contribution In this article we present the first recognition algorithm for both path graphs and directed path graphs, it has $O(p(m+n))$ time complexity. Our algorithm is based on the characterization of path graphs and directed path graphs given by Monma and Wei (1986), and we strictly use the recent characterization by Apollonio and Balzotti (2023) that simplifies the one in Monma and Wei (1986).

On the side of path graphs, we believe that compared to Chaplick (2008); Schäffer (1993), our algorithm provides a simpler and very shorter treatment (the whole explanation is in Section 4). Moreover, it does not need complex data structures while the algorithm in Chaplick (2008) is based on PQR-trees and the algorithm in Schäffer (1993) is a complex backtracking algorithm.

On the side of directed path graphs, at the state of art, our algorithm is the only one that does not use the results in Chaplick et al. (2010), in which it is given a linear time algorithm able to establish whether a path graph is a directed path graph too (see Theorem 5 for further details). Thus, prior to this paper, it was necessary to implement two algorithms to recognize directed path graphs: a recognition algorithm for path graphs as in Chaplick (2008); Schäffer (1993), and the algorithm in Chaplick et al. (2010) that in linear time is able to determining whether a path graph is also a directed path graph. Instead, we obtain our recognition algorithm for directed path graphs by slightly modifying the recognition algorithm for path graphs. In this way, we do not improve the running time, rather we provide a simpler algorithm using known characterizations.

Our approach The recognition algorithm `RecognizePG` for path graph is mainly built on path graphs' characterization in Apollonio and Balzotti (2023). This characterization decomposes the input graph G by clique separators as in Monma and Wei (1986), then at the recursive step one has to find a proper vertex coloring of an antipodality graph satisfying some particular conditions; see Section 3 for the definition of *clique separator* and *antipodality graph*, and Theorem 6 for the characterization. In a few words, an antipodality graph has as vertex set some subgraph of G, and two vertices are connected if the corresponding subgraphs of G are antipodal. Building all the antipodality graphs by brute force requires more time than the overall complexity of algorithms in Chaplick (2008); Schäffer (1993). We overcome this problem by visiting the connected components in a smart order. This order allows us to establish all the antipodality relations faster. This is done in Step 4, Step 5, and Step 6 that are the core of algorithm `RecognizePG`.

On the side of directed path graphs, we first extend the characterization in Apollonio and Balzotti (2023) for path graphs to directed path graphs, and then we adapt the recognition algorithm for path graphs to directed path graphs, obtaining algorithm `RecognizeDPG`.

We stress that the algorithm `RecognizePG` and algorithm `RecognizeDPG` have minimal differences, and they could be both merged in the same algorithm. However, for enhanced readability and clarity, we prefer to separate them.

Organization The paper is organized as follows. In Section 2 we present the characterization of path graphs and directed path graphs given by Monma and Wei (1986), while in Section 3 we explain the characterization of path graphs by Apollonio and Balzotti (2023). In Section 4 we present our recognition algorithm for path graphs, we prove its correctness, we present some implementation details and we compute its time complexity. Finally, in Section 5 we provide a similar analysis for directed path graphs.

2 Earlier characterizations of path graphs and directed path graphs

In this section, we present the characterization of path graphs and directed path graphs as described in Monma and Wei (1986). We start with a formal definition of these classes of graphs.

We denote by $G = (V, E)$ a finite connected undirected graph, where V, $|V| = n$, is a set of *vertices* and E, $|E| = m$, is a collection of pairs of vertices called *edges*. Let P be a finite family of nonempty sets. The intersection graph of P is obtained by associating each set in P with a vertex and by connecting two vertices with an edge exactly when their corresponding sets have a nonempty intersection. The intersection graph of a family of paths in a tree is called *path graph*. The intersection graph of a family of directed paths in a directed tree is called *directed path graph*. We say that two directed or undirected paths intersect if and only if they have at least one vertex in common.

The first characterizations of path graphs and directed path graphs are due to Gavril (1975, 1978). We let $\mathbf{C}$ denote the set of cliques of G, and for every $v \in V(G)$ let $\mathbf{C}_v = \{C \in \mathbf{C} \mid v \in C\}$. We recall that a clique is a maximal induced complete subgraph. Moreover, for a graph G and for a subset A of $V(G)$, we denote the graph induced by A in G by $G[A]$.

Theorem 1 (Gavril (1975, 1978)) *A graph $G = (V, E)$ is a path graph (resp. directed path graph) if and only if there exists a tree T (resp. directed tree T) with vertex set $\mathbf{C}$, such that for every $v \in V$, $T[\mathbf{C}_v]$ is a path (resp. directed path) in T.*

The tree T of the previous theorem is called the *clique path tree of* G if G is a path graph or the *directed clique path tree of* G if G is a directed path graph. In Figure 1, the left part shows a path graph G, and on the right there is a clique path tree of G. Symmetrically, in Figure 2, the left part shows a directed path graph G, and on the right there is a directed clique path tree of G.

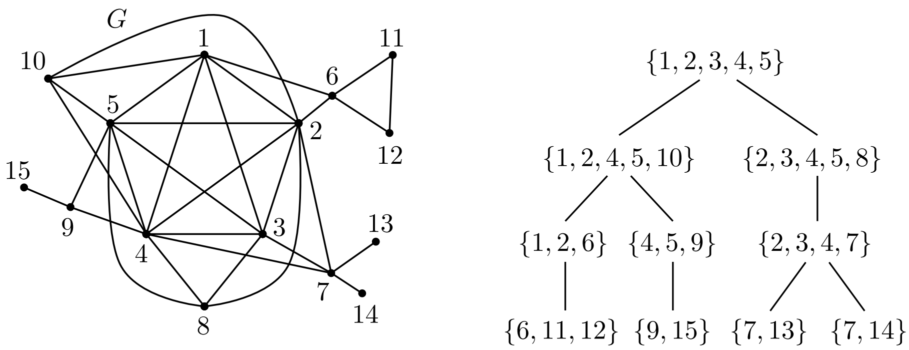

Fig. 1: a path graph G (on the left) and a clique path tree of G (on the right).

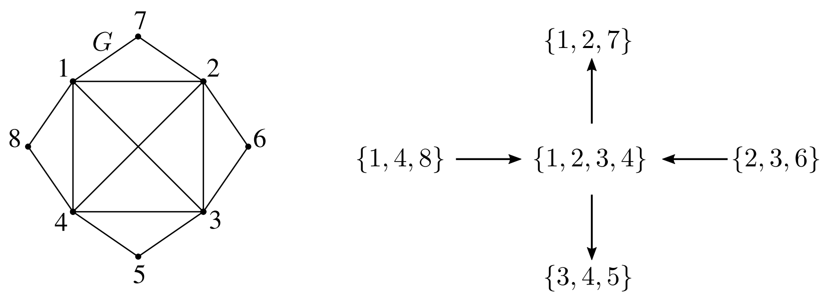

Fig. 2: a directed path graph G (on the left) and a directed clique path tree of G (on the right).

