Linear Recognition of Almost (Unit) Interval Graphs*

Yixin Cao

Abstract

Give a graph class $\mathcal{G}$ and a nonnegative integer $k$, we use $\mathcal{G} + kv$, $\mathcal{G} + ke$, and $\mathcal{G} - ke$ to denote the classes of graphs that can be obtained from some graph in $\mathcal{G}$ by adding $k$ vertices, adding $k$ edges, and deleting $k$ edges, respectively. They are called almost (unit) interval graphs if $\mathcal{G}$ is the class of (unit) interval graphs. Almost (unit) interval graphs are well motivated from computational biology, where the data ought to be represented by a (unit) interval graph while we can only expect an almost (unit) interval graph for the best. For any fixed $k$, we give linear-time algorithms for recognizing all these classes, and in the case of membership, our algorithms provide also a specific (unit) interval graph as evidence.

When $k$ is part of the input, all the recognition problems are NP-complete. Our results imply that all of them are fixed-parameter tractable parameterized by $k$, thereby resolving the long-standing open problem on the parameterized complexity of recognizing (unit) interval+ke, first asked by Bodlaender et al. [Comput. Appl. Biosci., 11(1):49–57, 1995]. Moreover, our algorithms for recognizing (unit-)interval+kv and (unit-)interval−ke have single-exponential dependence on $k$ and linear dependence on the graph size, which significantly improve all previous algorithms for recognizing the same classes. In particular, we show that: (n and m stand for the numbers of vertices and edges respectively in the input graph)

- interval−ke can be recognized in time $O(6^k \cdot (n + m))$, improved from $O(k^{2k} \cdot n^3 m)$ [Heggernes et al., STOC 2007];
- unit-interval−ke can be recognized in time $O(4^k \cdot (n + m))$, improved from $O(16^k \cdot (m + n))$ [Kaplan et al., FOCS 1994];
- interval+kv can be recognized in time $O(8^k \cdot (n + m))$, improved from $O(10^k \cdot n^9)$ [Cao and Marx, SODA 2014]; and
- unit-interval+kv can be recognized in time $O(6^k \cdot (n + m))$, improved from $O(6^k \cdot n^6)$ [Villanger, IPEC 2010].

These problems have natural optimization versions, which are known as graph modification problems. For those related to interval graphs, we show that under certain condition, there always exist optimum solutions that preserve all modules of the input graph. Another important ingredient of our algorithms is combinatorial and algorithmic characterizations of graphs free of small non-interval graphs. These studies might be of their own interest.

Keywords: (unit) interval graphs, locally interval graphs, forbidden induced subgraphs, asteroidal triples, holes, modular decomposition, quotient graphs, prime graphs, interval models, (normal Helly) circular-arc models, (normal Helly) circular-arc graphs, clique decompositions, olive-ring decompositions, physical mapping, profile minimization (sparse matrix computation), graph modification problems, parameterized computation, (unit) interval vertex deletion, (unit) interval edge deletion, (unit) interval completion, maximum induced interval subgraphs, maximum spanning interval subgraphs, minimum interval supergraphs.

---

* Dedicated to Jianer Chen on the occasion of his 60th birthday.
† Institute for Computer Science and Control, Hungarian Academy of Sciences, (MTA SZTAKI) Hungary. Email: yixin@sztaki.hu. Supported by the European Research Council (ERC) under the grant 280152 and the Hungarian Scientific Research Fund (OTKA) under the grant NK105645.
1 Introduction

All graphs discussed in this paper shall always be undirected and simple. A graph is an interval graph if its vertices can be assigned to intervals on the real line such that there is an edge between two vertices if and only if their corresponding intervals intersect. The study of interval graphs has been closely associated with computational biology [4, 87]. For example, the physical mapping of DNA asks for reconstructing the relative positions of clones along the target DNA based on their pairwise overlap information [63, 2]. These data are naturally formulated as a graph, where each clone is a vertex, and two clones are adjacent if and only if they overlap [87, 92, 52], hence an interval graph. A wealth of literature has been devoted to algorithms on interval graphs, which include a series of linear-time recognition algorithms [10, 60, 49, 42, 22]. Ironically, however, they are never used as they are intended to be. Computational biologists never need to roll up their sleeves and feed their data into any recognition algorithm before claiming the answer is “NO” with full confidence, i.e., their data would not give an interval graph though they ought to. The reason is that biological data, obtained by mainly experimental methods, are destined to be flawed.

Fortunately, computational biologists are also confident that their data, though not perfect, are of reasonably good quality. That is, there are only few errors [63]. This leads us naturally to consider graphs that are not interval graphs, but close to one in some sense. In general, we call such a graph as an almost interval graph: it is unnecessarily an interval graph itself, but can be obtained from an interval graph by small amount of modifications. It is observed that in some models of physical mapping, all clones might have (roughly) the same length [38]. The graph representing these clones is then expected to be a unit interval graph, which is a special interval graph and has interval models where every interval has the same length. Therefore, we consider also almost unit interval graphs, which are defined analogously.

Different applications are afflicted with different types of errors. For example, there might be false-positive overlaps, false-negative overlaps, or outliers. We can accordingly define different measures for closeness. Let $\mathcal{G}$ be a hereditary graph class, which is closed under taking induced subgraphs, and let $k$ be a nonnegative integer. We use $\mathcal{G} + kv, \mathcal{G} + ke$, and $\mathcal{G} - ke$ to denote the classes of graphs that can be obtained from some graph $G \in \mathcal{G}$ by adding at most $k$ vertices, adding at most $k$ edges, and deleting at most $k$ edges, respectively. Recall that (unit) interval graphs are hereditary.

The first task is of course to efficiently decide whether a graph is an almost (unit) interval graph or not, and more importantly, identify an object (unit) interval graph if one exists. Computationally, finding an object (unit) interval graph is equivalent to pinpointing the few but crucial errors in the data. There is a trivial algorithm that finds them in polynomial time: given a graph $G$ on $n$ vertices, we can in $n^{O(k)}$ time try every subset of $k$ vertices, edges, or missing edges of $G$. Such an algorithm is nevertheless inefficient even for very small $k$, as $n$ is assumed to be large. The main result of this paper is linear-time recognition algorithms for all six classes of almost (unit) interval graphs.

**Theorem 1.1.** Let $\mathcal{G}$ be the class of interval graphs or unit interval graphs, and let $k$ be any fixed nonnegative integer. Given a graph $G$ on $n$ vertices and $m$ edges, the membership of $G$ in $\mathcal{G} + kv, \mathcal{G} + ke$, and $\mathcal{G} - ke$ can be decided in $O(n + m)$ time. Moreover, in case of the membership is asserted, an object graph in $\mathcal{G}$ can be produced in the same time.

Thm. 1.1 extends all linear-time recognition algorithms for interval graphs [10, 60, 49, 42, 22] and unit interval graphs [20, 25]. We point out that only unit-interval−$ke$ was previously known to be linear-time recognizable [50].

In all algorithms of Thm. 1.1, the constants hidden by big-Oh rely on $k$. Since all problems are NP-hard when $k$, instead of being a constant, is a part of the input [65, 52, 38], the dependence on $k$ is necessarily super-polynomial (assuming $P \neq NP$). Observing that the linear dependence on the graph size is already optimum, we would like to minimize the factor of $k$. We are thus brought into the framework of parameterized computation. Recall that a problem, associated with some parameter, is fixed-parameter tractable (FPT) if it admits a polynomial-time algorithm where the exponent on $n$ is a global constant independent on the parameter [29]. From the lens of parameterized computation, the recognition of almost (unit) interval graphs is conventionally defined as graph modification problems, where the parameter is $k$, and the task is to transform a graph to a (unit) interval graph by at most $k$ modifications [14]. For the classes $\mathcal{G} + kv, \mathcal{G} + ke$,

---

Footnote: Here we use “at most” instead of “precisely” for both practical and theoretical reasons. Practically, this formulation is more natural for aforementioned applications, where less modifications are preferred. Theoretically, it allows all classes fully contain $\mathcal{G}$ itself; in particular, we allow (unit-)interval−$ke$ and (unit-)interval−$ke$ to contain graphs with no edge and cliques, respectively. As a matter of fact, one can show that except the trivial cases (i.e., the input graph has less than $k$ vertices, $k$ edges, or $k$ missing edges), if a graph can be made from an interval graph $G'$ by $k'$ operations, where $k' < k$, then we can also obtain it from another interval graph $G$ by exactly $k$ operations of the same type.
and \( \mathcal{F} \), the modifications are vertex deletion, edge deletion, and completion (i.e., edge addition) respectively, which are the most commonly considered on hereditary graph classes. The parameterized problems are thus named (UNIT) INTERVAL VERTEX DELETION, (UNIT) INTERVAL EDGE DELETION, and (UNIT) INTERVAL COMPLETION, and our algorithms can then be more specifically stated as:

**Theorem 1.2.** Given a graph on \( n \) vertices and \( m \) edges and nonnegative parameter \( k \), the problems UNIT INTERVAL VERTEX DELETION, UNIT INTERVAL EDGE DELETION, UNIT INTERVAL COMPLETION, INTERVAL VERTEX DELETION, INTERVAL EDGE DELETION, and INTERVAL COMPLETION can be solved in time \( O(6^k \cdot (n + m)) \), \( O(4^k \cdot (n + m)) \), \( O(8^k \cdot (n + m)) \), \( k^{O(k)} \cdot (n + m) \), and \( O(6^k \cdot (n + m)) \), respectively.

