# Cyclability in Graph Classes

**Christophe Crespelle**
Department of Informatics, University of Bergen, Norway
Christophe.Crespelle@uib.no

**Carl Feghali**
Department of Informatics, University of Bergen, Norway
Carl.Feghali@uib.no

**Petr A. Golovach**
Department of Informatics, University of Bergen, Norway
Petr.Golovach@uib.no

## Abstract

A subset $T \subseteq V(G)$ of vertices of a graph $G$ is said to be *cyclable* if $G$ has a cycle $C$ containing every vertex of $T$, and for a positive integer $k$, a graph $G$ is *$k$-cyclable* if every subset of vertices of $G$ of size at most $k$ is cyclable. The Terminal Cyclability problem asks, given a graph $G$ and a set $T$ of vertices, whether $T$ is cyclable, and the $k$-Cyclability problem asks, given a graph $G$ and a positive integer $k$, whether $G$ is $k$-cyclable. These problems are generalizations of the classical Hamiltonian Cycle problem. We initiate the study of these problems for graph classes that admit polynomial algorithms for Hamiltonian Cycle. We show that Terminal Cyclability can be solved in linear time for interval graphs, bipartite permutation graphs and cographs. Moreover, we construct certifying algorithms that either produce a solution, that is, a cycle, or output a graph separator that certifies a no-answer. We use these results to show that $k$-Cyclability can be solved in polynomial time when restricted to the aforementioned graph classes.

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis

**Keywords and phrases** Cyclability, interval graphs, bipartite permutation graphs, cographs

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2019.16

**Funding** The research leading to these results has received funding from the Research Council of Norway via the projects "CLASSIS" and "MULTIVAL", and from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 749022.

## 1 Introduction

A subset $T \subseteq V(G)$ of vertices of a graph $G$ is said to be *cyclable* if $G$ has a cycle $C$ containing every vertex of $T$. In this case, $C$ is said to *cover* $T$. We assume that a single element set is cyclable. For a positive integer $k$, a graph $G$ is *$k$-cyclable* if every set $T$ of size at most $k$ is cyclable. The *cyclablity* of $G$, denoted $\mathsf{cyc}(G)$, is the maximum $k$ such that $G$ is $k$-cyclable. We consider the following generalizations of the classical Hamiltonian Cycle problem.

Terminal Cyclability

| | |
|---|---|
| *Input:* | A graph $G$ and a nonempty set $T \subseteq V(G)$ of *terminals*. |
| *Task:* | Decide whether $T$ is cyclable. |

$k$-Cyclability

| | |
|---|---|
| *Input:* | A graph $G$ and a positive integer $k$. |
| *Task:* | Decide whether $G$ is $k$-cyclable. |

The investigation of Terminal Cyclability and $k$-Cyclability started in the 1960s with the pioneer work of Dirac [18] who proved that, for each $k \geq 2$, every $k$-connected graph is $k$-cyclable. Since then, a number of related results have been obtained and the majority of them follow the line of research of Dirac [18]: to give sufficient conditions for a graph $G$ to be $k$-cyclable or for a given subset $T \subseteq V(G)$ to be cyclable; we refer the reader to the survey paper of Gould [22] for results of this type.

From the computational complexity viewpoint, both Terminal Cyclability and $k$-Cyclability are at least as hard as the Hamiltonian Cycle problem, which is well-known to be NP-complete [19]. Positive results can be found in the Parameterized Complexity framework (we refer to the recent book of Cygan et al. [12] for an introduction to the field). For instance, by the celebrated results of Robertson and Seymour [29] about the Disjoint Paths problem, Terminal Cyclability is fixed-parameter tractable (FPT) when parameterized by $|T|$. So far, the best known FPT (randomized) algorithm is due to Björklund, Husfeldt and Taslaman [3]. Golovach et. al. [20] also proved that deciding if $G$ is $k$-cyclable is co-W[1]-hard for split graphs and that $k$-Cyclability is FPT on planar graphs when parameterized by $k$.

There is also a long history of research on Hamiltonian Cycle and related problems for the classes of cographs, bipartite permutation graphs, interval graphs and some of their superclasses (see [5, 7, 8, 11, 13, 14, 15, 16, 23, 24, 26, 28] and the bibliography therein).

A lot of this research is connected with the conjecture of Chvátal [9]; see the survey of Bauer, Broersma and Schmeichel [2] for the statement, history and details around the conjecture. Let $c(G)$ denote the number of connected components of a graph $G$, Chvátal [9] observed that if there exists a vertex separator $S$ of a graph $G$ such that $c(G-S) > |S|$, then $G$ has no Hamiltonian cycle. Hence, the condition that $c(G-S) \leq |S|$ holds for every separator of a graph $G$ is a necessary Hamiltonicity condition. For interval graphs, bipartite permutation graphs and cographs that are connected and have at least three vertices, this condition turns out to be also be sufficient [11, 13, 15]. Motivated by this necessary condition, Jung [25] defined the *scattering* number of a noncomplete graph $G$ as

$$\mathsf{sc}(G) = \max\{c(G-S) - |S| \mid S \text{ is a separator of } G\}, \tag{1}$$

and the set $S^*$ for which the maximum in (1) is achieved is called a *scattering* set. For a complete graph $G$, $\mathsf{sc}(G) = -\infty$. For the class of cocomparability graphs $G$ with at least three vertices (that is a superclass of the classes of interval graphs and permutation graphs), the following two dualities were established in [15]. Firstly, it is shown that $G$ has a Hamiltonian cycle if and only if $\mathsf{sc}(G) \leq 0$ and, secondly, that the set of vertices of $G$ can be covered by at most $k$ vertex-disjoint paths if and only if $\mathsf{sc}(G) \leq k$.

From these equivalences, one can construct *certifying* polynomial time algorithms for Hamiltonian Path and Hamiltonian Cycle problems. Note that a certifying algorithm outputs, together with a solution, a *certificate* that demonstrates the correctness of the solution that can be verified independently. Typically, the size of a certificate should be small with respect to the input size and the verification algorithm should be simple. The main advantage of certifying algorithms over standard ones is that their implementations are a great deal more reliable and can be used without knowing the code; see the survey papers [1, 27] for an introduction to certifying algorithms. The certifying algorithms for Hamiltonian Path and Hamiltonian Cycle either output a Hamiltonian path or a Hamiltonian cycle, or produce a separator that certifies a no-answer [10, 15, 11, 7].