Theorem 1 specializes the celebrated characterization of chordal graphs, still due to Gavril (1974), as those graphs possessing a *clique tree* as stated below.

Theorem 2 (Gavril (1974)) *A graph G is a chordal graph if and only if it there exists a tree T, called a* clique tree*, with vertex set $\mathbf{C}$ such that, for every $v \in V$, $T[\mathbf{C}_v]$ is a tree in T.*

The main goal of our paper is: given a graph G, to find a (directed) clique path tree of G or to say that G is not a (directed) path graph. To achieve our goal, we adopt the same approach as in Monma and Wei (1986), by decomposing G recursively by clique separators.

Monma and Wei (1986) characterized several classes of intersection families, two of which are path graphs and directed path graphs. Now we present some of their theorems and lemmata that concern path graphs and directed path graphs.

A clique is a *clique separator* if its removal disconnects the graph in at least two connected components. A graph with no clique separator is called *atom*. For example, every cycle has no clique separator, and the butterfly/hourglass graph (the graph obtained by joining two copies of the cycle graph C_3 with a common vertex) has two cliques and it is an atom. In Monma and Wei (1986) it is proved that an atom is a path graph and/or a directed path graph if and only if it is a chordal graph; moreover, every chordal graph that is an atom has at most two cliques.

From now on, let us assume that a clique C separates $G = (V, E)$ into subgraphs $\gamma_i = G[C \cup V_i]$, $1 \leq i \leq s$, $s \geq 2$. Let $\Gamma_C = \{\gamma_1, \ldots, \gamma_s\}$.

Monma and Wei (1986) defined the following binary relations on Γ_C. A clique K of a member γ in Γ_C is called a *relevant clique* if $K \cap C \neq \emptyset$.

- *Attachedness*, denoted by $\bowtie$ and defined as $\gamma \bowtie \gamma'$ if and only if there is a relevant clique K of γ and a relevant clique K' of γ' such that $K \cap K' \neq \emptyset$.
- *Dominance*, denoted by $\leq$ and defined as $\gamma \leq \gamma'$ if and only if $\gamma \bowtie \gamma'$ and for each relevant clique K' of γ' either $K \cap C \subseteq K' \cap C$ or $K \cap K' \cap C = \emptyset$ for each relevant clique K of γ.
- *Antipodality*, denoted by $\leftrightarrow$ and defined as $\gamma \leftrightarrow \gamma'$ if and only if there are relevant cliques K' of γ' and K of γ such that $K \cap K' \cap C \neq \emptyset$ and $K \cap C$ and $K' \cap C$ are inclusion-wise incomparable.

We observe that if $\gamma \bowtie \gamma'$, then either $\gamma \leq \gamma'$, or $\gamma' \leq \gamma'$ or $\gamma \leftrightarrow \gamma'$. To better understand these relations we use Figure 3, that shows a clique separator C of G of Figure 1 and its connected components. We stress that $C = \{1, 2, 3, 4, 5\}$, $\Gamma_C = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5\}$, where $\gamma_1 = \{1, 2, 6, 11, 12\}$, $\gamma_2 = \{2, 3, 4, 7, 13, 14\}$, $\gamma_3 = \{2, 3, 4, 5, 8\}$, $\gamma_4 = \{4, 5, 9, 15\}$ and $\gamma_5 = \{1, 2, 4, 5, 10\}$. Clearly, $\gamma_1, \ldots, \gamma_5$ are path graphs. It holds that $\gamma_5 \geq \gamma_1$, $\gamma_5 \geq \gamma_4$, $\gamma_3 \geq \gamma_2$, $\gamma_3 \geq \gamma_4$, $\gamma_1 \leftrightarrow \gamma_2$, $\gamma_1 \leftrightarrow \gamma_3$, $\gamma_3 \leftrightarrow \gamma_5$, $\gamma_2 \leftrightarrow \gamma_5$, $\gamma_2 \leftrightarrow \gamma_4$, and the relation $\bowtie$ follows from above.

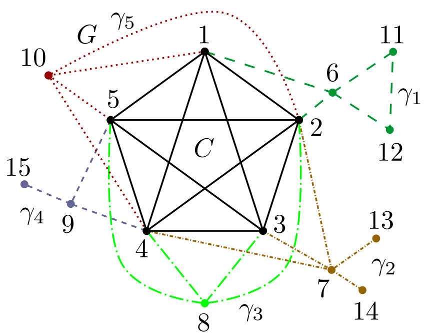

Fig. 3: a clique separator C in G of Figure 1. The connected components $\gamma_1, \ldots, \gamma_5$ are highlighted by different colors and hatchings.

Now we present the two theorems in Monma and Wei (1986) that characterize path graphs and directed path graphs. A subgraph γ_i is a neighboring subgraph of $v \in V(C)$ if some vertex in V_i is adjacent to v. For a function f and a set X we define $f(X) = \bigcup_{x \in X} f(x)$. The following theorem characterizes directed path graphs.

Theorem 3 (Monma and Wei (1986)) *A chordal graph G is a path graph if and only if G is an atom or for a clique separator C each graph $\gamma \in \Gamma_C$ is a path graph and there exists $f : \Gamma_C \to [s]$ such that:*

3.1. *if $\gamma \leftrightarrow \gamma'$, then $f(\gamma) \neq f(\gamma')$;*

3.2. *if γ, γ' and γ'' are neighboring subgraphs of v, for some vertex $v \in C$, then $|f(\{\gamma, \gamma', \gamma''\})| \leq 2$.*

Theorem 4 (Monma and Wei (1986)) *A chordal graph G is a directed path graph if and only if G is an atom or for a clique separator C each graph $\gamma \in \Gamma_C$ is a path graph and the γ_i's can be 2-colored such that no antipodal pairs have the same color.*

Note that the recursive step in Theorem 3 and Theorem 4 is a coloring problem: we have to find a proper vertex coloring f of the *antipodality graph A_C*, that is the graph on Γ_C induced by $\leftrightarrow$, formally $A_C = (\Gamma_C, \{\gamma\gamma' \mid \gamma, \gamma' \in \Gamma_C \text{ and } \gamma \leftrightarrow \gamma'\})$.

Chaplick et al. (2010) gave an algorithm that, by starting from the clique path tree of a path graph, builds the directed clique path tree, if it exists, in linear time. This result is summarized in the following theorem.