Therefore, both UNIT INTERVAL EDGE DELETION and INTERVAL EDGE DELETION are FPT. This resolves a long-standing open problem first asked by Bodlaender et al. \[8\]. Further, our algorithms for INTERVAL VERTEX DELETION and INTERVAL COMPLETION significantly improve the \( O(k^{2k} \cdot n^3 m) \)-time algorithm of Heggernes et al. \[85\] and the \( O(10^k \cdot n^9) \)-time algorithm of Cao and Marx \[17\], respectively; and our algorithms for UNIT INTERVAL VERTEX DELETION and UNIT INTERVAL COMPLETION improve the \( O(6^k \cdot n^9) \)-time algorithm of van’t Hof and Villanger \[85\] and the \( O(16^k \cdot (n + m)) \)-time algorithm of Kaplan et al. \[50\], respectively.

We feel obliged to point out that computational biologists cannot claim all credit for the discovery and further study of interval graphs. Independent of \[4\], Hajós \[45\] formulated the class of interval graphs out of nothing but coffee. Since its inception in 1950s, its natural structure earns itself a position in many other applications, among which the most cited ones include jobs scheduling in industrial engineering \[3\], temporal reasoning \[41\], and seriation in archeology \[55\]. All these applications involve some temporal structure which is understandable: before the final invention of time traveling vehicles, a graph representing a temporal structure has to be an interval graph, or a unit interval graph when the time slots are of the same size.

With errors involved, almost (unit) interval graphs arise naturally.

### 1.1 Our major results

We state here the major results of this paper (besides Thms.\[1.1\] and \[1.2\]) that are of independent interest.

A graph \( G \) is given by its vertex set \( V(G) \) and edge set \( E(G) \). Its order \( |G| \) and size \( ||G|| \) are defined to be the cardinalities of \( V(G) \) and \( E(G) \) respectively. We assume without loss of generality that \( G \) is connected and nontrivial (it contains at least two vertices); thus \( |G| = O(||G||) \). Technical terms used in this section either are defined later in the paper or can be found in standard textbooks, e.g., \[26\] 39\).

One important ingredient of our algorithms is a comprehensive study of the following graph class, which properly contains all interval graphs. Net, sun, rising sun, long claw, and whipping top are depicted in Fig.\[1\]

A **hole** is a cycle induced by 4 or more vertices, and in particular, a hole of \( d \) vertices is called a \( d \)-hole.

\textbf{Definition 1.} Locally interval graphs are defined by forbidding nets, suns, rising suns, long claws, whipping tops, and 4- and 5-holes.

We will use \( \mathcal{F}_1 \), \( \mathcal{F}_{UI} \), and \( \mathcal{F}_{L1} \) to denote the sets of minimal forbidden induced subgraphs for interval graphs, unit interval graphs, and locally interval graphs, respectively.

---

![Figure 1: Minimal chordal asteroidal witnesses (terminals are marked as squares).](image-url)
 Recall that a subset M of vertices forms a module if every vertex in M has the same neighbors outside M. Every graph G uniquely defines a quotient graph G_Q (formal definition in Section 3.2), which is either a clique or a prime graph (i.e., containing only trivial modules) when G is connected. A vertex v is universal in G if N[v] = V(G). A vertex v is simplicial in G if N[v] induces a clique of G.

**Proposition 1.3.** Let $\mathcal{G}$ be the class of interval graphs or locally interval graphs. A graph G that contains no universal vertex is in $\mathcal{G}$ if and only if

1. the quotient graph G_Q of G is in $\mathcal{G}$ but not a clique;
2. every module that corresponds to a simplicial vertex of G_Q induces a graph in $\mathcal{G}$; and
3. every module that corresponds to a non-simplicial vertex of G_Q induces a clique.

In other words, any (locally) interval graph can be constructed from a single vertex by recursively replacing non-simplicial vertices by cliques and simplicial vertices by prime (locally) interval graphs, and then adding universal vertices.

Let $\mathcal{K}$ be a connected graph whose nodes, called bags, are the set of all maximal cliques of G. We say that $\mathcal{K}$ is a clique decomposition of G if for any vertex $v \in V(G)$, the set of maximal cliques containing $v$ induces a connected subgraph of $\mathcal{K}$. A caterpillar is a tree that consists of a central path and all other vertices are leaves connected to it. An olive ring is a uni-cyclic graph that consists of a hole and all other vertices are pendant—having degree 1—and connected to this hole. The deletion of any edge from the main cycle of an olive ring results in a caterpillar.

**Theorem 1.4.** A prime locally interval graph G has a clique decomposition that is either a caterpillar when it is chordal; or an olive ring otherwise. This decomposition can be constructed in $O(||G||)$ time.

Indeed, given a prime graph G that does not have a caterpillar or olive-ring decomposition, our algorithm will identify a subgraph of G in $\mathcal{F}_{1,1}$. The following statement is thus stronger than Thm. 1.4 and implies it.

**Theorem 1.5.** For a prime graph G, we can in $O(||G||)$ time either build an olive-ring/caterpillar decomposition for G or find a subgraph of G in $\mathcal{F}_{1,1}$.

An induced interval subgraph of G is an interval subgraph induced by a set $U \subseteq V(G)$ of vertices. It is maximum if there is no vertex set $U' \subseteq V(G)$ with $|U'| > |U|$ such that G[U'] is an interval graph. The induced interval subgraph G[U] is also denoted by $G - V_-$, where $V_- = V(G) \setminus U$. An interval graph G (resp., $G'$) is called a spanning interval subgraph (resp., an interval supergraph) of G if it has the same vertex set as G and $E(G) \subseteq E(G)$ (resp., $E(G) \subseteq E(G')$), and is called maximum (resp., minimum) if there is no spanning interval subgraph $G'$ (resp., interval supergraph $\hat{G}'$) of G satisfying $||G'|| > ||G||$ (resp., $||\hat{G}'|| < ||G||$). The following theorems relate maximum interval subgraphs and minimum interval supergraphs of a graph to its modules; they hold regardless of k.

**Theorem 1.6.** Let $G - V_-$ be a maximum induced interval subgraph of graph G. For any module M of G such that $M \not\subseteq V_-$, the set $M \setminus V_-$ is a module of $G - V_-$, and if G is 4-hole-free, then replacing $G[M \setminus V_-]$ by any maximum induced interval subgraph of G[M] in $G - V_-$ gives a maximum induced interval subgraph of G.

**Theorem 1.7.** Let G be a 4-hole-free graph. There is a maximum spanning interval subgraph G of G such that the following hold for every module M of G: i) M is a module of $\hat{G}$; and ii) replacing $G[M]$ by any maximum spanning interval subgraph of G[M] in $\hat{G}$ gives a maximum spanning interval subgraph of G.

**Theorem 1.8.** For any graph G, there is a minimum interval supergraph $\hat{G}$ of G such that the following hold for every module M of G: i) M is a module of $\hat{G}$; and ii) if $\hat{G}[M]$ is not a clique, then replacing G[M] by any minimum interval supergraph of G[M] in $\hat{G}$ gives a minimum interval supergraph of G.

Note that Thm. 1.8 applies to general graphs, while Thms. 1.6 and 1.7 require the graph to be 4-hole-free. Thm. 1.6 applies to every maximum induced interval subgraph, while Thms. 1.7 and 1.8 apply to only some of maximum spanning interval subgraphs or minimum interval supergraphs. Thms. 1.6 and 1.7 allow us to use divide and conquer on these problems.

In addition to the above listed concrete results, our algorithms also suggest a meta approach for designing fixed-parameter algorithms for graph modification problems, which extends the result of Cai [14]: If the object graph class can be characterized by a set of forbidden induced subgraphs of which only a finite number are not prime, then we may break them first and then work in a top-down way along the modular decomposition tree of the resulted graph. This enables us to concentrate on prime graphs and use their structural properties.
1.2 Related graph classes and modules

A graph is chordal if it contains no hole. It is known that a graph is an interval graph if and only if it is chordal and does not contain a structure called asteroidal triple (AT for short), i.e., three vertices such that each pair of them is connected by a path avoiding neighbors of the third one [64]. Lekkerkerker and Boland [64] went further to list all minimal chordal graphs that contain an AT. These graphs, reproduced in Fig. 1, are called chordal asteroidal witnesses (CAWs for short). There are infinitely many of distinct CAWs, and $\mathcal{F}_1$ comprises all holes and CAWs. By contrast, $\mathcal{F}_{UI}$ is far simpler: apart from holes, $\mathcal{F}_{UI}$ contains only three small chordal subgraphs, namely, sun, net, and claw (i.e., a graph on four vertices with all three edges incident to a single vertex) [88]. We point out that except sun and net, every other CAW contains a claw. Interestingly, the class of unit interval graphs coincides with proper interval graphs, which are those interval graphs that have a representation with no interval containing another one.

A set of intervals on the real line representing an interval graph is called an interval model for it. This suggests yet another way to generalize interval graphs, i.e., to use a circle and arcs in place of the real line and intervals, and then a pair of vertices is adjacent if and only if their corresponding arcs intersect. Such a model is a circular-arc model, and a graph having a circular-arc model is a circular-arc graph. Compared to interval graphs which represent temporal reasoning, circular-arc graphs are natural models for periodic activities. Trivially, every interval graph is a circular-arc graph, and this containment is again proper. In a general circular-arc model, some pathetic behavior is possible, which makes the class of circular-arc graphs far less understood and harder to manipulate than interval graphs. Most notably, two arcs might intersect at both ends, and three or more arcs might pairwise intersect but contain no common point. A circular-arc model that is free of these two patterns is normal and Helly, which are known to be those without three or less arcs covering the whole circle [66]. A graph that has such a model is a normal Helly circular-arc graph.