We continue the study of Terminal Cyclability and $k$-Cyclability from a complexity viewpoint by first showing that analogous dualities hold for these problems on interval graphs, bipartite permutation graphs and cographs (see Section 2 for the formal definitions of these

graph classes). We will then show how to construct, from these dualities, polynomial time algorithms for Terminal Cyclability and $k$-Cyclability on these graph classes, which are also certifying algorithms in the case of Terminal Cyclability. In fact, for Terminal Cyclability we will consider a slightly more general problem. To be more precise, let $G$ be a graph and let $T \subseteq V(G)$ such that $T$ is not a clique. Let $c_T(G)$ denote the number of connected components of $G$ containing some vertex of $T$ and say that a subset $S \subseteq V(G)$ is a *$T$-separator* of $G$ if $c_T(G-S) \geq 2$. The *$T$-scattering* number of $G$ is given by

$$\mathsf{sc}_T(G) = \max\{c_T(G-S) - |S| \mid S \text{ is a } T\text{-separator}\}, \tag{2}$$

and the set $S^*$ for which the maximum in (2) is achieved is called a *$T$-scattering set*. A cycle or a family of vertex-disjoint paths containing the vertices of $T$ is said to be a *$T$-cycle-segment cover*. The *size* of a $T$-cycle-segment cover is defined to be zero if it is a cycle and to be the number of paths in the family otherwise. The *$T$-cycle-segment cover number*, denoted $\mathsf{seg}_G(T)$, is the minimum size of a $T$-cycle-segment cover.

As one of our main contributions, we will show that if $G$ is an interval graph, a bipartite permutation graph or a cograph and $T$ is not a clique, then $\mathsf{seg}_G(T) \leq r$ if and only if $\mathsf{sc}_T(G) \leq r$. This, in turn, will allow us to solve in polynomial (linear) time the following decision problem that generalizes Hamiltonian Cycle and Path Cover.

Cycle Segment Cover

| | |
|---|---|
| *Input:* | A graph $G$, a nonempty set $T \subseteq V(G)$ of *terminals* and a nonnegative integer $r$. |
| *Task:* | Decide whether $\mathsf{seg}_G(T) \leq r$. |

Moreover, our algorithms for each graph class either produce a solution, that is, a $T$-cycle-segment cover, or return a $T$-separator $S^*$ such that $c_T(G-S^*) - |S^*| > r$ that certifies a no-answer (unless $T$ is a 2-clique and is not cyclable in $G$, which is the only case when there exists no $T$-separator $S^*$ that certifies a no-answer – in this case, it suffices to check whether $T$ induces a bridge in $G$). More formally, we will establish the following theorem.

▶ **Theorem 1.** *There is an algorithm that, given an instance $(G, T, r)$ of* Cycle Segment Cover*, where $G$ is an interval graph, a bipartite permutation graph or a cograph and $T$ is not a 2-clique, either finds a $T$-cycle segment cover of size at most $r$ or a $T$-separator with $c_T(G-S^*) - |S^*| > r$ that certifies a no-answer in $\mathcal{O}(|V(G)| + |E(G)|)$ time.*

In fact, for cographs we have a slightly better result: the algorithm runs in time $\mathcal{O}(|V(G)|)$ if the cotree of $G$ is given (see Section 5 for the definition). We then use these results to solve $k$-Cyclability for interval graphs, bipartite permutation graphs and cographs.

▶ **Theorem 2.** *For a graph $G$ that is an interval graph, a bipartite permutation graph or a cograph, $k$-*Cyclability *can be solved in time $\mathcal{O}(|V(G)|^3)$.*

In proving Theorem 2, the following definition will be essential. For a positive integer $k$, we define the *$k$-scattering* number of a graph $G$ as

$$\mathsf{sc}^k(G) = \max\{c(G-S) - |S| \mid S \text{ is a separator of } G \text{ s.t. } |S| \leq k-1\}, \tag{3}$$

and $\mathsf{sc}^k(G) = -\infty$ if it has no separator of size at most $k-1$ (we assume that the empty set is a separator of a disconnected graph).

To prove Theorem 2, we first show that if $G$ is an interval graph, a bipartite permutation graph or a cograph and $k$ is a positive integer, then $G$ is $k$-cyclable if and only if $\mathsf{sc}^k(G) \leq 0$. Our approach for solving $k$-Cyclability for interval graphs and cographs then consists of constructing polynomial time algorithms that compute the $k$-scattering number for these graph classes for all $k \in \{1, \ldots, |V(G)|\}$ in cubic time. For bipartite permutation graphs, we use a different approach that gives a better running time.

The extended abstract is organized as follows. In Section 3, we sketch our algorithm for Cycle Segment Cover for interval graphs and describe how to solve $k$-Cyclability for interval graphs. In Sections 4 and 5, we very briefly discuss our afore-mentioned results for, respectively, bipartite permutation graphs and cographs. We conclude the paper in Section 6 with some open problems. Due to space constraints, the details of most proofs are omitted.

## 2 Preliminaries

We consider only finite undirected simple graphs and follow the standard graph theoretic notation and terminology (see, e.g., [17]). We use $n$ to denote the number of vertices and $m$ the number of edges of the considered graphs unless it creates confusion. We say that a graph $G$ is an *interval* graph if there is a family $\mathcal{I}$ of closed intervals of the line (called *interval model* or *representation*) such that $G$ is isomorphic to the intersection graph of $\mathcal{I}$. A graph $G$ is a *permutation* graph if there is an ordering $v_1, \ldots, v_n$ of its vertices and a permutation $\pi\colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that for $1 \leq i < j \leq n$, $v_i$ and $v_j$ are adjacent in $G$ if and only of $\pi(i) > \pi(j)$. A graph is a *bipartite permutation* graph if it is both a bipartite graph and a permutation graph. A graph is a *cograph* if it has no induced subgraph isomorphic to the path on four vertices. We refer to [6, 21] for detailed introductions to these graph classes.

It is convenient to dispense with easy instances of our problems. An instance $(G, T, r)$ of Cycle Segment Cover, where $T$ is a clique, is a yes-instance, unless $|T| = 2$, $r = 0$ and $T$ induces a bridge in $G$. Similarly, an instance $(G, k)$ of $k$-Cyclability, where $G$ is a complete graph, is a yes-instance unless $|V(G)| = 2$ and $k = 2$.