Theorem 5 (Chaplick et al. (2010)) *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.*

Theorem 5 implies that the algorithms in Chaplick (2008); Schäffer (1993) can be extended to algorithms able to recognize directed path graphs with the same time complexity.

3 A recent characterization of path graphs

In this section we introduce some results and notations in Apollonio and Balzotti (2023), that give a new characterization of path graphs presented in Theorem 6. Indirectly, some of these results allow us to efficiently recognize directed path graphs too (see Section 5 and Theorem 9).

For the following lemmata and discussions, let us fix a clique separator C of G. We use round brackets to denote ordered sets. For $\ell \in \mathbb{N}$, we denote by $[\ell]$ the interval $\{1, \dots, \ell\}$.

Let $\sim$ be the equivalence relation defined on Γ_C by $\gamma \sim \gamma' \Leftrightarrow (\gamma \leq \gamma' \wedge \gamma' \leq \gamma)$, for all $\gamma, \gamma' \in \Gamma_C$. Moreover, given $\gamma \in \Gamma_C$, we denote by $[\gamma]_\sim$ the equivalence class of γ w.r.t. $\sim$, i.e., $[\gamma]_\sim = \{\gamma' \in \Gamma_C \mid \gamma' \sim \gamma\}$. We say that $[\gamma]_\sim \leftrightarrow [\gamma']_\sim$ if and only if $\eta \leftrightarrow \eta'$ for any $\eta \in [\gamma]_\sim$ and $\eta' \in [\gamma']_\sim$; moreover, we say that $[\gamma]_\sim$ is a neighboring of v if and only if η is a neighboring of v, for any $\eta \in [\gamma]_\sim$. In the following lemma, see Lemma 2 in Apollonio and Balzotti (2023), we state that it is not restrictive to work in $\Gamma_C/\sim$.

Lemma 1 (Apollonio and Balzotti (2023)) *If there exists $f : \Gamma_C/\sim \to [\ell]$ satisfying conditions 3.1 and 3.2, then $\widetilde{f} : \Gamma_C \to [\ell]$ defined by $\widetilde{f}(\gamma) = f([\gamma]_\sim)$ satisfies conditions 3.1 and 3.2.*

From now on, unless otherwise noted, we assume that $\Gamma_C = \Gamma_C/\sim$. Even if this works from a theoretical point of view, we need to compute the classes of equivalence w.r.t. $\sim$ in our algorithms.

We define $\text{Upper}_C = \{u \in \Gamma_C \mid u \not\leq \gamma, \text{ for all } u \neq \gamma \in \Gamma_C\}$. We note that Upper_C is the set of "upper bounds" of Γ_C w.r.t. $\leq$. From now on we fix an ordering $(u_1, u_2, \dots, u_r)$ of Upper_C. For all $i < j \in [r]$ we define

$$D_i^C = \{\gamma \in \Gamma_C \mid \gamma \leq u_i \text{ and } \gamma \not\leq u_j,\ \forall j \in [r] \setminus \{i\}\}, \tag{1}$$

$$D_{i,j}^C = \{\gamma \in \Gamma_C \mid \gamma \leq u_i, \gamma \leq u_j \text{ and } \gamma \not\leq u_k,\ \forall k \in [r] \setminus \{i,j\}\}, \tag{2}$$

$$\mathcal{D}^C = \Big(\bigcup_{i \in [r]} D_i^C\Big) \cup \Big(\bigcup_{i<j \in [r]} D_{i,j}^C\Big). \tag{3}$$

If it is clear from the context, then we omit the superscript C. Note that $D_i \neq \emptyset$ for all $i \in [r]$, indeed $u_i \in D_i$. However, $D_{i,j}$ can be empty for some $i < j \in [r]$.

Given $\gamma, \gamma', \gamma'' \in \Gamma_C$, we say that $\{\gamma, \gamma', \gamma''\}$ is an *antipodal triangle* if $\gamma, \gamma', \gamma''$ are pairwise antipodal; moreover, if $\gamma, \gamma', \gamma''$ are also neighbouring of v, for some $v \in C$, then we say that $\{\gamma, \gamma', \gamma''\}$ is a *full antipodal triangle*. We note that if G is a path graph, then Theorem 3 implies that Γ_C has no full antipodal triangle, for all clique separator C of G.

In order to better understand the characterization of path graphs in Apollonio and Balzotti (2023), we provide the following lemma (see Remark 1 and Lemma 4 in Apollonio and Balzotti (2023)), explaining that the absence of full antipodal triangle in Upper_C implies a rigid structure to the antipodality graph A_C. In particular, for any $D \in D^C$, we know how antipodality works between elements in D and elements not in D.

Lemma 2 (Apollonio and Balzotti (2023)) *Let C be a clique separator of G and let $i < j \in [r]$. If Upper_C has no full antipodal triangle, then the following statements hold:*

(a) *$\mathcal{D}$ forms a partition of Γ_C,*

(b) *$\gamma \leftrightarrow \gamma'$, $\gamma \in D_{i,j}$ and $\gamma' \notin D_{i,j} \Rightarrow \gamma' \in D_i \cup D_j$,*

(c) *$\gamma \leftrightarrow \gamma'$, $\gamma \in D_i$ and $\gamma' \notin D_i \Rightarrow \gamma \leftrightarrow u_k$ for $\gamma' \le u_k$ and $k \neq i$.*

Referring to Figure 3, up to permutations, $\text{Upper}_C = (u_1, u_2) = (\gamma_3, \gamma_5)$, $\mathcal{D}^C = \{D_1, D_2, D_{1,2}\}$, $D_1 = \{\gamma_2, \gamma_3\}$, $D_2 = \{\gamma_1, \gamma_5\}$ and $D_{1,2} = \{\gamma_4\}$.

The characterization of path graphs given in Apollonio and Balzotti (2023) is summarized in the following theorem (see Definition 2, Definition 3, and Theorem 4 in Apollonio and Balzotti (2023)). Note that, as for Theorem 3, in the recursive step we have to find a proper coloring f of A_C.