Although all classes of locally interval graphs, chordal graphs, AT-free graphs, and normal Helly circular-arc graphs contain interval graphs as a proper subset, they are incomparable to each other. This fact can be evidenced by the following non-interval graphs: a $t$-interval model (Fig. 1) and another vertex adjacent to $l$ and $r$ only; a long claw; $K_{2,3}$ (a complete bipartite graph on two and three vertices); and a 5-hole and another vertex that is adjacent to two consecutive vertices in the hole only. They belong to (prime) locally interval graphs, chordal graphs, AT-free graphs, and normal Helly circular-arc graphs, respectively, but no others.

All graph classes mentioned in this paper are hereditary. For a comprehensive treatment and for references to the extensive literature on various graph classes, one may refer to Golumbic [39], Brandstädt et al. [11], and Spinrad [79].

The development of the concept of modular decomposition is in symbiosis with graph classes. It was originally proposed by Gallai [34] in his study of comparability graphs. Since it is well known that (1) the complement of an interval graph is a comparability graph; and (2) a module of a graph is also a module of its complement graph, it is natural to apply modular decomposition to interval graphs. The relationship between modular decomposition and interval graphs has been built ever since (see, e.g., [69]). Hsu [48] characterized prime interval graphs, and used this characterization to develop a linear-time recognition algorithm for interval graphs [49], which is arguably the simplest among all known recognition algorithms for interval graphs. In general, many problems on a graph can be solved in a top-down or bottom-up approach along with its modular decomposition tree [69, 44].

1.3 (Unit) interval vertex/edge deletion

The aforementioned physical mapping of DNA is a central problem in computational biology [63, 2]. In a utopia where experimental data are perfect, they should define an interval graph. Then the problem is equivalent to constructing an interval model for the graph, which can be done in linear time [10, 60, 49, 42, 22]. In the real world we live, however, data are always inconsistent and contaminated by a few but crucial errors, which have to be detected and fixed. In particular, on the detection of false-positive errors that correspond to fake edges, i.e., (unit-)interval + ke, Goldberg et al. [38] formulated the minimum (unit) interval edge deletion problems and showed their NP-completeness.

Likewise, the deletion of vertices can be used to formulate the detection of outliers, i.e., (unit-)interval + kv. The NP-completeness of minimum (unit) interval vertex deletion problems follow from a classical result of Lewis and Yannakakis [65]. Let us point out that, when the object graph class $\mathcal{G}$ is hereditary, the vertex deletion version can be considered as the most robust variant, which in some sense encompasses both edge addition and edge deletion: if $G$ can be made a member of $\mathcal{G}$ by $k_-$ edge deletions and $k_+$ edge additions, then it can be also made a member of $\mathcal{G}$ by deleting at most $k_-$ vertices (e.g., by removing one endvertex of each added/deleted edge). In other words, the graph class $\mathcal{G} + k_-$ contains both classes $\mathcal{G} + ke$ and $\mathcal{G} + ke$.
On the other hand, \( G + ke \) and \( G - ke \) are incomparable in general, e.g., a long hole is in interval \(+1e\) and \( K_{2,3} \) is in interval \(-1e\) but not the other way.

Solving the minimum (unit) interval vertex deletion problem and (unit) interval edge deletion problem is equivalent to finding the maximum induced (unit) interval subgraph and the maximum spanning (unit) interval subgraph respectively. Pálvölgyi [74] and Erdős et al. [31] studied the maximum induced interval subgraph problem and maximum spanning interval subgraph problem from the aspect of extremal combinatorics. Bliznets et al. [7] considered exact algorithms for the maximum induced interval subgraph problem.

In light of the importance of interval graphs, it is not surprising that some natural combinatorial problems can be formulated as, or computationally reduced to the interval deletion problem. For instance, Narayanaswamy and Subashini [71] recently solved the maximum consecutive ones sub-matrix problem and the minimum convex bipartite deletion problem by a reduction to minimum interval vertex deletion.

### 1.4 (Unit) interval completion

The minimum interval completion problem is NP-hard [53, 91], and the proof in [91] applies to minimum unit interval completion problem. Besides computational biology, the most important application of the minimum interval completion problem should be sparse matrix computations [80]. The profile method is an extension of the bandwidth method [77, 75], and their purpose is to minimize the storage during Gaussian elimination for a symmetric sparse matrix. Both methods attempt to reorder the rows and columns of the input matrix such that all elimination are limited within a band or an envelop around the main diagonal, while all entries outside are always zeroes during the whole computation. Therefore, we only need to store the elements in the band or envelop, which are accordingly called the bandwidth and profile. We refer to [37] for more background. Rose [77] correlated bandwidth with graphs. Tarjan [80] showed that a matrix has a reordering such that its profile coincides with non-zero entries if and only if it corresponds to an interval graph, and finding the minimum profile is equivalent to solving the minimum interval completion problem.

A very similar problem is the minimum pathwidth problem, which also asks for an interval supergraph \( \hat{G} \) of \( G \) but the objective is to minimize the size of the maximum clique in \( \hat{G} \). This problem was also known to be NP-hard [54]. In light of the NP-completeness of both problems, people turn to finding minimal interval completions, which can be viewed as a relaxation of both of them. Ohtsuki et al. [73] designed an algorithm that finds a minimal interval completion in \( O(|G| \cdot ||G||) \) time. Very recently, Crespelle and Todinca [23] proposed an improved algorithm that runs in \( O(|G|^2) \) time. This is the best known, and it remains open to develop a linear-time algorithm for finding a minimal interval completion. See also Heggernes et al. [46] for a characterization of minimal interval completions. We point out that a minimal interval completion can be arbitrarily far from a minimum one, e.g., see the graph and its minimal interval completion in Fig. 2, while the minimum completion needs one single edge, \( uv \).

![Figure 2: A non-interval graph and a minimal interval completion (dotted edges).](image)

Mohring [70] showed that if a graph is free of ATs, then any minimal chordal supergraph of it is an interval graph. The converse was later shown to be true as well [21]. Since the minimum chordal completion problem (also known as minimum fill-in) is known to be NP-hard on AT-free graphs [1], the minimum interval completion problem remains NP-hard on AT-free graphs. Other graph classes on which the minimum interval completion problem remains NP-hard include chordal graphs [76], permutation graphs [9], and cocomparability graphs [43]. On the positive side, see [58] for some polynomial solvable special cases.

---

2A historical note is worthy on both the NP-completeness of minimum interval completion and its equivalence to profile minimization. Both were settled in 1970s, but there are quite a lot of mis-references in literature. First, the most widely cited reference for the NP-completeness of the minimum interval completion problem is Garey and Johnson [35] (GT35), which, nevertheless, refers to an unpublished manuscript that has never appeared. The first published proof was given by [53], and another proof was given implicitly by Yannakakis [91] (see [50]). Second, as far as I can ascertain, Rose [77] first correlated the optimization problems (fill-in and bandwidth) in sparse matrix computation and graph modification problems, and Tarjan [80] first showed that profile minimization is equivalent to minimum interval completion. Therefore, the complexity of profile minimization was settled by [53] as well.
1.5 Graph modification problems and fixed-parameter tractability

Many classical graph-theoretic problems can be formulated as graph modification problems to special graph classes. They include 4 out of the 21 problems in Karp’s list [51], and at least 18 problems in Garey and Johnson’s magnum opus [35]. On the other hand, many NP-hard problems are known to be solvable in polynomial time when restricted to some special graph class \( \mathcal{G} \), e.g., coloring and maximum independent set on interval and chordal graphs. Therefore, a heuristic approach is to work on the closest graph \( G \in \mathcal{G} \) to the input graph, and then adjust the obtained solution of \( G \) to \( G \). Indeed, since the graph considered, e.g., the earlier work of Kaplan et al. [50] and Cai [14] studied parameterized complexity of completion problems to chordal graphs, unit interval graphs, and related graphs. Indeed, since the graph modifications problems are natural computational model for detecting errors in experimental data, which are assumed to be small, they gave one important motivation of parameterized computation. In the special case when the desired graph class \( \mathcal{G} \) can be characterized by a finite number of forbidden (induced) subgraphs, the fixed-parameter tractability of modification problems to \( \mathcal{G} \) follows from a basic bounded search tree algorithm [14]. However, many important graph classes, e.g., forests, bipartite graphs, and chordal graphs, have minimal obstructions of arbitrarily large size (cycles, odd cycles, and holes, respectively). It is much more challenging to obtain fixed-parameter tractability results for such classes.

Modification problems related to chordal graphs have been intensively studied, and all variations are known to be FPT [50, 14, 68] (see also [16]). In particular, to fill holes by adding edges is well understood. Since all other three graphs in \( \mathcal{F}_{111} \) have a (small) constant number of vertices, the unit interval completion problem is FPT as well [50]. Likewise, for unit interval vertex deletion, we can dispose of all chordal subgraphs in \( \mathcal{F}_{111} \), and then apply the algorithm for chordal vertex deletion [68, 16]; here we are using the facts that vertex deletions leave an induced subgraph and that chordal graphs are hereditary. This settles the fixed-parameter tractability of unit interval vertex deletion. However, neither of these two observations above applies to unit interval edge deletion: on the one hand, the deletion of a single edge breaks a hole of an arbitrary length; on the other hand, the subgraph obtained after edge deletions is not induced.