▶ Remark 3. In the sequel, we assume that $G$ is not complete and $T$ is not a clique.

## 3 Interval graphs

In this section, we prove Theorems 1 and 2 for interval graphs. Our algorithms use a specific interval representation of the input graph. A *clique path* of a graph $G$ is a sequence of cliques $C_1, \ldots, C_s$ of $G$ such that

- **(i)** $C_1 \cup \ldots \cup C_s = V(G)$,
- **(ii)** for all $uv \in E(G)$, there is $i \in \{1, \ldots, s\}$ such that $u, v \in C_i$,
- **(iii)** for all $v \in V(G)$, if $v \in C_i \cap C_j$ for some $1 \leq i < j \leq s$, then $v \in C_h$ for all $h \in \{i, \ldots, j\}$.

It is usually assumed in the definition of a clique path (see, e.g., [6, 21]) that $C_1, \ldots, C_s$ are maximal cliques of $G$. Here, we relax the standard definition and do not require the cliques to be inclusion-wise maximal so that some cliques may be identical or empty. It is well-known [6, 21] that a graph is an interval graph if and only if it has a clique path. The classical recognition algorithm for interval graphs of Booth and Lueker [4] constructs a clique path in time $\mathcal{O}(n + m)$. As we intend to design an $\mathcal{O}(n + m)$-time algorithm, we can assume from now on that the input graph is given with its clique path.

For a vertex $v \in V(G)$, we let $\ell_v = \min\{i \in \{1, \ldots, s\} \mid v \in C_i\}$ and $r_v = \max\{i \in \{1, \ldots, s\} \mid v \in C_i\}$. We say that $\ell_v$ and $r_v$ are the *left bound* and *right bound* of $v$ respectively. Notice that the intervals $[\ell_v, r_v]$ of the real line for $v \in V(G)$ form an interval representation of $G$. For $1 \leq i \leq j \leq s$, we denote $C_{i,j} = \cup_{h=i}^{j} C_h$.

We use the following well-known observation about separators of interval graphs that results from the definition of a clique path (see, e.g., [6, 21]).

▶ **Observation 4.** *Let $G$ be a connected interval graph with a clique path $C_1, \ldots, C_s$. If $X = C_{1,i} \setminus C_{i+1} \neq \emptyset$ and $Y = C_{i+1,s} \setminus C_i \neq \emptyset$ for some $i \in \{1, \ldots, s-1\}$, then $C_i \cap C_{i+1}$ is a separator of $G$ such that $X$ and $Y$ are in distinct components of $G - (C_i \cap C_{i+1})$.*

In Subsection 3.1 we solve Cycle Segment Cover for interval graphs. In Subsection 3.2, we show how to compute the $k$-scattering number for interval graphs and use this result to solve $k$-Cyclability.

## 3.1 Algorithm for Terminal Cyclability and Cycle Segment Cover

In this subsection, we describe our algorithms for Terminal Cyclability and Cycle Segment Cover. More formally, we prove the following theorem.

▶ **Theorem 5.** *There is an algorithm that, given an instance $(G, T)$ of* Terminal Cyclability *where $G$ is an interval graph and $T$ is not a 2-clique, finds either a cycle of $G$ covering $T$ or a $T$-separator $S^*$ with $c_T(G - S^*) - |S^*| > 0$ that certifies a no-answer in time $\mathcal{O}(n + m)$.*

The next part of the subsection contains a sketch of the proof of Theorem 5. We construct an algorithm that tries to find a cycle of a graph $G$ that covers $T$. If it fails, we use the information obtained by the algorithm to construct a $T$-separator. The algorithm is inspired by the algorithm for finding a Hamiltonian cycle in interval graphs of Keil [26]. For us, it is more convenient to use a tailored variant of the algorithm from [7] for a more general problem as this allows us to use some results of [7] as black boxes. For this, we need some auxiliary results.

Let $G$ be an interval graph given together with its clique path $C_1, \ldots, C_s$, and let $T \subseteq V(G)$ such that $T$ is not a clique (see Remark 3). If $G$ has at least two distinct connected components containing vertices of $T$, then $G$ has no cycle covering $T$ and the algorithm returns $S^* = \emptyset$. We can thus assume that the vertices of $T$ are in the same connected component, so we can discard the other components if they exist. Clearly, all this can be done in linear time. So we can safely assume, from now on, that $G$ is connected.

Our algorithm (Algorithm 1 below) scans the clique path of $G$ from the leftmost clique to the rightmost and selects vertices from these cliques depending on their bounds. In order to break ties between a subset of vertices having the same right bound (Lines 4 and 9) or the same left bound (Line 14), we use a pre-decided arbitrary total order $\pi$ on the vertices of $G$ and always select the leftmost vertex in the subset with respect to $\pi$. Let $p = \min\{r_v \mid v \in T\}$ and $q = \max\{\ell_v \mid v \in T\}$. Let $w_b$ be the minimum vertex of $T$ with respect to $\pi$ such that $r_{w_b} = p$. Analogously, let $w_e$ be the maximum vertex of $T$ with respect to $\pi$ such that $\ell_{w_e} = q$. Since $T$ is not a clique, it follows that $p < q$, $w_b \neq w_e$ and $w_b w_e \notin E(G)$.

Algorithm 1 tries to construct two $(w_b, w_e)$-paths $P_1$ and $P_2$ that are internally vertex-disjoint such that $T \subseteq V(P_1) \cup V(P_2)$. If the algorithm succeeds, then the concatenation of $P_1$ and $P_2$ is a cycle covering $T$. Initially, $P_1 = P_2 = w_b$. Afterwards, the algorithm attaches new vertices to one of the end-vertex of the two paths, which we call the *extremity* of the path. For each $P_i$, the initial extremity of $P_i$ is $w_b$ and whenever we append a new vertex to the path, this vertex becomes the new extremity. It is easy to prove the following property.