Theorem 6 (Apollonio and Balzotti (2023)) *A chordal graph G is a path graph if and only if G is an atom or for a clique separator C each graph $\gamma \in \Gamma_C$ is a path graph, $\text{Upper}_C = (u_1, u_2, \dots, u_r)$ has no full antipodal triangle and there exists $f : \Gamma_C \to [r+1]$ such that:*

6.1. *for all $i \in [r]$, $f(u_i) = i$,*

6.2. *for all $i \in [r]$, $f(D_i) \subseteq \{i, r+1\}$,*

6.3. *for all $i < j \in [r]$, $f(D_{i,j}) \subseteq \{i, j\}$,*

6.4. *for all $i \in [r]$, for all $\gamma \in D_i$, if $\exists u \in \text{Upper}_C$ such that $\gamma \leftrightarrow u$, then $f(\gamma) = i$,*

6.5. *for all $i < j \in [r]$, for all $\gamma \in D_{i,j}$ such that $\exists \gamma' \in D_k$, for $k \in \{i, j\}$, satisfying $\gamma \leftrightarrow \gamma'$, then $f(\gamma) = \{i, j\} \setminus \{k\}$,*

6.6. *for all $D \in \mathcal{D}^C$, for all $\gamma, \gamma' \in D$ such that $\gamma \leftrightarrow \gamma'$, $f(\gamma) \neq f(\gamma')$.*

Let us comment briefly on Theorem 6. We have to check for the absence of a full antipodal triangle in Upper_C, this is an easy request because we are working on upper bounds. The first condition sets the colors of elements in Upper_C and the second and third conditions state which colors we can use to color elements in D, for every $D \in \mathcal{D}^C$; note that every $D \in \mathcal{D}^C$ is 2-colored by f. To better understand the fourth and fifth conditions let Cross_C be the set of elements that are antipodal to elements belonging to different sets of partition $\mathcal{D}^C$, formally $\text{Cross}_C = \{\gamma \in \Gamma_C \mid \gamma \in D \text{ for some } D \in \mathcal{D}^C \text{ and there exists } \gamma' \notin D \text{ satisfying } \gamma \leftrightarrow \gamma'\}$. The fourth and fifth conditions explain how to color elements in Cross_C. The last condition says that two antipodal elements in D, for any $D \in \mathcal{D}^C$, have different colors under f. Finally, we observe that in Apollonio and Balzotti (2023) u in condition 6.4 satisfies $u \notin D_i$, but, for algorithmic convenience, we choose $u \in \text{Upper}_C$ because of Lemma 2.

4 Recognition algorithm for path graphs

In this section we introduce algorithm `RecognizePG`, that is able to recognize path graphs. In Subsection 4.1 we present the algorithm and we prove its correctness. In Subsection 4.2 we compute its time complexity.

4.1 The algorithm and its correctness

We present the algorithm `RecognizePG`. Note that it is an implementation of Theorem 6 with very small changes. W.l.o.g., we assume that G is connected, indeed a graph G is a path graph if and only if all its connected components are path graphs. Moreover, we can obtain the clique path tree of G by merging arbitrarily the clique path tree of every connected component.

`RecognizePG`
Input: a graph G
Output: if G is a path graph, then return a clique path tree of G; else QUIT

1. Test if G is chordal. If not, then QUIT.

2. If G has at most two cliques, then return a clique path tree of G. Else find a clique separator C, let $\Gamma_C = \{\gamma_1, \ldots, \gamma_s\}$, $\gamma_i = G[V_i \cup C]$, be the set of connected components separated by C.

3. Recursively test the graphs $\gamma \in \Gamma_C$. If any one is not a path graph, then QUIT, otherwise, return a clique path tree T_γ for each $\gamma \in \Gamma_C$.

4. Compute $\Gamma_C/\sim$ and initialize $f(\gamma) = \texttt{NULL}$, for all $\gamma \in \Gamma_C/\sim$. Compute Upper_C and fix an order of its element, i.e., $\text{Upper}_C = (u_1, \ldots, u_r)$, and set $f(u_i) = i$, for all $i \in [r]$. If a full antipodal triangle in Upper_C is found, then QUIT. Compute D_i, for all $i \in [r]$, and $D_{i,j}$, for $i < j \in [r]$.

5. • For all $i \in [r]$, if there exist $\gamma \in D_i$ and $u \in \text{Upper}_C$ such that $\gamma \leftrightarrow u$, then $f(\gamma) = i$.
   • For all $i < j \in [r]$,
     – if there exist $\gamma \in D_{i,j}$, $\gamma' \in D_i$, $\gamma'' \in D_j$, such that $\gamma \leftrightarrow \gamma'$ and $\gamma \leftrightarrow \gamma''$, then QUIT,
     – if there exist $\gamma \in D_{i,j}$ and $\gamma' \in D_j$ such that $\gamma \leftrightarrow \gamma'$, then $f(\gamma) = i$,
     – if there exist $\gamma \in D_{i,j}$ and $\gamma' \in D_i$ such that $\gamma \leftrightarrow \gamma'$, then $f(\gamma) = j$.

6. • For all $i \in [r]$, extend f to all elements in D_i so that $f(D_i) \subseteq \{i, r+1\}$ and $f(\gamma) \neq f(\gamma')$ for all $\gamma, \gamma' \in D_i$ satisfying $\gamma \not\leftrightarrow \gamma'$. If it is not possible, then QUIT.
   • For all $i < j \in [r]$, extend f to all elements in $D_{i,j}$ so that $f(D_{i,j}) \subseteq \{i, j\}$ and $f(\gamma) \neq f(\gamma')$ for all $\gamma, \gamma' \in D_{i,j}$ satisfying $\gamma \not\leftrightarrow \gamma'$. If it is not possible, then QUIT.

7. Convert the coloring $f : \Gamma_C/\sim \to [r+1]$ in a clique path tree of Γ_C.

Theorem 7 *Given a graph G, algorithm* `RecognizePG` *can establish whether G is a path graph. If so, algorithm* `RecognizePG` *returns a clique path tree of G.*

Proof: The first three steps of algorithm `RecognizePG` are implied by the first part of Theorem 6. By following Theorem 6, we have to check that there are no full antipodal triangle in Upper_C (this is performed in Step 4), and we have to find $f : \Gamma_C \to [r+1]$ satisfying 6.1,...,6.6, where $r = |\text{Upper}_C|$. This latter part is done in Step 4, Step 5 and Step 6. In particular 6.1 is done in Step 4, 6.4 and 6.5 are achieved in Step 5, and 6.2, 6.3 and 6.6 are reached in Step 6. Note that the first condition in the second case of Step 5 is indirectly present in Theorem 6: if it happens, then we cannot satisfy condition 6.5 (moreover, $\gamma, \gamma', \gamma''$ would form a full antipodal triangle). Finally, Step 7 completes the recursion started in Step 3 by building the clique path tree on Γ_C. □

4.2 Implementation details and time complexity

In this section we analyze all steps of algorithm `RecognizePG`. We want to explain them in details and precise the computational complexity of the algorithm. Some of these steps are already discussed in Schäffer (1993), anyway, we describe them in order to have a complete treatment.