Besides holes, \( \mathcal{F}_1 \) has another infinite set of obstructions (CAWs), which is far less understood [21, 62]. Therefore, the modification problems related to interval graphs are more complicated, e.g., adding or deletion a single edge is sufficient to fix a arbitrarily large CAW. Their fixed-parameterized tractability were frequently posed as important open problems [50, 29, 8]. Only after about two decades were interval completion and interval vertex deletion shown to be FPT [86, 17]. Both algorithms, however, suffer from high time complexity. Again, neither approach generalizes to interval edge deletion in a natural way, whose parameterized complexity remained open.

1.6 Efficient detection of (small) forbidden induced subgraphs

If every minimal forbidden induced subgraph of the object graph class can be destroyed by a bounded number of ways, the parameterized modification problem reduces to efficiently detecting a forbidden induced subgraph, preferably in linear time. For example, both chordal completion and unit interval completion can be solved in linear time [50]. This observation can be used even a graph class has an infinite number of minimal forbidden subgraphs that we do not know how to destroy directly. For example, a naive algorithm for unit interval vertex deletion is to call an algorithm for chordal deletion after destroying all minimal forbidden induced subgraphs that are small. It was later observed that the second step is unnecessarily, and a graph without small forbidden induced subgraph can be easily handled [85]. See [16, 17] for similar results. This approach poses a more demanding question, i.e., the efficient detection of (problem-specifically) small forbidden induced subgraphs.

Kratsch et al. [61] presented a linear-time algorithm for detecting a hole or an AT from a non-interval graph. It first calls the hole-detection algorithm of Tarjan and Yannakakis [82], which either returns a hole, or reduces the problem to finding an AT in a chordal graph. A witness for this AT is also provided, which, although unnecessarily minimal, can be used to easily retrieved a minimal one, i.e., a CAW (see also [67]). The additional chordal condition for the detection of an AT is crucial: we do not know how to find an AT in a general graph in linear time. The best known recognition algorithm for AT-free graphs takes \( O(|G|^{2.82}) \) time [59], and Kratsch and Spinrad [62] showed that this algorithm can be used to find an AT in the same time if the graph contains one. A more important result of [62] is that recognizing AT-free graphs is at least as hard as finding a triangle. The detection of an AT cannot be easier than the recognition of AT-free graphs,
and hence a linear-time algorithm for it is very unlikely to exist. (See also \[79\].)

Obviously, for any graph class defined by a set \(\mathcal{F}\) of forbidden induced subgraphs, the detection of a subgraph in \(\mathcal{F}\) is never easier than the recognition of this graph class. On the other hand, we have seen that the detection of holes, ATs, and subgraphs in \(\mathcal{F}_1\) can be done in the same asymptotic time as the recognition of chordal graphs, AT-free graphs, and interval graphs, respectively. From these examples one may have observed that the requirement of explicit evidence does not seem to pose an extra burden to the recognition algorithms. This is known to be true for almost all polynomial-recognizable graph classes with known characterization of forbidden induced subgraphs.

However, it changes drastically when the evidence is further required to have a small or minimum number of vertices. A shortest hole can be found in \(O(|G|^2)\) time as follows: (1) guess an edge \(uv\) in the hole; (2) delete all common neighbors of \(u\) and \(v\) from \(V(G)\) and \(uv\) from \(E(G)\); and (3) run breadth-first search starting from \(u\) until it reaches \(v\). There is no asymptotically faster algorithm known in literature. To make it worse, we are not aware of any algorithm that finds a CAW of the minimum number of vertices in \(o(|G|^3 \cdot |G|)\) time (cf. the discrepancy between the time complexity of finding an arbitrary hole and finding an arbitrary AT). On the other hand, a linear-time algorithm for finding a minimum subgraph in \(\mathcal{F}_1\) is very unlikely to exist. Let \(G'\) be the graph obtained by subdividing a graph \(G\) (i.e., for each edge \(uv \in E(G)\), adding a new vertex \(x\), connecting it to both \(u\) and \(v\), and deleting \(uv\), then \(G\) contains a triangle if and only if the minimum subgraph of \(G'\) in \(\mathcal{F}_1\) is a 6-hole. Since \(G'\) has \(|G| + |G|\) vertices and \(2|G|\) edges, a linear-time algorithm for finding a minimum subgraph in \(\mathcal{F}_1\) can be used to detect a triangle in linear time. For the same reason, we do not expect to find a shortest hole in linear time.

The problem most closely related to detecting a subgraph in \(\mathcal{F}_1\) should be detecting a Tucker submatrix from a matrix that does not have consecutive-ones property \[83\]. Recall that a graph is an interval graph if and only if its clique matrix has the consecutive-ones property \[33\], and thus algorithms for recognizing consecutive-ones matrices and finding a Tucker submatrix can be used to recognize interval graphs and finding a subgraph in \(\mathcal{F}_1\) respectively. Both the recognition of consecutive-ones matrices and the detection of any Tucker submatrix can be done in linear time \[67\]. However, given an \(m \times n\) matrix, the best known algorithm takes \(O(nm^4)\)-time to find a Tucker submatrix of the minimum size \[6\]. Likewise, Chudnovsky et al. \[19\] showed that the shortest even hole can be found in \(O(|G|^3)\) time, while the recognition of even-hole-free graphs and the detection of any even hole can be done in \(O(|G|^{11})\) time \[18\]. The complexity of detecting any odd hole remains open, while some partial result is known, e.g., a shortest odd hole in a claw-free graph can be found in \(O(|G|^2 + |G|^2 \log |G|)\) time \[56\].

One crucial step of our algorithm is to find all simplicial vertices. Again, it is unlikely to be done in linear time: Kratsch and Spinrad \[62\] showed that counting the number of simplicial vertices is already at least as hard as detecting a triangle. Indeed, there is even no known linear-time algorithm that detects a single simplicial vertex. The only known way of finding a simplicial vertex is the trivial one, which takes \(O(|G|^m)\) time or \(O(|G| \cdot |G|)\) time without using fast matrix multiplication. Kloks et al. \[57\] showed that in the same time one can actually list all simplicial vertices. Their algorithm can be adapted to iteratively eliminate all simplicial vertices, i.e., find the maximum induced subgraph that contains no simplicial vertex. This is the recognition algorithm for chordal graphs suggested in Fulkerson and Gross \[33\].

1.7 Our techniques

The development of our algorithms involves several nontrivial techniques. As far as I know, many of them are novel. We describe here the main challenges for problems on interval graphs and intuitions behind the techniques that we use to address them. They can be roughly put into two categories: for the linear dependence on the graph size and for the smaller exponential dependence on the parameter—the usage of modules is important for both, but it is only discussed in the second part as the usage in the first one is standard \[34\], \[69\], \[48\], \[49\], \[17\]. Further extensions of these techniques will be discussed in Section 6.

In general, our algorithms attend to small subgraphs in \(\mathcal{F}_1\) and then large ones. We say that CAWs and holes in \(\mathcal{F}_1\) are small and short respectively; other CAWs, namely, \(s\) and \(\S\), are large, and holes of length \(j\) are long. However, the best algorithms

\[\text{for the trivial ones, which takes fast matrix multiplication time, or } O(|G|^3) \text{ without using matrix multiplication. Recall that the current fastest algorithm for matrix multiplication takes } O(|G|^\omega) \text{ time, where } \omega < 2.3727. \text{ Spinrad listed an } o(|G|^2) \text{-time combinatorial algorithm for detecting triangles as an open problem \[79\]. Open Problem 8.1.} \text{ In the same work he also conjectured that it is computationally equivalent to 0,1-matrix multiplication verification problem. Recall that in matrix multiplication verification problem, we are given three matrices } A, B, \text{ and } C, \text{ and asked whether } A \times B = C \text{ or not. See also \[60\]}.\]

\[\text{In the published version of the paper \[19\], the algorithm is stated as detecting an arbitrary even hole and it was asked as an open problem for an algorithm that finds a shortest one. But according to Seymour (private communication), the authors later observed that the return of their algorithm has to be the shortest. We remark that techniques in \[18\] does not seem to be generalizable to this task.} \]
six or more are long. This is why we define locally interval graphs in the first place. It is worth noting that the threshold is chosen by structural properties instead of sizes, e.g., a † of seven vertices is considered large and a 6-hole is considered long.

The biggest challenge is surely how to detect in linear time a short hole or small CAW, for which a trivial algorithm takes \( O(|G|^3 \cdot |G|) \) time. As explained in Section 1.6 even if there is a better algorithm, its runtime is very unlikely to be in linear time. This explains why all previous algorithms employing a similar approach have a polynomial factor of \( \Omega(|G|^3 \cdot |G|) \), e.g., \( |G|^3 \cdot |G| \) in [86], \( |G|^6 \) in [85], and \( |G|^{10} \) in [17]. Our observation here is that we may find short-hole-or-small-CAW in linear time. One should be noted that this task is far more demanding than finding a hole-or-AT with no size requirement: an arbitrary hole can be found easily, while neither short holes nor small CAWs can be found in linear time. For this purpose, we study the class of locally interval graphs, of which those chordal ones turn out to be easy to manipulate. Our main technique appears in the way we deal with a non-chordal locally interval graph. We focus on its quotient graph \( G_Q \) and introduce an auxiliary graph when \( G_Q \) is prime. The auxiliary graph carries all structural information of \( G_Q \) useful for us and is easy to manipulate. For the purpose of the current paper, we do not need a complete certifying algorithm for recognizing prime locally interval graphs. Instead, we are satisfied with a dichotomy outcome, namely, either a subgraph in \( F_{1,1} \) or a structural decomposition of the original graph.