▶ **Lemma 6.** *If Algorithm 1 returns $P_1$ and $P_2$, then $P_1$ and $P_2$ are internally vertex-disjoint $(w_b, w_e)$-paths that contain all the vertices of $T$.*

**Algorithm 1** An algorithm for interval graphs that finds two internally vertex-disjoint $(w_b, w_e)$-paths $P_1$ and $P_2$ such that $T \subseteq V(P_1) \cup V(P_2)$.

```
1  begin
2    let P_1 = P_2 = w_b;
3    for t = p to q − 1 do
4      choose P_i ∈ {P_1, P_2} such that the extremity of P_i has the leftmost right
         bound;
5      attach the vertices x ∈ T \ (V(P_1) ∪ V(P_2)) s.t. r_x = t to P_i;
6      for i = 1, 2 do
7        if the extremity of P_i has right bound at most t then
8          if the subset of vertices y ∈ (C_t ∩ C_{t+1}) \ (V(P_1) ∪ V(P_2)) is not empty
9            then extend P_i by attaching such a y having the leftmost right
               bound;
10           else report that T is not cyclable and quit;
11       end
12     end
13   end
14   attach the vertices x ∈ T \ {w_e} s.t. l_x = q to P_1, then attach w_e to P_1 and P_2;
15   return P_1 and P_2;
16 end
```

Our next aim is to show that if Algorithm 1 reports that $T$ is not cyclable, then there is a $T$-separator $S^*$ such that $c_T(G - S^*) > |S^*|$. The main observation that we shall use to construct the set $S^*$ is that for $T = V(G)$, Algorithm 1 is precisely an algorithm for finding a Hamiltonian cycle in an interval graph. Our algorithm can be interpreted, to a large extent, as a variant of Keil's algorithm [26] or of Algorithm 1 of Broersma et al. [7] (the main difference between our algorithm and theirs is that our algorithm does not try to include, in the constructed paths, all vertices that it encounters). In particular, in [7] an explicit construction is given of a separator $S$ of $G$ such that $c(G - S) > |S|$ for the case when $G$ has no Hamiltonian cycle. We adapt their approach by first altering our graph so as to allow the use of some of their results. The rest of our arguments are related to the ones in [7] but have their own features and are more than just a variation.

Assume that Algorithm 1 stops at Line 10 for $t = t^*$. Note that from the range of variation of $t$ in the main loop (Line 3), we have $t^* < q$. Denote by $P_1^*$ and $P_2^*$ the paths constructed by the algorithm before it quits. We require some additional notations from [7].

For real numbers $a \leq b$, $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$, $[a, b) = \{x \in \mathbb{R} \mid a \leq x < b\}$, and $(a, b) = \{x \in \mathbb{R} \mid a < x < b\}$. If vertex $u$ has been processed by the algorithm and attached to a path at some step $t$ of the for loop at Lines 3–13, we say that $u$ has been *activated at time* $a_u = t$. We define $a_{w_b} = p$. If $u$ is activated and a vertex $v$ has been attached to $u$ at some step $t' \geq t$ of the for loop, we say that $u$ has been *deactivated at time* $d_u = t'$. Thus, $\ell_u \leq a_u \leq d_u \leq r_u$ and $u$ is said to be *free*, *active* or *depleted* on, respectively, the intervals $[\ell_u, a_u)$, $[a_u, d_u)$ and $[d_u, r_u]$. Note that some of these intervals may be empty. Whenever we say that $u$ is free (respectively, active or depleted) on an interval $I$ of the real line, this means that $I \subseteq [\ell_u, a_u)$ (respectively, $I \subseteq [a_u, d_u)$ or $I \subseteq [d_u, r_u]$). We also say that $v \in V(P_i^*)$ for $i \in \{1, 2\}$ is a *descendant* of $u \in V(P_i^*)$ if $v$ was attached to $P_i^*$ after $u$ and that $v$ is the *last descendant* on an interval $I$ if $v$ is the last vertex attached to $P_i^*$ at steps $t \in I$ of the for loop at Lines 3–13. A vertex $v$ is said to be *renounced* if it is missed by the algorithm, that is, $\ell_v \leq t^*$ and $v \notin V(P_1^*) \cup V(P_2^*)$. The set of renounced vertices is denoted by $R$.

Let $G^* = G - R$, and let $T^* = V(G) \setminus R$. For $i \in \{1, \ldots, s\}$, denote $C_i^* = C_i \setminus R$. Clearly, $C_1^*, \ldots, C_s^*$ is a clique path of $G^*$. Recall that for $1 \leq i \leq j \leq s$, $C_{i,j}^* = \bigcup_{h=i}^{j} C_h^*$.

The description of Algorithm 1, with tie-breaking order $\pi$, implies the following property.

▶ **Lemma 7.** *Algorithm 1 for the instance $(G^*, T^*)$ of* Terminal Cyclability, *with tie-breaking order $\pi$, quits at Line 10 and constructs the paths $P_1^*$ and $P_2^*$.*

As mentioned above, the fact that our Algorithm 1 for $(G^*, T^*)$ works along the same lines as Algorithm 1 of [7] will allow us to use the following Lemma 2.2 of [7].

▶ **Lemma 8** ([7])**.** *Let $t \in \{p, \ldots, q-1\}$ such that Algorithm 1 with input $(G^*, T^*)$ either finishes iteration $t$ of the for loop at Lines 3–13 or terminates at Line 10 within iteration $t$. If there is at least one depleted vertex on the interval $(t, t+1)$, then there exists an integer $t'$ such that $p \leq t' < t$ with the following properties:*

- **(i)** *$(C_{t'+1,t}^* \setminus (C_{t'}^* \cup C_{t+1}^*)) \neq \emptyset$,*
- **(ii)** *there exists a unique vertex $u \in C_{t'}^* \cap C_{t+1}^*$ such that $u$ is active on $(t', t'+1)$ and $u$ is depleted on $(t, t+1)$,*
- **(iii)** *all vertices that are active on $(t, t+1)$ are also active on $(t', t'+1)$, with the only possible exception of the last descendant $v$ of $u$ on $(t', t+1)$ which may be free on $(t', t'+1)$,*
- **(iv)** *all vertices that are depleted on $(t, t+1)$ are also depleted on $(t', t'+1)$, except $u$ which is active on $(t', t'+1)$,*
- **(v)** *all vertices that are active on $(t', t'+1)$ are also active on $(t, t+1)$, except $u$ which is depleted on $(t, t+1)$, and*
- **(vi)** *all vertices that are free on $(t', t'+1)$ are also free on $(t, t+1)$, with the only possible exception of $v$ if it is active on $(t, t+1)$.*

For our purposes, we need one additional property (vii), stated by Lemma 9 below, which can be proved to be satisfied by the minimum $t'$ satisfying properties (i)-(vi) of Lemma 8.