4.2.1 Step 1 and Step 2

We can recognize chordal graphs in $O(m+n)$ time by using either an algorithm due to Rose et al. (1976), or an algorithm due to Tarjan and Yannakakis (1984). Both recognition algorithms can be extended to an algorithm that also produces a clique tree in $O(m+n)$ time Novick (1990). In particular, we can list all cliques in G. It holds that a clique in **C** is a clique separator if and only if it is not a leaf of the clique tree.

4.2.2 Step 3

This step can be done by calling recursively algorithm `RecognizePG` for all $\gamma \in \Gamma_C$. Obviously, Step 1 has to be done only for G, indeed, the property to be chordal is inherited by subgraphs.

From now on, we are interested exclusively in the recursive part of algorithm `RecognizePG`, i.e., from Step 4 to Step 7. Thus we assume that G is a chordal graph that is not an atom and G is separated by a clique separator C. Let $\Gamma_C = \{\gamma_1, \ldots, \gamma_s\}$ be the set of connected components and we assume that all elements in Γ_C are path graphs. It holds that $s \geq 2$ because C is a clique separator.

4.2.3 Step 4

We have to compute $\Gamma_C/\sim$, this problem is already solved in Schäffer (1993). Thus we first provide some definitions and results in Schäffer (1993).

For any $\gamma \in \Gamma_C$, let T_γ be the clique path tree of γ. Let n_γ be the unique neighbour of C in T_γ (its uniqueness is proved in Monma and Wei (1986), in particular, it is proved that C is a leaf of T_γ). Moreover, let $W(\gamma) = V(n_\gamma) \cap V(C)$.

Definition 1 (Schäffer (1993)) *Let $\gamma \in \Gamma_C$ and $v \in V(C)$. We define $F(\gamma, v)$ as the node of T_γ representing the clique containing v that is furthest from C. We observe that if $v \notin W(\gamma)$, then $F(\gamma, v) = \emptyset$.*

By using Definition 1, one can obtain the following lemma.

Lemma 3 (Schäffer (1993)) *Let $\gamma, \gamma' \in \Gamma_C$. It holds that*

$$\gamma' \leq \gamma \Leftrightarrow \gamma \cap \gamma' \neq \emptyset \text{ and } F(\gamma, v) = F(\gamma, v') \text{ for all } v, v' \in W(\gamma').$$

Moreover, computing if $\gamma' \leq \gamma$ and/or $\gamma' \leftrightarrow \gamma$ costs $\min(|W(\gamma)|, |W(\gamma')|)$.

Remark 1 (Schäffer (1993)) *To compute $F(\gamma, v)$ we do one breadth-first traversal of T_γ starting at n_γ. Each time we visit a new node C', for each $v \in V(C')$, if $v \in V(C)$, we update $F(\gamma, v)$. This costs constant time for every pair (C', v) such that C' is a clique in γ and $v \in V(C')$. There are at most $m + n$ such pairs.*

Now we can explain how to compute $\Gamma_C/\sim$. First, we sort all $\gamma \in \Gamma_C$ so that the γ_i precedes γ_j if $|W(\gamma_i)| \geq |W(\gamma_j)|$: we compute $W(\gamma)$ for all $\gamma \in \Gamma_C$ (it costs $|W(\gamma)|$ for every γ), then the sorting can be executed in $O(s)$ time by using bucket sort (we remember that $s \leq n$).

Now, let $\gamma, \gamma', \gamma'' \in \Gamma_C$ satisfy $|W(\gamma)| = |W(\gamma')| = |W(\gamma'')|$ and $v \in W(\gamma) \cap W(\gamma') \cap W(\gamma'')$, for any $v \in C$. By Lemma 3, we check if $\gamma \sim \gamma'$ in $O(|W(\gamma)|)$ time. If $\gamma \not\sim \gamma'$, then either ($\gamma'' \sim \gamma$ or $\gamma'' \sim \gamma'$) or $\{\gamma, \gamma', \gamma''\}$ is a full antipodal triangle (this follows from definition of $\leftrightarrow$ and W). Hence we can compute $\Gamma_C/\sim$ in $O(\sum_{\gamma \in \Gamma_C} |W(\gamma)|)$ time, indeed every element in Γ_C need to be checked with at most two other elements in Γ_C.

To argue with the second part of Step 4, let $u(v) = \{u \in \text{Upper}_C \,|\, v \in W(u)\}$. Note that $|u(v)| \leq 2$ for all $v \in V(C)$, otherwise $u(v) \supseteq \{u_i, u_j, u_k\}$ for some $v \in V(C)$, thus $\{u_i, u_j, u_k\}$ is a full antipodal triangle. Hence, by Lemma 3, $\gamma \in \text{Upper}_C$ if and only if there not exists $u_i \in \text{Upper}_C$ such that $F(u_i, v) = F(u_i, v') \neq \emptyset$ for all $v, v' \in W(\gamma)$. Hence, to establish if γ is in Upper_C it is sufficient to look at $u(v)$ for all $v \in W(\gamma)$. By using a similar argument, we can compute D, for $D \in \mathcal{D}$. Hence Step 4 can be performed in $O(\sum_{\gamma \in \Gamma_C} |W(\gamma)|)$ time.

4.2.4 Step 5

Lemma 4 *Step 5 has $O(\sum_{\gamma \in \Gamma_C} |W(\gamma)|)$ time complexity.*

Proof: For the first case of Step 5, let $\gamma \in D_i$. It suffices to check if there exists $u \in U \setminus \{u_i\}$ neighbor of v, for some $v \in W(\gamma)$; if so, then $u \leftrightarrow \gamma$ because of the definitions of Upper_C and D_i. Thus this check costs at most $O(|W(\gamma)|)$ time.

For the second case of Step 5, let $\gamma \in D_{i,j}$, we have to check if there exists $\eta \in D_i$ such that $\gamma \leftrightarrow \eta$, the D_j's case is symmetric. Let $v \in W(\gamma)$, note that if $\eta, \eta' \in D_i$ are neighboring of v and $\eta \leftrightarrow \eta'$, then η, η' and u_j form a full antipodal triangle. So if $\eta \leftrightarrow \eta'$, then we fall in a QUIT in Step 6 for D_i; indeed $f(\eta) = f(\eta') = i$ because of Step 5 and thus f is not a 2-coloring. Hence without loss of generality with respect to the output of algorithm `RecognizePG`, we can assume that $\eta \leq \eta'$ or $\eta' \leq \eta$ for all couples $\eta, \eta' \in D_i$ neighboring of v, for all $v \in W(u_i) \cap W(u_j)$.

For any $v \in W(u_i)$, we define $m_i(v)$ the element in D_i with minimal W among all neighboring of v. We can compute $m_i(v)$, for all $v \in W(u_i)$, in total $O(\sum_{\eta \in D_i} |W(\eta)|)$ time by ordering elements in D_i w.r.t. W with bucket sort, and then visiting them in this order and updating $m_i(v)$ for all $v \in W(\eta)$ (this can be done in Step 5 in the same time complexity).