The algorithm of [17] breaks holes first, and then CAWs in a chordal graph. There is no clear way to implement this tactic in linear time: the disposal of holes introduces a factor \( |G| \), while finding a CAW gives another factor \( |G| \). Neither of them seems to be improvable to \( o(|G|) \). We are thus forced to consider an alternative approach, i.e., we may have to deal with CAWs in a non-chordal graph. Hence completely new techniques are required. Overcoming these two difficulties enables us to deliver linear-time algorithms.

To claim the fixed-parameter tractability of INTERVAL EDGE DELETION and better dependence on \( k \) for INTERVAL COMPLETION, we still have some major concerns to address. We have seen that techniques behind the algorithm for INTERVAL COMPLETION [86] do not apply to deletion versions. We explain here why the techniques behind the algorithm for INTERVAL VERTEX DELETION [17] fail for INTERVAL EDGE DELETION as well. The algorithm in [17] heavily relies on the fact that the deletion of vertices leaves an induced subgraph. Essentially, it looks for a minimum set of vertices intersecting all subgraphs in \( F_1 \), so called hitting set. Deleting any vertex from a subgraph in \( F_1 \) breaks this subgraph once and for all, but adding/deleting an edge to break an erstwhile subgraph in \( F_1 \) might introduce new one(s). As a result, the “hitting set” observation does not apply to edge modifications. In particular, we cannot break all holes and then assume that the graph is chordal thereafter.

The first difficulty which presented itself at this point is on the preservation of modules, which is trivial for the vertex deletion. Simple examples tell us that not all maximum spanning interval subgraphs and minimum interval supergraphs preserve all modules. We observe that under appropriate technical conditions, there exist some maximum spanning interval subgraph and minimum interval supergraph that preserve all modules. These observations focus us on prime graphs again, for which the main technical idea appears in the way we deal with large CAWs. It is observed that in a large CAW, some special vertices play a very important role. (Coincidently, the number of them is either six or seven, matching that of small CAWs.) We call the subgraph induced by them as the frame of this large CAW. There are only bounded number of possible ways to alter a frame, and hence it suffices to consider what happens when the frame holds unchanged: the modification in this CAW has to be local and small. We find a CAW that is minimal in a sense, and show that if the frame of is unchanged, then we can apply a canonical modification to this CAW. The same basic observation works for all three problems.

2 Outline

This section sketches the outline of our algorithms. Let \( \{M_1, \ldots, M_p\} \) be the set of maximal strong modules (inclusive-wise maximal strong modules that are not \( V(G) \) of \( G \)). They partition \( V(G) \) and for each \( 1 \leq i \leq p \), module \( M_i \) defines a vertex \( v_i \) in the quotient graph \( G_Q \) (see Section 3.2 for definitions and explanations). Denote by \( SI(G) \) the set of all simplicial vertices of a graph \( G \), which are pairwise nonadjacent if \( G \) is prime. It is easy to observe from Fig. 1 that each CAW contains precisely three simplicial vertices. They form the unique AT of this CAW, and are called the terminals of this CAW:\[^5]\n
\[^5\] It is easy to verify that a hole of six or more vertices also witnesses an AT (e.g., any three pairwise nonadjacent vertices from it) and is minimal. It, however, behaves quite differently from a CAW, e.g., none of its vertices is simplicial, and it has more than one ATs—indeed, every vertex is in some AT.
Part I. Characterization and decomposition of locally interval graphs (Prop. 1.3 and Thm. 1.5). The quotient graph \( G_Q \) of a graph \( G \) is isomorphic to some induced subgraph of it. Thus, if \( G \) is a (locally) interval graph, then \( G_Q \) is a (locally) interval graph as well, so are all the subgraphs induced by its maximal strong modules. The converse, nevertheless, does not hold in general. Even if subgraphs \( G|M_i \) for all \( 1 \leq i \leq p \) and \( G_Q \) are (locally) interval graphs, \( G \) might still contain some subgraph in \( \mathcal{F}_1 \) (or \( \mathcal{F}_2 \)). We observe (in Section 3.2) that such a subgraph has to be a 4-hole. In particular, if two non-clique modules \( M_1 \) and \( M_2 \) are adjacent, then we can find a 4-hole by taking two nonadjacent vertices \( v_1, v_2 \in M_1 \) and two nonadjacent vertices \( v_3, v_4 \in M_2 \); if a non-clique module \( M \) is adjacent to both modules \( M_1 \) and \( M_2 \) but \( M_1 \) and \( M_2 \) are nonadjacent, then we can find a 4-hole by taking any pair of vertices \( v_1 \in M_1 \) and \( v_2 \in M_2 \) and two nonadjacent vertices \( v_3, v_4 \in M \). There is no further possibility, and this concludes Prop. 1.3.

Prop. 1.3 reduces the problem of finding a subgraph of a graph \( G \) in \( \mathcal{F}_{11} \) to two simple tasks, namely, finding a subgraph of its quotient graph \( G_Q \) in \( \mathcal{F}_{11} \) and finding all simplicial vertices of \( G_Q \) when it is a locally interval graph (more explanation is given at the end of this part). Both tasks are trivial when the quotient graph \( G_Q \) is a locally interval graph, then for any vertex \( v \in V(G) \), there are a vertex \( u \in V(G) \) and another vertex \( l \in V(G) \) in \( N[v] \) that \( u \) and \( l \) are nonadjacent, then we can find a 4-hole by taking any pair of vertices \( v_1 \in M_1 \) and \( v_2 \in M_2 \) and two nonadjacent vertices \( v_3, v_4 \in M \). There is no further possibility, and this concludes Prop. 1.3.

Theorem 2.1. Let \( W \) be a large CAW of a prime graph \( G \). We can in \( O(|G|) \) time find a subgraph of \( G \) in \( \mathcal{F}_{11} \) if the shallow terminal of \( W \) is non-simplicial in \( G \).

If a locally interval graph \( G \) is chordal, then by Thm. 2.1 \( G - SI(G) \) is an interval graph. A caterpillar decomposition for \( G \) can be obtained by adding \( SI(G) \) to a clique path decomposition for \( G - SI(G) \) (Section 5.3). This settles the chordal case of Thm. 1.5 and hence in the rest of Part I, we may assume that \( G \) is not chordal and contains a long hole \( H \).

Theorem 2.2. Let \( H \) be a hole of a prime graph \( G \). We can in \( O(|G|) \) time find a subgraph of \( G \) in \( \mathcal{F}_{11} \) if there exists \( v \in V(G) \) satisfying one of the following: (1) the neighbors of \( v \) in \( H \) are not consecutive; (2) \( v \) is adjacent to \( |H| - 2 \) or more vertices in \( H \); and (3) \( v \) is non-simplicial and nonadjacent to \( H \).

If \( G \) is a locally interval graph, then for any vertex \( h \) of the hole \( H \), the subgraph \( G - N[h] \) must be chordal; otherwise, \( h \) and any hole of \( G - N[h] \) will satisfy Thm. 2.2(3). Therefore, combining Thms. 2.1 and 2.2 we conclude that \( G - SI(G) - N[h] \) must be an interval subgraph, and has a linear structure. These observations inspire the definition of the auxiliary graph \( \mathcal{U}(G) \) (with respect to \( H \)), which is the main technical tool for analyzing prime non-chordal graphs. Here we need some local properties on a special vertex of \( H \), which can be obtained by some local operations in linear time (Section 5.1). We number vertices in \( H \) such that \( u_0 \) is this special vertex and define \( T = N[u_0] \). We designate the ordering \( u_0, u_1, u_2, \ldots \) of traversing \( H \) as clockwise, and the other counterclockwise. The local properties enable us to assign a direction to each edge between \( T \) and \( T \) in accordance with the direction of \( H \) itself. We use \( E_c \) and \( E_{cc} \) to denote the set of clockwise and counterclockwise edges from \( T \), respectively; \( \{E_c, E_{cc}\} \) partitions \( T \times T \).

Definition 2. The vertex set of \( \mathcal{U}(G) \) consists of \( T \cup L \cup R \cup \{w\} \), where \( L \) and \( R \) are distinct copies of \( T \), i.e., for each \( v \in T \), there are a vertex \( v^l \) in \( L \) and another vertex \( v^r \) in \( R \), and \( w \) is a new vertex distinct from \( V(G) \). For each edge \( uv \in E(G) \), we add to the edge set of \( \mathcal{U}(G) \):

- an edge \( uv \) if neither \( u \) nor \( v \) is in \( T \);
- two edges \( u^l v^l \) and \( u^r v^r \) if both \( u \) and \( v \) are in \( T \); or
- an edge \( u^l v \) or \( u v^r \) if \( v \in T \) and \( u v \in E_c \) or \( uv \in E_{cc} \) respectively.

Finally, we add an edge \( wv \) for every \( v \in T : uv \in E_{cc} \).

It is easy to see that the order and size of \( \mathcal{U}(G) \) are upper bounded by \( 2|G| \) and \( 2|G| \) respectively. We will show in Section 5.4 that an adjacency list representation of \( \mathcal{U}(G) \) can be constructed in linear time.