▶ **Lemma 9.** *Let $t \in \{p, \ldots, q-1\}$ such that Algorithm 1 with input $(G^*, T^*)$ either finishes iteration $t$ of the for loop at Lines 3–13 or terminates at Line 10 within iteration $t$. If there is at least one depleted vertex on the interval $(t, t+1)$, then there exists an integer $t' < t$ that satisfies the conditions (i)–(vi) and the following property:*

- **(vii)** *there is $x \in V(G^*)$ such that $a_x = t'$ and $x$ is active during $(t', t'+1)$.*

We now use Lemma 9 to construct the following decreasing sequence $t_1, t_2, \ldots$ of positive integers. We set $t_1 = t^*$. Then we construct $t_{i+1}$ from the already constructed $t_i$ as follows. If, for $t = t_i$, there is at least one depleted vertex on $(t, t+1)$, then find $t' < t$ such that the conditions (i)–(vii) of Lemmas 8 and 9 are satisfied and set $t_{i+1} = t'$. We stop the construction if there is no depleted vertex on $(t, t+1)$ for $t = t_i$. Clearly, the constructed sequence is finite and we denote it by $t_1, \ldots, t_k$, with $k$ being its number of elements.

For $i \in \{1, \ldots, k\}$, we define $S_i = C_{t_i}^* \cap C_{t_i+1}^*$ and $S^* = \cup_{i=1}^{k} S_i$. We require the following crucial property of $S^*$ that was shown in the proof of Theorem 2.1 of [7].

▶ **Lemma 10** ([7])**.** *The set $S^*$ is a separator of $G^*$ and $c(G^* - S^*) \geq k + 1 > |S^*|$.*

From Lemma 10, we establish an essential result for the proof of Theorem 5.

▶ **Lemma 11.** *The set $S^*$ is a $T$-separator in $G$ and $c_T(G - S^*) > |S^*|$.*

Mindful of Lemma 10, Lemma 11 intuitively states that the set $R$ of renounced vertices of $G$ does not play an important role in finding a $T$-separator of $G$ whose removal "maximises" the number of resulting components containing some member of $T$.

**Proof.** We use the second inequality of Lemma 10 ($k+1 > |S^*|$) and we prove in addition that $c_T(G-S^*) \geq k+1$. To this purpose, we define subsets $X_i$ with $0 \leq i \leq k$ (see below) for which we show that each $X_i$ has a non-empty intersection with $T$ (Claim 12) and the sets $X_i$ are separated by $S^*$ in $G$ (Claim 13). Let $X_k = C^*_{1,t_k} \setminus C^*_{t_k+1}$, $X_j = C^*_{t_{j+1}+1,t_j} \setminus (C^*_{t_{j+1}} \cup C^*_{t_j+1})$ for $j \in \{1,\ldots,k-1\}$ and $X_0 = C^*_{t_1+1,s} \setminus C^*_{t_1}$. We have two claims.

▷ Claim 12. For all $i$ such that $0 \leq i \leq k$, $X_i \cap T \neq \emptyset$.

Let us first argue that $w_b \in X_k$ and $w_e \in X_0$. As the main loop of Algorithm 1 (Lines 3–13) starts iterating with $t = p$, there is no depleted vertex on $(t, t+1)$ for $t < p$. Hence $t_k \geq p$ and given that $w_b \in C_p \setminus C_{p+1}$ it follows that $w_b \in C^*_p \setminus C^*_{p+1}$. In other words, $w_b \in C^*_{1,p} \setminus C^*_{p+1}$ and so $w_b \in X_k$. Similarly, $w_e \in C_q \setminus C_{q-1} = C^*_q \setminus C^*_{q-1}$ since $t^* < q$, which implies that $w_e \in C^*_{q,s} \setminus C^*_{q-1}$ and so $w_e \in X_0$.

Now fix some $i \in \{1,\ldots,k-1\}$. By construction of the sequence $t_1,\ldots,t_k$ the conditions (i)–(vii) of Lemmas 9 are satisfied with $t = t_i$ and $t' = t_{i+1}$. By (ii), there is a vertex $u \in C^*_{t_{i+1}} \cap C^*_{t_i+1}$ that is active on $(t_{i+1}, t_{i+1}+1)$ and depleted on $(t_i, t_i+1)$. This means that $t_{i+1}+1 \leq d_u \leq t_i$ and $r_u \geq t_i+1$. From these bounds, some vertex $x$ must have been attached to the path with extremity $u$ at time $t = d_u$ in Line 5 of Algorithm 1 and so, again by the algorithm, must be a member of $T$ with $r_x = d_u < t_i+1$.

If we can show that $x \in X_i = C^*_{t_{i+1}+1,t_i} \setminus (C^*_{t_{i+1}} \cup C^*_{t_i+1})$, then the claim follows. Since $r_x < t_i+1$, $x$ is not the last descendant of $u$ on $(t_{i+1}, t_i+1)$ and is not free on $(t_i, t_i+1)$. Hence, by (vi), $x$ is also not free on $(t_{i+1}, t_{i+1}+1)$. Therefore, $t_{i+1} < \ell_x \leq r_x < t_i+1$ and hence $x \notin C^*_{t_{i+1}} \cup C^*_{t_i+1}$. This means that $x \in X_i$ and the claim is proved.

▷ Claim 13. For all distinct $i, j \in \{0,\ldots,k\}$ and every $x \in X_i$ and $y \in X_j$, $x$ and $y$ are in distinct components of $G - S^*$.

It suffices to show that for every $z \in R$ there is $i \in \{0,\ldots,k\}$ such that $t_{i+1}+1 \leq \ell_z \leq r_z \leq t_i$, where we assume that $t_0 = s$ and $t_{k+1} = 0$. Indeed, this implies that for all $i \in \{1,\ldots,k\}$, $C_{t_i} \cap C_{t_i+1} \subseteq S^*$ and the claim then follows from Observation 4.

Suppose, towards a contradiction, that there is some $z \in R$ and some $i \in \{1,\ldots,k\}$ with the property that $\ell_z \leq t_i < r_z$, and assume without loss of generality that $i$ is minimum with respect to these conditions. We first show that $i > 1$. Indeed, if $i = 1$ then $\ell_z \leq t_1 = t^*$ and $r_z > t^*$. But as $z$ is a member of $R$, it follows that $z \in (C_{t^*} \cap C_{t^*+1}) \setminus (V(P_1) \cup V(P_2))$, which is empty since the condition at Line 8 failed at time $t = t^*$ and Algorithm 1 quit at Line 10, a contradiction.