Now let $\bar{v} \in W(\gamma)$ be so that $|W(m_i(\bar{v}))| \leq |W(m_i(v))|$ for all $v \in W(\gamma)$, we check if $\gamma \leq m_i(\bar{v})$ or $\gamma \leftrightarrow m_i(\bar{v})$. The first case implies $m_i(\bar{v})$ is a neighboring of v' for all $v' \in W(\gamma)$, and thus $\gamma \leq \eta'$ for all $\eta' \in D_i$

neighboring of v, implying that there are not antipodal elements to γ in D_i. If the second case applies, then we finished the checks for γ. In both cases we spend $O(|W(\gamma)|)$ time, and the claim follows. $\square$

4.2.5 Step 6

To our goals, we introduce $f_i(\cdot)$ as $f(\cdot)$ after Step i requiring that in Step i algorithm `RecognizePG` does not terminate in QUIT. We will use f_5 and f_9.

Lemma 5 *The following statements hold:*

1. *let $i \in [r]$ and let $\gamma, \gamma' \in D_i$ be such that $\gamma' \leq \gamma$. If $f_5(\gamma) = \texttt{NULL}$, then $f_5(\gamma') = \texttt{NULL}$,*
2. *let $i < j \in [r]$ and let $\gamma, \gamma' \in D_{i,j}$ be such that $\gamma' \leq \gamma$. If $f_5(\gamma) = \texttt{NULL}$, then $f_5(\gamma') = \texttt{NULL}$.*

Proof: – Assume by contradiction that $f_5(\gamma') \neq \texttt{NULL}$ and $f_5(\gamma) = \texttt{NULL}$. By Step 5 there exists $i \neq k \in [r]$ such that $u_k \leftrightarrow \gamma'$, and thus u_k, γ' are neighboring of v, for some $v \in C$. Thus γ' is a neighboring of v by transitivity of $\leq$. Hence it holds either $\gamma \leq u_k$, or $u_k \leq \gamma$, or $u_k \leftrightarrow \gamma$. Now, $u_k \not\leq \gamma$ because $u_k \in \text{Upper}_C$, $\gamma \not\leq u_k$ because $\gamma \in D_i$ and $k \neq i$. Thus $u_k \leftrightarrow \gamma$ and Step 5 implies $f_5(\gamma) = i = f_5(\gamma')$, absurdum.
– Assume by contradiction that $f_5(\gamma') \neq \texttt{NULL}$ and $f_5(\gamma) = \texttt{NULL}$. W.l.o.g., let us assume that $f_5(\gamma') = i$. By Step 5, there exists $\eta \in D_j$ such that $\eta \leftrightarrow \gamma'$. Thus $\eta \bowtie \gamma$ by transitivity of $\leq$. Hence, as before, it holds either $\gamma \leq \eta$, or $\eta \leq \gamma$, or $\eta \leftrightarrow \gamma$. Now, $\eta \not\leq \gamma$, otherwise $\eta \in D_{i,j}$, $\gamma \not\leq \eta$, otherwise $\gamma \leq \eta$ by transitivity of $\leq$. Thus $\eta \leftrightarrow \gamma$ and Step 5 implies $f_5(\gamma) = i = f_5(\gamma')$, absurdum. $\square$

Lemma 6 *Step 6 has $O(\sum_{\gamma \in \Gamma_C} |W(\gamma)|)$ time complexity.*

Proof: It suffices to prove that Step 6 can be executed in $O(\sum_{\gamma \in D_{i,j}} |W(\gamma)|)$ for $D = D_{i,j}$ (second case). Indeed, the same procedure can be applied for $D = D_i$ (first case).

Let Colored be the set of elements in $D_{i,j}$ already colored before Step 6, i.e., Colored $= \{\gamma \in D_{i,j} \,|\, f_5 \neq \texttt{NULL}\}$. We first check that there are not $\gamma, \gamma' \in$ Colored such that $\gamma \leftrightarrow \gamma'$ and $f(\gamma) = f(\gamma')$.

We sort the elements in $D_{i,j} = (\gamma^1, \ldots, \gamma^{|D_{i,j}|})$ so that $|W(\gamma^k)| \geq |W(\gamma^{k+1})|$, for $k \in [|D_{i,j}| - 1]$ (this can be done in Step 4 without changing its time complexity). We visit the elements in Colored by following this sorting.

For any $v \in W(D_{i,j})$ we define $\ell_i(v)$ as the lowest $\gamma \in D_{i,j}$ w.r.t $\leq$ among all visited element satisfying $f_5(\gamma) = i$. Similarly, we define $\ell_j(v)$. We initialize $\ell_i(v) = \ell_j(v) = \emptyset$ for all $v \in W(D_{i,j})$.

Let $\gamma \in$ Colored, and w.l.o.g., let us assume that $f_5(\gamma) = i$. Then γ is not antipodal to previous visited elements colored with i if $\ell_i(u) = \ell_i(v)$ for all $u, v \in W(\gamma)$. Indeed, either $\ell_i(v) = \emptyset$ for all $v \in W(\gamma)$ and thus $\gamma \not\bowtie \gamma'$ for all visited γ' colored with i, or $\ell_i(v) = \gamma'$ for all $v \in W(\gamma)$ and hence there cannot exist γ'' already visited satisfying $\gamma'' \leftrightarrow \gamma$ because it would imply $\ell_i(v) = \gamma''$ for some $v \in \gamma''$.

Now we deal with blank elements. We define Blank as the set of all elements in $D_{i,j}$ that do not yet have an assigned color. We say that $\gamma \in$ Blank is *solved* if there exists $u, v \in W(\gamma)$ such that $\ell_i(u) \neq \ell_i(v)$ or $\ell_j(u) \neq \ell_j(v)$. Note that if γ is solved, then either we can set (uniquely) its color or we have to QUIT; both cases are implied by Lemma 5 and by the above reasoning done for Colored.

For all $v \in W(D_{i,j})$ we write $\gamma \in M_v$ if $v \in W(\gamma)$, $\gamma \in$ Blank and $|W(\gamma)|$ is maximal among all $\gamma' \in$ Blank satisfying $v \in W(\gamma')$. Note that $|M_v| \leq 2$ for all $v \in W(D_{i,j})$, otherwise there would be a full antipodal triangle.