The shape of symbol \( \mathcal{U} \) is a good hint for understanding the structure of the auxiliary graph. Suppose \( G \) has an olive-ring structure, then \( \mathcal{U}(G) \) has a linear structure, which is obtained by unfolding the olive-ring as follows. The subgraph \( G - T \) has a caterpillar structure, to the ends of which we append two copies of \( T \). The two copies of \( T \), namely, \( L \) and \( R \), are identical, and every edge between \( T \) and \( T \) is carried by only one copy of it. The auxiliary graph has the following properties when \( G \) is a non-chordal prime locally interval graph, which allow us to fold (the reverse of the “unfolding” operation) the linear structure of \( \mathcal{U}(G) \) back to produce the olive-ring decomposition for \( G \).
Theorem 2.3. A vertex different from \((w) \cup R\) is simplicial in \(U(G)\) if and only if its original vertex is simplicial in \(G\). Moreover, we can in \(O(|G|)\) time find a subgraph of \(G\) in \(\mathcal{F}_{1,1}\) if 1) \(U(G)\) is not chordal; or 2) \(U(G - SI(G))\) is not an interval graph.

We may assume the graph \(U(G)\) is chordal, whose simplicial vertices can be identified easily. As a result of Thm. 2.3, this gives also \(SI(G)\), and we can obtain the graph \(U(G - SI(G))\). If it is not an interval graph, then we are done with Thm. 1.5. Otherwise, we apply the following operation to sequentially build a hole decomposition for \(G - SI(G)\) and an olive-ring decomposition for \(G\).

Lemma 2.4. Given a clique path decomposition for \(U(G - SI(G))\), we can in \(O(|G|)\) time construct a clique decomposition for \(G - SI(G)\) that is a hole.

Theorem 2.5. Given a clique hole decomposition for \(G - SI(G)\), we can in \(O(|G|)\) time construct a clique decomposition for \(G\) that is an olive-ring.

Putting together these steps, we get the decomposition algorithm in Fig. 3, from which Thm. 1.5 follows. This concludes the proof of the characterization and decomposition of prime locally interval graphs.

Algorithm `decompose-LIG-prime(G)`

**INPUT:** a prime graph \(G\).

**OUTPUT:** a caterpillar/olive-ring decomposition for \(G\) or a subgraph of \(G\) in \(\mathcal{F}_{1,1}\).

1. if \(G\) is chordal then
   - if \(G\) is an interval graph then return a clique path decomposition;
   - if \(G - SI(G)\) is an interval graph then return a caterpillar decomposition for \(G\);
   - else call Thm. 2.1
2. find a hole \(H\) of \(G\); build \(U(G)\);
3. if \(U(G)\) is not chordal then call Thm. 2.3(1);
4. if \(SI(U(G))\) and \(SI(G)\); construct \(G - SI(G)\);
5. if \(U(G - SI(G))\) is not an interval graph then call Thm. 2.3(2);
6. call Lem. 2.4 and Thm. 2.5 to build an olive-ring decomposition for \(G\).

Figure 3: The decomposition algorithm for Thm. 1.5

Summarizing this part, let us explain how to recursively use Prop. 1.3 and Thm. 1.5 to find a subgraph of \(G\) in \(\mathcal{F}_{1,1}\). If \(G_Q\) has no edge, then \(G\) is disconnected, and we work on its components separately. We have three cases when \(G_Q\) is a clique: (i) if all maximal strong modules are trivial, then \(G\) is a clique; (ii) if there is precisely one maximal strong module \(M\) that is nontrivial, then we work on \(G[M]\); (iii) otherwise, two or more maximal strong modules are nontrivial, then we find a 4-hole. In the remaining case, \(G_Q\) is prime, to which we apply Thm. 1.5; if it does not output a subgraph in \(\mathcal{F}_{1,1}\), then we are in possession of a caterpillar/olive-ring decomposition for \(G_Q\). As a byproduct, we have \(SI(G_Q)\) and are able to check the other two conditions of Prop. 1.3. If every module corresponding to a vertex in \(V(G_Q)\) \(\backslash SI(G_Q)\) induces a clique, then we recursively work on other modules one by one.

Part II. Recognition of almost unit interval graphs. The recognition of unit-interval–ke has been well studied, and our improvement is obtained by some simple observation on filling claws, nets, and suns (Lem. 6.2). Hence we focus on unit-interval+ke and unit-interval+kv, i.e., the deletion problems. We can use Thm. 1.5 to find a claw, net, sun, or short hole efficiently (Lem. 6.5), and then use simple branching to break it. After all of them have been destroyed, if the graph is chordal, then problems are solved. Otherwise, its quotient graph has a clique hole decomposition and every module induces a clique; in other words, the graph is a claw-free normal Helly circular-arc graph (Section 5). We dispose of its holes as follows. We say that a set \(V_+\) of vertices (resp., a set \(E_+\) of edges) is a hole cover (resp., an edge hole cover) of \(G\) if \(G - V_+\) (resp., \(G - E_+\)) is chordal. Note that \(V_+\) is empty if and only if \(G\) is chordal.

Proposition 2.6. We can find in \(O(|G|)\) time a minimum hole cover of a normal Helly circular-arc graph \(G\).

Lemma 2.7. Given a normal Helly circular-arc graph \(G\), the deletion of a minimum edge hole cover will not introduce a CAW or claw. Moreover, we can in \(O(2^k \cdot |G|)\) time decide whether there exists a minimum edge hole cover of at most \(k\) edges.

Therefore, all holes can be broken at a fell swoop in linear time, which concludes the recognition of unit-interval+ke and unit-interval+kv.
Part III. Recognition of almost interval graphs. Recall that every modification problem has a minimization version, which asks for the minimum number of operations. Clearly, a graph \( G \) is in interval+kv, interval+ke, or interval–ke if and only if the minimum number of vertex deletions, edge deletions, or edge additions respectively that transform \( G \) into an interval graph is no more than \( k \). Although the parameterized versions we are working on do not explicitly ask for a minimum solution, a minimum one will serve our purpose. We observe that there are always minimum solutions that are well aligned with modules of the input graph (Thms. 1.6-1.8). Indeed, we will prove the following variations of Thms. 1.6-1.8 which are more convenient for the analysis of our algorithms. As we will show later, they are actually equivalent to Thms. 1.6-1.8.

**Theorem 2.8.** Let \( G \) be a graph of which every 4-hole is contained in some maximal strong module. Let \( G - V_+ \) be a maximum induced interval subgraph of \( G \). For any maximal strong module \( M \) of \( G \) such that \( M \not\subseteq V_- \), the set \( M \setminus V_- \) is a module of \( G - V_- \), and replacing \( G[M \setminus V_-] \) by any maximum induced interval subgraph of \( G[M] \) in \( G - V_- \) gives a maximum induced interval subgraph of \( G \).

**Theorem 2.9.** Let \( G \) be a graph of which every 4-hole is contained in some maximal strong module. There exists a maximum spanning interval subgraph \( \hat{G} \) of \( G \) such that every maximal strong module \( M \) of \( G \) is a module of \( \hat{G} \), and replacing \( G[M] \) by any maximum spanning interval subgraph of \( G[M] \) in \( \hat{G} \) gives a maximum spanning interval subgraph of \( G \).

The condition of Thms. 2.8 and 2.9 is satisfied if \( G_Q \) is not a clique, \( G_Q \) contains no 4-hole, and every non-simplicial vertex of \( G_Q \) corresponds to a clique of \( G \). All three conditions are easy to check with the olive-ring decomposition.

**Theorem 2.10.** There is a minimum interval supergraph \( \hat{G} \) of \( G \) such that every maximal strong module \( M \) of \( G \) is a module of \( \hat{G} \), and if \( \hat{G}[M] \) is not a clique, then replacing \( \hat{G}[M] \) by any minimum interval supergraph of \( G[M] \) in \( \hat{G} \) gives a minimum spanning interval subgraph of \( G \).

Therefore, we may consider only solutions that satisfy Thms. 2.8, 2.10 which focus us on the quotient graph \( G_Q \), to which we apply Thm. 1.5. There are only a constant number of modifications applicable to a small CAW, and thus we may assume that its outcome is an olive-ring decomposition \( \hat{K} \). Every hole \( H \) of \( X \) is global in the sense that it dominates all holes, i.e., every vertex in the main cycle of the olive-ring decomposition is adjacent to \( H \). In contrast, every CAW is local: it sees at most five vertices of a shortest hole. This structural difference suggests that different techniques are required to handle them. Viewing from a high level and informally speaking, what we have is an olive-ring decomposition, while what we want is a path decomposition (noting that the bags of the latter is unnecessarily a subset of the former).

This task has two natural options with different intermediate steps, one is via a caterpillar decomposition and the other is via a hole decomposition. For the completion problem, as holes can be easily filled, we can always assume that the graph is chordal, i.e., we take the caterpillar decomposition as intermediate. For the deletion problems, we serve CAWs first, and then call Prop. 2.6 or Lem. 2.7 to finish the task.

Let \((s: c_1, c_2 : l, b, r)\) be a large CAW. We consider its terminals as well as their neighbors, i.e., \((s, c_1, c_2, l, b_1, b_d, r)\). It is observed that if all of them are retained and their adjacencies—except \( l \) and \( c_2r \) which exist in a \( \| \) but not a \( | \)—are not changed, then in an interval model of the objective interval graph, they must be arranged in the way depicted in Fig. 4. As indicated by the dashed extensions, the interval for \( c_1 \) (resp., \( c_2 \)) might or might not extend to the left (resp., right) to intersect the interval for \( l \) (resp., \( r \)). Our main observation is on the position of the interval for \( s \), which has to lie between \( b_1 \) and \( b_d \). Recall that \( s \) is originally adjacent to no vertex in the \((b_1, b_d)\)-path \( B \). Therefore, for deletion, we need to delete some vertex or edge to break \( B \), while for completion, we need to add edges to connect \( s \) to some vertex in \( B \).