Therefore $i > 1$. Recall in this case that $t_i$ was constructed from $t = t_{i-1}$ by choosing $t_i < t$ such that for $t' = t_i$ the conditions (i)–(vii) of Lemmas 8 and 9 hold. To finish off the proof of the claim, we will show that $\ell_z \leq t_i \leq t_{i-1} < r_z$, giving the final contradiction since, by the minimality of $i$, there is no $z \in R$ with $\ell_z \leq t_{i-1} < r_z$.

We already know that $\ell_z \leq t_i \leq t_{i-1}$. By (vii), there is $x \in V(G^*)$ such that $a_x = t_i$ and $x$ is active on $(t_i, t_i+1)$. By (ii) and (v), $x$ is either active or depleted on $(t_{i-1}, t_{i-1}+1)$. In either case, $r_x \geq t_{i-1}+1 > t_i$. Given that $a_x = t_i$ (that is, $x$ was attached to some path at the $t_i$-th iteration of the for loop of Algorithm 1 at Lines 3–13) and $r_x > t_i$, it follows that $x$ was attached to some path at Line 9 of Algorithm 1. Hence, the right bound of $x$ is less than or equal to that of $z$, which implies $r_z \geq t_{i-1}+1$ and the claim is proved.

From Claims 12 and 13, it follows that $c_T(G-S^*) \geq k+1$ and consequently $c_T(G-S^*) \geq |S^*|$ from the second inequality of Lemma 10. ◀

We are now ready to complete the sketch of the proof of Theorem 5.

**Proof sketch of Theorem 5.** As mentioned earlier, we can assume that $G$ is connected. We can also assume that we can compute in $O(n+m)$ time a clique path $C_1, \ldots, C_s$ of $G$, where each clique is inclusion maximal (by the algorithm of Booth and Lueker [4]), so $s \leq n$. We also compute the left bound and right bound $\ell_v$ and $r_v$ of each vertex $v \in V(G)$, which allows us to find the vertices $w_b$ and $w_e$ in time $\mathcal{O}(n)$. Also in time $\mathcal{O}(n)$, we construct the list $L$ consisting of the right bounds of the elements of $T \setminus \{w_b, w_e\}$ in increasing order.

Next, we run Algorithm 1. At each iteration $t$ of the for loop, Algorithm 1 only needs to decide whether the path under consideration should be extended (at Line 5 and/or Line 9), after which we also need to determine which vertex of $G$ is to be attached to the extremity of this path. Now, given that a path is extended only if the right bound of its extremity (condition at Line 7) or of some vertex of $T$ (condition at Line 5) is precisely $t$, the first computation takes constant time with the list $L$ at hand and hence $\mathcal{O}(n)$ time in total. Moreover, whenever a path is to be extended, we scan the vertices of $C_t \subseteq N_G(v)$. As we never extend more than once a path with the same extremity, this takes in total time $O(\sum_{v \in V} d_G(v)) = O(m)$. Thus, Algorithm 1 runs in $\mathcal{O}(n+m)$ time. Now, if Algorithm 1 finishes at Line 15 and outputs two paths $P_1$ and $P_2$, then we are done by Lemma 6. Otherwise, Algorithm 1 finishes at Line 10, so we work backwards through the algorithm in order to construct the sequence $t_1, \ldots, t_k$ and the set $S^* = \bigcup_{i=1}^{k} S_i$ that certifies a negative answer. With a careful implementation, this can be done in $\mathcal{O}(n+m)$ time as well. ◀

▶ Remark 14. To avoid any misunderstanding, the assumption that the cliques of the clique path of $G$ in the proof of Theorem 5 are maximal is crucial for the running time analysis. But it is also necessary to prove Lemmas 6–11 without this maximality assumption, since the cliques $C_1^*, \ldots, C_s^*$ of the graph $G^*$ (obtained from $G$ by the removal of the set of renounced vertices) are not necessarily maximal. In other words, it is essential to start off with an input graph whose clique path consists of maximal cliques but to also prove statements that concern interval graphs whose clique path may contain non-maximal cliques.

To solve Cycle Segment Cover, we require a folklore observation (see, e.g., [13]).

▶ **Observation 15.** *Let $G$ be a graph, $T \subseteq V(G)$ and $k$ be a positive integer. If $G'$ is obtained from $G$ by adding $k$ universal vertices to $G$, then $\mathsf{sc}_T(G) \leq k$ if and only if $\mathsf{sc}_T(G') \leq 0$.*

Combining this observation with a careful analysis of the running time of the previous algorithm applied to $G'$ instead of $G$, we obtain the following result.

▶ **Theorem 16.** *There is an algorithm that, given an instance $(G, T, r)$ of* Cycle Segment Cover*, where $G$ is an interval graph and $T$ is not a 2-clique, finds either a cycle or a family of at most $r$ paths that cover $T$ or a $T$-separator $S^*$ with $c_T(G - S^*) - |S^*| > r$ that certifies a no-answer in time $\mathcal{O}(n+m)$.*

## 3.2 $k$-Cyclability for interval graphs

In this subsection we prove Theorem 2 for interval graphs. From Theorem 5 and from the definition of $\mathsf{sc}^k(G)$, an interval graph $G$ with at least three vertices is $k$-cyclable if and only if $\mathsf{sc}^k(G) \leq 0$. So to solve $k$-Cyclability on interval graphs, it is sufficient to construct a polynomial algorithm that computes the $k$-scattering number of $G$ for any $k \leq n-1$. The only remaining task will consist in finding the largest integer $k$ such that $\mathsf{sc}^k(G) \leq 0$. We use the following lemma.