Now we describe how to set the color of elements in Blank. We search (in any order) $v \in W(D_{i,j})$ such that M_v is solved. We observe that if we visit M_v before M_u, M_v is not solved, M_u is solved and $W(M_u) \cap W(M_v) \neq \emptyset$, then M_v becomes solved after the (uniquely) choose of the color of M_u. Moreover, if for all $v \in W(D_{i,j})$ M_v is not solved, then for all $\gamma \in$ Blank and $\gamma' \in D_{i,j} \setminus$ Blank it holds $\gamma \not\leftrightarrow \gamma'$ because of definition of $M_i(v)$'s. Thus the 2-coloring of $D_{i,j}$ required in Step 6 (if it exists) it is not unique, and thus we can choose $v \in W(D_{i,j})$ and the color of M_v arbitrarily.

In this way we visit in constant time every $\gamma \in D_{i,j}$ at most $O(|W(\gamma)|)$ times, and we assign its color in $O(|W(\gamma)|)$ time. Finally, we can update $\ell_i(v)$'s and $\ell_j(v)$'s every time we visit an element γ in Colored or we set the color of γ' in Blank in $O(|W(\gamma)|)$ and $O(|W(\gamma')|)$ time, respectively. We update M_v's in $O(1)$ time if we represent M_v as a vector; we build these vectors varying $v \in W(D_{i,j})$ in $O(\sum_{\gamma \in D_{i,j}} |W(\gamma)|)$ time. The claim follows. $\square$

4.2.6 Step 7

Let C be the clique separator of G fixed at Step 2. In Monma and Wei (1986) (proof of Proposition 9) it is shown how to build a clique path tree on the cliques in Γ_C starting from a coloring satisfying 3.1 and 3.2 in $O(|\Gamma_C|)$ time. Finally, by Theorem 7, f_9 satisfies 3.1 and 3.2, and Lemma 1 implies that Step 7 has $O(|\Gamma_C|)$ time complexity.

4.2.7 Time complexity

In Theorem 8 we show that the algorithm `RecognizePG` has $O(p(m+n))$ time complexity by summarizing the results of previous subsections and by using the following lemma proved in Schäffer (1993).

Lemma 7 (Schäffer (1993)) *For every clique separator C of G it holds $\sum_{\gamma\in\Gamma_C}(|W(\gamma)|) \leq m+n$.*

Theorem 8 *Algorithm `RecognizePG` can establish whether a graph G is a path graph in $O(p(m+n))$ time, where p is the number of cliques in G.*

Proof: By Theorem 7, it suffices to prove that the algorithm `RecognizePG` can be executed in $O(p(m+n))$ time. Step 1 and Step 2 have $O(m+n)$ time complexity, while Step 3 has p times the complexity of steps 4, 5, 6 and 7. Let C be the clique separator in a recursive call of Step 3. Steps 4, 5 and 6 can be executed in $O(\sum_{\gamma\in\Gamma_C}|W(\gamma)|)$ time and Step 7 has $O(|\Gamma_C|)$ time complexity. By Lemma 7 and being $|\Gamma_C| \leq n$ for every clique separator C, the claim follows. □

5 Recognition algorithm for directed path graphs

In this section we present algorithm `RecognizeDPG` that is able to recognize directed path graphs. It is based on Theorem 4 and on algorithm `RecognizePG`, both algorithms have the same time complexity.

Thanks to Theorem 4, given a clique separator C, we have to check whether the antipodality graph A_C is 2-colorable. If so, then there are no antipodal triangle in Γ_C. Moreover, the absence of an antipodal triangle implies the absence of full antipodal triangle. Thus, if there are no full antipodal triangles in Γ_C, then the rigid structure described in Lemma 2 still holds. Consequently, we obtain the following characterization by noting that the coloring in Theorem 6 is a proper vertex coloring of A_C, thus it suffices to reduce the color set from $[r+1]$ to $\{0,1\}$ and modify accordingly all the conditions.

Theorem 9 *A chordal graph G is a directed path graph if and only if G is an atom or for a clique separator C each graph $\gamma \in \Gamma_C$ is a directed path graph, $Upper_C = (u_1, u_2, \ldots, u_r)$ has no antipodal triangle and there exists $f : \Gamma_C \to \{0,1\}$ such that:*

9.1. for all $u, u' \in Upper_C$, if $u \leftrightarrow u'$, then $f(u) \neq f(u')$,

9.2. for all $i \in [r]$, for all $\gamma \in D_i$ if $\exists u \in Upper_C$ such that $\gamma \leftrightarrow u$, then $f(\gamma) = f(u_i)$,

9.3. for all $i < j \in [r]$, for all $\gamma \in D_{i,j}$ such that $\exists \gamma' \in D_k$, for $k \in \{i,j\}$, satisfying $\gamma \leftrightarrow \gamma'$, then $f(\gamma) = \{0,1\} \setminus f(u_k)$,

9.4. for all $D \in \mathcal{D}^C$, for all $\gamma, \gamma \in D$ such that $\gamma \leftrightarrow \gamma'$, $f(\gamma) \neq f(\gamma')$.

Theorem 9 implies algorithm `RecognizeDPG`. For a proof of its correctness, we redirect the reader to Theorem 7's proof, by using Theorem 9 in place of Theorem 6.

`RecognizeDPG`
Input: a graph G
Output: if G is a directed path graph, then return a directed clique path tree of G; else QUIT

1.D Test if G is chordal. If not, then QUIT.

2.D If G has at most two cliques, then return a directed clique path tree of G. Else find a clique separator C, let $\Gamma_C = \{\gamma_1, \ldots, \gamma_s\}$, $\gamma_i = G[V_i \cup C]$, be the set of connected components separated by C.

3.D Recursively test the graphs $\gamma \in \Gamma_C$. If any one is not a directed path graph, then QUIT, otherwise, return a directed clique path tree T_γ for each $\gamma \in \Gamma_C$.

4.D Compute $\Gamma_C/\sim$ and initialize $f(\gamma) = \texttt{NULL}$, for all $\gamma \in \Gamma_C/\sim$. Function f has values in $\{0, 1\}$. Compute Upper_C and fix an order of its element, i.e., $\text{Upper}_C = (u_1, \ldots, u_r)$. Assign values of f to elements in Upper_C so that antipodal elements have different color. If it is not possible, then QUIT. Compute D_i, for all $i \in [r]$, and $D_{i,j}$, for $i < j \in [r]$.

5.D For all $i \in [r]$, if there exist $\gamma \in D_i$ and $u \in \text{Upper}_C$ such that $\gamma \leftrightarrow u$, then $f(\gamma) \neq f(u)$.

6.D For all $i \in [r]$, extend f to all elements in D_i so that $f(\gamma) \neq f(\gamma')$ for all $\gamma, \gamma' \in D_i$ satisfying $\gamma \not\leftrightarrow \gamma'$. If it is not possible, then QUIT.