![Figure 4: Interval model of an unchanged frame.](image)

In the discussion above, what matters is only the terminals of a CAW and their neighbors, while the particular \((b_1, b_d)\)-path \( B \) becomes irrelevant. Indeed, any induced \((b_1, b_d)\)-path \( B' \) in \( N(c_1) \cap N(c_2) \setminus N(s) \) can be used in place of \( B \) to give a CAW, which has the same set of terminals. The similar operation is
thus needed for every B', which inspires us to consider the following two sets of vertices. Of a large CAW (s : c₁, c₂ : l, B, r), the frame is denoted by (s : c₁, c₂ : l, b₁; b₄, r), and the set U₁ of inner vertices is composed of all vertices that can be used to make a CAW with frame (s : c₁, c₂ : l, b₁; b₄, r). We have seen that the particular CAW is immaterial, and thus we use (s : c₁, c₂ : l, l' ; r', r) to denote a frame without a specific path B. For such a frame F, we define

\[ E_(−)(F) = \{sc₁, sc₂, ll', l'c₁, l'c₂, r'c₁, r'c₂, r'r\}; \]
and
\[ E_(+)(F) = \{lc₂, c₁r, l'r', sl', sr'\}. \]

We can find in linear time a frame F that is minimal in a sense (see Definition 8 in Section 8.1). The rest is then devoted to the disposal of this minimal frame.

Consider first maximum induced interval supergraphs. We show that any maximum induced interval subgraph either deletes some vertex of F or a minimum (l', r')-separator in the subgraph induced by U₁ ∪ {l', r'}. More importantly, any minimum (l', r')-separator, which can be found in polynomial time, will suffice in the second case. Recalling that F has at most seven vertices, we have then a 8-way branching for disposing of this CAW. The second case can be viewed informally as follows. All vertices in U are totally decided by I and l', r', c₁, c₂ must be in the main cycle of the olive-ring decomposition, while s is not. What we need to do is thus to find some "place" of the main cycle between l' and r' to accommodate s. We show that it suffices to "cut any thinnest place" between l' and r', and use this space for s.

The "thinnest places" are also crucial in minimum interval supergraphs, though they become far more delicate. Likewise, our focus is on a minimum interval supergraph U that contains no edge in E_(−)(F), and thus in an interval model for \( \hat{G} \), intervals for F are arranged as Fig. 4. In particular, l' and r' remain nonadjacent in \( \hat{G} \). As holes are easy to fill, we will assume that the graph is chordal, and the clique decomposition is a caterpillar. Therefore, l' and r' can be used to decide a left-right relation for both the caterpillar decomposition of \( \hat{G} \) and an interval model of \( \hat{G} \). After adding edges, an interval for a vertex right to r' might intersect part or all between intervals for l' and r'. We argue that such a interval either reaches l', or is to the right of some position (informally speaking, the "rightmost thinnest place"). A symmetric argument works for a vertex to the left of l'. As a result, we have two points such that all structures between them is totally decided by F and U₁; in particular, it suffices to put s in any "thinnest place" in between. This gives a 6-way branching.

The situation becomes even more complicated for maximum spanning interval subgraphs. The assumption that no edge in E_(−)(F) is deleted does not suffice. We find a shortest (l', r')-path B with all inner vertices from U₁. If B has a bounded length, then we try every edge in it. Otherwise, we argue that either one of the first or last O(k) edges of B is deleted, or it suffices to find a minimum set of edges whose deletion separates l' and r' in U₁. This gives an O(k) branching.

Putting together these steps, a high-level outline of algorithms for all problems related to interval graphs is given in Fig. 5. This concludes the recognition of interval+kv, interval+ke, and interval−ke.

Figure 5: Outline of algorithms for problems related to interval graphs.

**Organization.** The rest of the paper is organized as follows. Section 3 sets the definitions and recalls some basic facts. Section 4 gives the characterization of large CAWs and long holes in prime locally interval graphs, and proves Thms 2.1 and 2.2. Section 5 presents the details of decomposing prime graphs and...
proves Thms. 2.3 and 2.5. Section 6 proves Prop. 2.6 and Lem. 2.7, and use them to recognize almost unit interval graphs. Section 7 relates modules to minimum modifications, and proves Thms. 2.8 and 2.10. Section 8 presents the algorithms for recognizing almost interval graphs. Section 9 closes this paper by describing some follow-up work and discussing some possible improvement and new directions.

Considering the length of this paper, efforts have been made to ensure reasonable independence between sections. Sections 4 and 5, together can be viewed as the first unit, and later reference to it will not go beyond Prop. 1.3 and Thm. 1.5. Both section 6 and section 7 are self-contained, and thus can be read independently. However, Section 8 relies on all previous sections.

3 Preliminaries

This section recalls graph-theoretic and algorithmic terminology and presents several simple procedures. More specific concepts will be introduced in later sections when they are needed.

We sometimes use the customary notation $v \in G$ to mean $v \in V(G)$, and $u \sim v$ to mean $uv \in E(G)$. Given a subset $V_\ldots$ of vertices (resp., $E_\ldots$ of edges), we use $G - V_\ldots$ (resp., $G - E_\ldots$) to denote the induced subgraph $G[V(G) \setminus V_\ldots]$ (resp., the spanning subgraph $(V(G), E(G) \setminus E_\ldots)$). The set of neighbors of $v$ in graph $G$ is denoted by $N_G(v)$, where the subscript $G$ is only specified when clarity demands. The degree of a vertex $v$ is defined by $d(v) = |N(v)|$. For a subset $U$ of vertices, we denote by $N(U)$ the set of common neighbors of $U$, i.e., $N(U) = \bigcap_{v \in U} N(v)$. A graph is complete if every pair of vertices is adjacent. A clique is a complete subgraph, and is maximal if its vertices have no common neighbor.

We give a symbol for each vertex in a large CAW, i.e., a $(\text{s})$ or $(\text{t})$ (see Fig. 1). Recall that $s$ is the shallow terminal, and the removal of $N[s]$ from this CAW leaves an induced path. This path connects the other pair of terminals $l, r$, called base terminals. The neighbor(s) $c_1, c_2$ of $s$ are the center(s), and all other vertices, $\{b_1, \ldots, b_d\}$, are called base vertices. The $(b_1, b_d)$-path through all base vertices is called the base, denoted by $B$. The center(s) and base vertices are called non-terminal vertices. Note that a $(\text{s})$ has only one center, and both $c_1$ and $c_2$ refer to it. Therefore, we uniformly use $(s : c_1, c_2 : l, B, r)$ to denote both kinds of large CAWs. For the sake of notational convenience, we will also use $b_0$ and $b_{d+1}$ to refer to the base terminals $l$ and $r$, respectively, even though they are not part of the base $B$.

All indices of vertices in a hole $H$ should be understood as modulo $|H|$, e.g., $h_{-1} = h_{|H| - 1}$. We define the ordering $h_{-1}, h_0, h_1, \ldots$ of traversing $H$ to be clockwise, and the other to be counterclockwise. In other words, vertices $h_1$ and $h_{-1}$ are the clockwise and counterclockwise successors, respectively, of $h_0$; and edges $h_0h_1$ and $h_0h_{-1}$ are clockwise and counterclockwise, respectively, from $h_0$.

More often than not, our algorithms need to choose one of a sequence of cases. This is usually written as “if ... else if ... else if ....” However, we omit “else” from succeeding cases when the previous branch always returns or calls another subroutines that surely terminates the algorithm. See, e.g., steps 2-6 of Fig. 6.

3.1 Algorithmic fundamentals

All algorithms in this paper are designed on a random-access machine, and all graphs are stored in the form of adjacency lists, both standard [81]. No advanced data structure will be involved.

Several places of our algorithm will need to explore the graph to find a vertex with some specific property. For this purpose we will apply the procedure known as breadth-first search (BFS). Let us first briefly describe the process of the standard form of BFS, and then explain the variation we are using later. It suffices to consider connected graphs. The main data structure is a queue for maintaining vertices to be visited. Initially the queue is empty, and all vertices are marked as “unvisited.” The procedure starts by placing an (arbitrary or pre-specified) vertex onto the queue and mark it as “visited.” In each iteration, the first vertex in the queue is removed from the queue and then all its “unvisited” neighbors are placed onto the queue, and their labels are updated to “visited.” This process terminates when the queue becomes empty again. The process of BFS gives an ordering of $V(G)$. On a connected graph, every vertex will be enqueued and dequeued precisely once, and each edge checked twice, when its end-vertices are dequeued respectively.

Depending on the applications, various operations are applied with the search process. A common one is to record for each vertex $v$ the neighbor that has just been removed when $v$ is enqueued, denoted by $\text{prev}(v)$. By definition, $u \sim v$, and is the earliest visited vertex that is adjacent to $v$. With the only exception of the starting vertex $t$, $\text{prev}(v)$ is defined on every other vertex. Letting $v_0 = v$ and $v_{i+1} = \text{prev}(v_i)$ for $i = 0, \ldots$ until $v_l = t$, we can always find a path that leads from $v$ to $t$, which is a shortest $(v, t)$-path.

BFS can be used to find some vertex that has some property and has the shortest distance to the starting vertex. As long as the total time for the property test for all vertices can be done in linear time, the whole
process remains linear-time. In particular, this is the case when the property for vertex \( v \) can be tested in \( O(d(v)) \) time. Another modification we need on BFS is that, instead of starting from a single vertex, we may start from a subset \( X \) of vertices. The path retrieved using the \( \text{prev} \) function then gives a shortest path from a vertex to \( X \).

We will need the following procedure in several places.