▶ **Lemma 17.** *Let $G$ be an interval graph, let $C_1, \ldots, C_s$ be a clique path of $G$, where $C_1, \ldots, C_s$ are pairwise-distinct maximal cliques of $G$, and let $S$ be a separator of $G$. Then,*

$$\text{there exist } 1 \leq t_1 < \ldots < t_r < s \text{ such that } S' = \bigcup_{i=1}^{r} (C_{t_i} \cap C_{t_i+1}) \tag{4}$$

*and $S' \subseteq S$ and $S'$ is a separator of $G$ such that $c(G - S') \geq c(G - S)$.*

Informally, the above lemma states that, in computing the $k$-scattering number, one can restrict their attention to a subset of separators, namely these separators that satisfy (4), which we call *canonical separators*. Thus, Lemma 17 implies that, for an interval graph $G$, $\mathsf{sc}^k(G)$ can be equivalently defined as

$$\mathsf{sc}^k(G) = \max\{c(G - S) - |S| \mid S \text{ is a canonical separator of } G \text{ s.t. } |S| \leq k - 1\}. \tag{5}$$

To solve $k$-Cyclability, our algorithm computes canonical separators via a dynamic programming scheme on the given clique path $C_1, \ldots, C_s$ of a non-complete interval graph.

▶ **Theorem 18.** *For a non-complete interval graph $G$, one can solve $k$-Cyclability and compute the scattering numbers $\mathsf{sc}^k(G)$ for all $k \in \{1, \ldots, n-1\}$ in time $\mathcal{O}(n^3)$.*

## 4 Bipartite permutation graphs

In this section we briefly sketch our results for Cycle Segment Cover and $k$-Cyclability on bipartite permutation graphs. Let $G = (V_1, V_2, E)$ a bipartite graph. Let $\sigma_1 = \langle u_1, \ldots, u_p \rangle$ and $\sigma_2 = \langle v_1, \ldots, v_q \rangle$ be orderings of, respectively, $V_1$ and $V_2$. It is said that $(\sigma_1, \sigma_2)$ is a *strong ordering* of $G$ if for every $1 \leq i < i' \leq p$ and $1 \leq j' < j \leq q$, if $u_i v_j, u_{i'} v_{j'} \in E(G)$, then $u_i v_{j'}, u_{i'} v_j \in E(G)$. Spinrad, Brandstädt and Stewart [30] showed that (1) a bipartite graph is a permutation graph if and only if it has a strong ordering and that (2) in any such ordering, for every $v \in V(G)$, the vertices of $N_G(v)$ are consecutive either in $\sigma_1$ or in $\sigma_2$. Using this and other results from [30], we prove the following theorem.

▶ **Theorem 19.** *There is an algorithm that, given an instance $(G, T, r)$ of* Cycle Segment Cover*, where $G$ is a bipartite permutation graph and $T$ is not a 2-clique, finds either a cycle or a family of at most $r$ paths that cover $T$ or a $T$-separator $S^*$ with $c_T(G - S^*) - |S^*| > r$ that certifies a no-answer in time $\mathcal{O}(n + m)$.*

To prove Theorem 19, we first establish the stronger fact that in a connected bipartite permutation graph $G$, if $T \subseteq V(G)$ is not cyclable (and not a 2-clique) then there is a $T$-separator $S^*$ with the property $c_T(G - S^*) - |S^*| > 0$ such that $S^*$ is formed by consecutive vertices with respect to the strong ordering of either $V_1$ or $V_2$. We use this fact to construct a certifying algorithm for Terminal Cyclability and then, finally, generalize this algorithm to Cycle Segment Cover. Note that, unlike for interval graphs, we cannot use an analogue of Observation 15 for bipartite permutation graphs because the class of bipartite permutation graphs is not closed under adding a universal vertex.

For $k$-Cyclability, we first show that a connected bipartite permutation graph $G = (V_1, V_2, E)$ with at least three vertices is $k$-cyclable if and only if all the subsets $T$ satisfying the following property are cyclable: *$T \subseteq V_i$ for some $i \in \{1, 2\}$, the vertices of $T$ are consecutive in $\sigma_i$ and $|T| = \min\{|V_i|, k\}$*. Using this equivalence, we test the $k$-cyclability of $G$ by running the $T$-cyclability algorithm for each subset $T$ of terminals satisfying the aforementioned property. As the number of such subsets $T$ is $O(n)$, we obtain the following theorem, which implies Theorem 2 for bipartite permutation graphs (notice the slightly better running time, namely $\mathcal{O}(nm)$ instead of $\mathcal{O}(n^3)$).

▶ **Theorem 20.** *$k$-Cyclability can be solved in time $\mathcal{O}(nm)$ on bipartite permutation graphs.*

Notice that, unlike our approach for solving $k$-Cyclability on interval graphs, we solve $k$-Cyclability on bipartite permutation graphs $G$ without determining $\mathsf{sc}^k(G)$.

## 5 Cographs

In this section, we briefly sketch our results for Cycle Segment Cover and $k$-Cyclability on cographs.

Let $G_1$ and $G_2$ be two vertex-disjoint graphs. The *union* operation creates the *disjoint union* $G_1 + G_2$ of $G_1$ and $G_2$, that is, the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$. The *join* operation adds an edge between every vertex of $G_1$ and every vertex of $G_2$. Cographs can be characterized as those graphs that can be generated from $K_1$ by a sequence of join and union operations. This gives each cograph $G$ a nice tree representation, called the *cotree* of $G$, whose leaves are the vertices of $G$ and whose internal nodes represent the join and union operations used in the construction of $G$.

Our algorithm for Cycle Segment Cover of a cograph $G$ is built using dynamic programming bottom-up along the cotree of $G$.

▶ **Theorem 21.** *There is an algorithm that, given an instance $(G, T, r)$ of Cycle Segment Cover where $G$ is a cograph given by its cotree and $T$ is not a 2-clique, finds either a $T$-cycle segment cover of size at most $r$ or a $T$-separator with $c_T(G - S^*) - |S^*| > r$ that certifies a no-answer in $\mathcal{O}(n)$ time.*

We solve $k$-Cyclability for cographs just as we did for interval graphs by determining all the scattering numbers $\mathsf{sc}^k(G)$ for $k \in \{1, \ldots, n\}$, again using a bottom-up dynamic programming scheme along the cotree of $G$.

▶ **Theorem 22.** *For a non-complete cograph $G$, the scattering numbers $\mathsf{sc}^k(G)$ for all $k \in \{1, \ldots, n\}$ can be computed and $k$-Cyclability can be solved in time $\mathcal{O}(n^3)$.*

## 6 Conclusion

In summary, we design certifying linear-time algorithms to solve Cycle Segment Cover, which is a generalization of Hamiltonian Cycle, for interval graphs, bipartite permutation graphs and cographs. We also use these results to show that $k$-Cyclability as well can be solved in polynomial time when restricted to these graph classes.