7.D For all $i < j \in [r]$,

- if there exist $\gamma \in D_{i,j}$, $\gamma' \in D_i$, $\gamma'' \in D_j$, such that $\gamma \leftrightarrow \gamma'$ and $\gamma \leftrightarrow \gamma''$, then QUIT,
- if there exist $\gamma \in D_{i,j}$ and $\gamma' \in D_j$ such that $\gamma \leftrightarrow \gamma'$, then $f(\gamma) = f(u_i)$,
- if there exist $\gamma \in D_{i,j}$ and $\gamma' \in D_i$ such that $\gamma \leftrightarrow \gamma'$, then $f(\gamma) = f(u_j)$.

8.D For all $i < j \in [r]$, extend f to all elements in $D_{i,j}$ so that $f(\gamma) \neq f(\gamma')$ for all $\gamma, \gamma' \in D_{i,j}$ satisfying $\gamma \not\leftrightarrow \gamma'$. If it is not possible, then QUIT.

9.D Convert the coloring $f : \Gamma_C/\sim \to \{0, 1\}$ in a directed clique path tree of Γ_C.

Any step of algorithm `RecognizeDPG` has the same time complexity of the corresponding step of algorithm `RecognizePG`, also implementation details are similar. Consequently, the following theorem applies.

Theorem 10 *Algorithm* `RecognizeDPG` *can establish whether a graph G is a directed path graph in $O(p(m + n))$ time, where p is the number of cliques in G.*

6 Conclusions

We presented the first recognition algorithm for both path graphs and directed path graphs. Both graph classes are characterized very similarly in Monma and Wei (1986), and we extended the simpler characterization of path graphs in Apollonio and Balzotti (2023) to include directed path graphs as well; this result can be of interest itself. Thus, now these two graph classes can be recognized in the same way both theoretically and algorithmically.

On the side of path graphs, we believe that, compared to algorithms in Chaplick (2008); Schäffer (1993), our algorithm is simpler for several reasons: the overall treatment is shorter, the algorithm does not require complex data structures, its correctness is a consequence of the characterization in Apollonio and Balzotti (2023), and there are fewer implementation details to achieve the same computational complexity as in Chaplick (2008); Schäffer (1993).

On the side of directed path graphs, prior to this paper, it was necessary to implement two algorithms to recognize them: a recognition algorithm for path graphs as in Chaplick (2008); Schäffer (1993), and the algorithm in Chaplick et al. (2010) that in linear time is able to determining whether a path graph is also a directed path graph. Our algorithm directly recognizes directed path graphs in the same time complexity, and the simplification is clear.

Acknowledgments

This work is partly funded by the HORIZON Research and Innovation Action 101135576 INTEND "Intent-based data operation in the computing continuum".

The author sincerely thanks the anonymous reviewers for their insightful comments and constructive feedback, which have significantly contributed to improving the quality and clarity of this manuscript.

References

N. Apollonio and L. Balzotti. Two new characterizations of path graphs. *Discrete Mathematics*, 346(12): 113596, 2023. ISSN 0012-365X. doi: https://doi.org/10.1016/j.disc.2023.113596. URL `https://www.sciencedirect.com/science/article/pii/S0012365X23002820`.

K. S. Booth and G. S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. URL `https://doi.org/10.1016/S0022-0000(76)80045-1`.

S. Chaplick. *PQR-Trees and Undirected Path Graphs*. PhD thesis, University of Toronto, 2008. URL `http://hdl.handle.net/1807/118597`.

S. Chaplick, M. Gutierrez, B. Lévêque, and S. B. Tondato. From Path Graphs to Directed Path Graphs. In *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010*, volume 6410 of *Lecture Notes in Computer Science*, pages 256–265, 2010. URL `https://doi.org/10.1007/978-3-642-16926-7_24`.

D. G. Corneil, S. Olariu, and L. Stewart. The Ultimate Interval Graph Recognition Algorithm? In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 175–180. ACM/SIAM, 1998. URL `http://dl.acm.org/citation.cfm?id=314613.314697`.

E. Dahlhaus and G. Bailey. Recognition of Path Graphs in Linear Time. In *5th Italian Conference on Theoretical Computer Science (Revello, 1995)*, pages 201–210. World Scientific, 1996.

P. F. Dietz. Intersection Graph Algorithms. Technical report, Cornell University, 1984.

F. Gavril. The Intersection Graphs of Subtrees in Trees are Exactly the Chordal Graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

F. Gavril. A Recognition Algorithm for the Intersection Graphs of Directed Paths in Directed Trees. *Discrete Mathematics*, 13(3):237–249, 1975. URL `https://doi.org/10.1016/0012-365X(75)90021-7`.

F. Gavril. A Recognition Algorithm for the Intersection Graphs of Paths in Trees. *Discrete Mathematics*, 23(3): 211–227, 1978. URL `https://doi.org/10.1016/0012-365X(78)90003-1`.

M. Habib, R. M. McConnell, C. Paul, and L. Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1-2):59–84, 2000. URL `https://doi.org/10.1016/S0304-3975(97)00241-7`.

W. Hsu. A Simple Test for Interval Graphs. In *Graph-Theoretic Concepts in Computer Science, 18th International Workshop, Proceedings*, volume 657 of *Lecture Notes in Computer Science*, pages 11–16. Springer, 1992. URL `https://doi.org/10.1007/3-540-56402-0_31`.

W. Hsu and R. M. McConnell. PC trees and circular-ones arrangements. *Theoretical Computer Science*, 296(1): 99–116, 2003. URL `https://doi.org/10.1016/S0304-3975(02)00435-8`.

N. Korte and R. H. Möhring. An Incremental Linear-Time Algorithm for Recognizing Interval Graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989. URL `https://doi.org/10.1137/0218005`.

C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

B. Lévêque, F. Maffray, and M. Preissmann. Characterizing Path Graphs by Forbidden Induced Subgraphs. *Journal of Graph Theory*, 62(4):369–384, 2009.

R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201 (1-3):189–241, 1999. URL `https://doi.org/10.1016/S0012-365X(98)00319-7`.

C. L. Monma and V. K. Wei. Intersection Graphs of Paths in a Tree. *Journal of Combinatorial Theory, Series B*, 41(2):141–181, 1986.

M. B. Novick. Parallel Algorithms for Intersection Graphs. Technical report, Cornell University, 1990.

P. Renz. Intersection Representations of Graphs by Arcs. *Pacific Journal of Mathematics*, 34(2):501–510, 1970.

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.

A. Schäffer. A Faster Algorithm to Recognize Undirected Path Graphs. *Discrete Appl. Math.*, 43(3):261–295, 1993. ISSN 0166-218X. URL `https://doi.org/10.1016/0166-218X(93)90116-6`.

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.