A natural open question is to consider the aforementioned problems for other graph classes. In particular, what can be said about the class of cocomparability graphs (see [6, 21] for the formal definition and properties of this class)? For instance, it is proved by Deogun, Kratsch and Steiner [15] that a cocomparability graph $G$ with at least three vertices has a Hamiltonian cycle if and only if $\mathsf{sc}(G) \leq 0$. They also proved that the set of vertices of $G$ can be covered by at most $k$ vertex-disjoint paths if and only if $\mathsf{sc}(G) \leq k$. This indicates that the class of cocomparability graphs is a natural candidate for Cycle Segment Cover and $k$-Cyclability. Still, we do not see how to extend the results of [15] to our settings.

Another interesting question is about the complexity of Terminal Cyclability. It is easy to see that the problem is in $\Pi_2^{\mathsf{P}}$. Golovach et al. conjectured in [20] that Terminal Cyclability is $\Pi_2^{\mathsf{P}}$-complete. The conjecture is still open.

## References


1 Eyad Alkassar, Sascha Böhme, Kurt Mehlhorn, and Christine Rizkallah. A Framework for the Verification of Certifying Computations. *J. Autom. Reasoning*, 52(3):241–273, 2014. doi:10.1007/s10817-013-9289-2.

2 Douglas Bauer, Hajo Broersma, and Edward F. Schmeichel. Toughness in Graphs - A Survey. *Graphs and Combinatorics*, 22(1):1–35, 2006. doi:10.1007/s00373-006-0649-0.

3 Andreas Björklund, Thore Husfeldt, and Nina Taslaman. Shortest cycle through specified elements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1747–1753. SIAM, 2012. URL: http://portal.acm.org/citation.cfm?id=2095255&CFID=63838676&CFTOKEN=79617016, doi:10.1137/1.9781611973099.139.

4 Kellogg S. Booth and George S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

5 Andreas Brandstädt and Dieter Kratsch. On the restriction of some NP-complete graph problems to permutation graphs. In *Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985*, volume 199 of *Lecture Notes in Computer Science*, pages 53–62. Springer, 1985. doi:10.1007/BFb0028791.

6 Andreas Brandstadt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999. doi:10.1137/1.9780898719796.

7 Hajo Broersma, Jirí Fiala, Petr A. Golovach, Tomás Kaiser, Daniël Paulusma, and Andrzej Proskurowski. Linear-Time Algorithms for Scattering Number and Hamilton-Connectivity of Interval Graphs. *Journal of Graph Theory*, 79(4):282–299, 2015. doi:10.1002/jgt.21832.

8 Maw-Shang Chang, Sheng-Lung Peng, and Jenn-Liang Liaw. Deferred-query: An efficient approach for some problems on interval graphs. *Networks*, 34(1):1–10, 1999. doi:10.1002/(SICI)1097-0037(199908)34:1<1::AID-NET1>3.0.CO;2-C.

9 Vasek Chvátal. Tough graphs and hamiltonian circuits. *Discrete Mathematics*, 5(3):215–228, 1973. doi:10.1016/0012-365X(73)90138-6.

10 Derek G. Corneil, Barnaby Dalton, and Michel Habib. LDFS-Based Certifying Algorithm for the Minimum Path Cover Problem on Cocomparability Graphs. *SIAM J. Comput.*, 42(3):792–807, 2013. doi:10.1137/11083856X.

11 Derek G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981. doi:10.1016/0166-218X(81)90013-5.

12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

13 Peter Damaschke. Paths in interval graphs and circular arc graphs. *Discrete Mathematics*, 112(1-3):49–64, 1993. doi:10.1016/0012-365X(93)90223-G.

14 Peter Damaschke, Jitender S. Deogun, Dieter Kratsch, and George Steiner. Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm. *Order*, 8(4):383–391, 1991. doi:10.1007/BF00571188.

15 Jitender S. Deogun, Dieter Kratsch, and George Steiner. 1-Tough cocomparability graphs are hamiltonian. *Discrete Mathematics*, 170(1-3):99–106, 1997. doi:10.1016/0012-365X(95)00359-5.

16 Jitender S. Deogun and George Steiner. Polynomial Algorithms for Hamiltonian Cycle in Cocomparability Graphs. *SIAM J. Comput.*, 23(3):520–552, 1994. doi:10.1137/S0097539791200375.

17 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

18 Gabriel Andrew Dirac. In abstrakten Graphen vorhandene vollständige 4-Graphen und ihre Unterteilungen. *Math. Nachr.*, 22:61–85, 1960. doi:10.1002/mana.19600220107.

19 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

20 Petr A. Golovach, Marcin Kaminski, Spyridon Maniatis, and Dimitrios M. Thilikos. The Parameterized Complexity of Graph Cyclability. *SIAM J. Discrete Math.*, 31(1):511–541, 2017. doi:10.1137/141000014.

21 M. C. Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

22 Ronald J. Gould. A look at cycles containing specified elements of a graph. *Discrete Mathematics*, 309(21):6299–6311, 2009. doi:10.1016/j.disc.2008.04.017.

23 Ruo-Wei Hung and Maw-Shang Chang. Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs' . *Theor. Comput. Sci.*, 341(1-3):411–440, 2005. doi:10.1016/j.tcs.2005.04.009.

24 Ruo-Wei Hung and Maw-Shang Chang. Linear-time certifying algorithms for the path cover and Hamiltonian cycle problems on interval graphs. *Appl. Math. Lett.*, 24(5):648–652, 2011. doi:10.1016/j.aml.2010.11.030.

25 H. A. Jung. On a class of posets and the corresponding comparability graphs. *J. Comb. Theory, Ser. B*, 24(2):125–133, 1978. doi:10.1016/0095-8956(78)90013-8.

26 J. Mark Keil. Finding Hamiltonian Circuits in Interval Graphs. *Inf. Process. Lett.*, 20(4):201–206, 1985. doi:10.1016/0020-0190(85)90050-X.

27 Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011. doi:10.1016/j.cosrev.2010.09.009.

28 Haiko Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156(1-3):291–298, 1996. doi:10.1016/0012-365X(95)00057-4.

29 Neil Robertson and Paul D. Seymour. Graph Minors .XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

30 Jeremy P. Spinrad, Andreas Brandstädt, and Lorna Stewart. Bipartite permutation graphs. *Discrete Applied Mathematics*, 18(3):279–292, 1987. doi:10.1016/S0166-218X(87)80003-3.