**Proposition 3.1.** Let \( U \) be a subset of ordered vertices. For any vertex \( v \), we can in \( O(d(v)) \) time compute an ordered list that contains \( i \) if and only if \( u_i \in N[v] \).

**Proof.** Let \( U = \{u_0, u_2, \ldots, u_{|U|-1}\} \). We pre-allocate a list \( \text{IND}(v) \) slots, initially all empty. For each neighbor of \( v \), if it is \( u_i \), then add \( i \) into the next empty slot of \( \text{IND} \). After all neighbors of \( v \) have been checked, we shorten \( \text{IND} \) by removing empty slots from the end, which leaves \( |N[v] \cap U| \) slots. We radix sort these indices and return the sorted list. \( \square \)

### 3.2 Modular decomposition

A subset \( M \) of vertices forms a **module** of \( G \) if all vertices in \( M \) have the same neighborhood outside \( M \). In other words, for any pair of vertices \( u, v \in M \), a vertex \( x \notin M \) is adjacent to \( u \) if and only if it is adjacent to \( v \) as well. Two disjoint modules are either nonadjacent or completely adjacent. The set \( V(G) \) and all singleton vertex sets are modules, called **trivial**. A graph on less than three vertices has only trivial modules, while a graph on three vertices always has a nontrivial module. A graph on at least four vertices is **prime** if it contains only trivial modules, e.g., all holes of length at least 5 and all CAWs are prime. The fact that for any module \( M \), a subgraph \( X \) in \( G \) different from a 4-hole contains 0, 1, or \( |X| \) vertices from \( M \) can be derived from the primality of \( X \) and the following well-known property:

**Proposition 3.2.** For any module \( M \) and vertex set \( U \) of \( G \), the set \( U \cap M \), if not empty, is a module of the subgraph \( G[U] \).

A module \( M \) is **strong** if for every other module \( M' \) that intersects \( M \), one of \( M \) and \( M' \) is a proper subset of the other. All trivial modules are strong. We say that a strong module \( M \) different from \( V(G) \) is **maximal** if the only strong module properly containing \( M \) is \( V(G) \). Let \( \{M_1, \ldots, M_p\} \) be the set of maximal strong modules of \( G \). Using definition it is easy to verify that they are disjoint and every vertex \( v \) appears in one of them, i.e., they partition \( V(G) \). There is a \( p \)-vertex **quotient graph** \( G_Q \) associated with \( G \), where for any pair of distinct \( i, j \) with \( 1 \leq i, j \leq p \), the \( i \)th and \( j \)th vertices of \( G_Q \) are adjacent if and only if \( M_i \) and \( M_j \) are adjacent in \( G \). The quotient graph is an induced subgraph of \( G \), e.g., for each \( 1 \leq i \leq p \), we can pick an arbitrary vertex from \( M_i \) as \( v_i \). From \( G_Q \) and \( G[M_i] \) for all \( 1 \leq i \leq p \), the original graph \( G \) can be easily and efficiently retrieved.

We can continue the process (finding maximal strong modules and extracting quotient graphs) for each subgraph \( G[M_i] \) with \( 1 \leq i \leq p \), until every maximal strong module is trivial. This process is known as modular decomposition of \( G \), and its output is a tree structure, called the modular decomposition tree of the graph, whose nodes have a one-to-one correspondence to all strong modules of \( G \) \([24]\). The modular decomposition tree of a graph can be constructed in linear time and stored in linear space; a comprehensive list of modular-decomposition algorithms can be found in de Montgolfier's PhD thesis \([24]\).

If \( G \) is disconnected, then each maximal strong module is a component of \( G \), and the quotient graph \( G_Q \) has no edge. Recall that the complement graph of a graph \( G \) is defined on the same vertex set \( V(G) \), and a pair of vertices \( u \) and \( v \) is adjacent in the complement graph if and only if \( u \not\sim v \) in \( G \). Thus, a graph and its complement graph have the same set of modules, and if the complement graph of \( G \) is disconnected, then the partition is given by the components of it, and the quotient graph of \( G \) is complete. If both the graph and its complement are connected, then its quotient graph is prime, and has the following property:

**Proposition 3.3.** A prime graph is connected, and all its simplicial vertices are pairwise nonadjacent.

### 3.3 Clique decompositions

With the customary abuse of notation, the same symbol \( K \) is used for a maximal clique of \( G \) and its corresponding bag in a clique decomposition \( \mathcal{K} \) for \( G \). A complete graph on all maximal cliques of a graph gives a trivial clique decomposition for it, which is uninteresting. We are only interested in clique decompositions that can be stored and manipulated in linear time. Every clique decomposition \( \mathcal{K} \) in this paper will satisfy (1) \( ||\mathcal{K}|| \leq |\mathcal{K}| \leq |G| \); and (2) each vertex \( v \in G \) appears in at most \( d(v) \) bags. Since \( \mathcal{K} \) is connected, it either is a tree or has a unique cycle.
For example, a chordal graph $G$ has at most $|G|$ maximal cliques \[27\], which can be arranged as a tree such that for every $v \in G$, the set of maximal cliques containing $v$ induces a subtree \[12\]. Interval graphs are chordal, and thus admit clique tree decompositions as well. Fulkerson and Gross \[33\] showed that an interval graph always has a clique path decomposition. Although a circular-arc graph can have exponential number of maximal cliques in general \[84\], it becomes more amicable when it is Helly. A Helly circular-arc graph $G$ has at most $|G|$ maximal cliques and admits a clique decomposition that is either a path or a cycle \[36\].

For any simplicial vertex $v$, the clique induced by $N[v]$ must be maximal. This observation allows us to find all simplicial vertices of a graph by traversing its clique decomposition: We count the occurrences of all vertices, and return those vertices with number 1. This approach runs in time $\sum_{K \in \mathcal{K}} |K| = O(|G|)$, and works for all aforementioned classes. In particular, noting that a clique tree for a chordal graph can be built in linear time, we have

**Lemma 3.4.** We can in $O(|G|)$ time find all simplicial vertices of a chordal graph $G$.

A path of at least four bags can be made an hole by adding an edge connecting its end bags: adding an extra edge to the path does not break any condition in the definition of clique decomposition. For the same reason, a caterpillar decomposition whose central path has at least four bags can be viewed as an olive-ring decomposition.

We point out that clique tree decomposition for chordal graphs has different formulations, all of which are equivalent; see \[5\] for a comprehensive survey. The definition we use here, not relying on the fact that a clique tree for a chordal graph can be built in linear time, we have

**Proposition 3.5.** Given a clique hole decomposition $\mathcal{K}$ for a graph $G$, an interval model for $G$ can be obtained by setting $I_v = [\left(v \right) - 1/3, \right(v) + 1/3$ \[4\].

In a circular-arc model, each vertex $v$ corresponds to a closed interval $I_v = [lp(v), rp(v)]$. Here $lp(v)$ and $rp(v)$ are the left and right endpoints of $I_v$, respectively, and $lp(v) < rp(v)$. For a subset $U$ of vertices, we define $lp(U) = \min_{u \in U} lp(u)$ and $rp(U) = \max_{u \in U} rp(u)$. Observe that if $U$ induces a connected subgraph, then the union of $\{I_v : v \in U\}$ also forms an interval, which is $[lp(U), rp(U)]$. For an interval graph, clique path decompositions and interval models are related by the following observation. We use $\left(v\right)$ (resp., $\right(v)$) to denote the index of the smallest (resp., largest) index of bags that contain $v$. Here we use $\pm 1/3$ to force $lp(v) \neq rp(v)$ for every vertex $v$; any positive constant strictly less than $1/2$ will serve this purpose.

**Proposition 3.6.** Given a clique hole decomposition $\mathcal{K}$ for a graph $G$, a circular-arc model with circle length $|\mathcal{K}|$ for $G$ is given by

$$A_v = \begin{cases} \left[\left(v\right) - 1/3, \right(v) + 1/3 \right] & \text{if } \left(v\right) > 0, \\ \left [|\mathcal{K}| - 1/3, \right(v) + 1/3 \right] & \text{if } \left(v\right) = 0. \end{cases}$$

We will use $\ell \in [p, q]$ to denote the fact that the point $\ell$ is contained in the interval or arc $[p, q]$. If $p \leq q$, then $p \leq \ell \leq q$. This is the only case in an interval model. In a circular-arc model, when $p > q$, then either $\ell \geq p$ or $0 \leq \ell \leq q$. This has a natural generalization, i.e., $[p', q'] \subseteq [p, q]$.

\footnote{Although these two representation can be interchanged in $O(|G|)$ time, there is a fundamental difference. It takes $\Theta(|G|)$ space to store a clique decomposition, while $\Theta(|G|)$ suffices for an interval model. The latter is thus more space efficient.}
An interval model is normalized if no pair of distinct intervals shares an endpoint. Note that the model given by Prop. 3.5 is not normalized. In a normalized model for graph G, there are always 2|G| distinct endpoints. For any point p in a normalized interval model I, we can find a positive value ϵ such that the only possible endpoint of p in [p − ϵ, p + ϵ] is p. Here the value of ϵ should be understood as a function (depending on the interval model as well as the point p) instead of a constant, e.g., after the interval model is mutated, we might need a smaller value of ϵ for next operation.

4 Asteroidal triples in prime locally interval graphs

This section is devoted to the study of large CAWs and long holes, both of which are minimal witnesses of asteroidal triples in prime locally interval graphs. We prove Thms. 2.1 and 2.2 and derive some of their implications. In this section, we will use H(v) as a shorthand for N[v] ∩ V(H), i.e., the closed neighborhood of v in a hole H. The notation B+(v), where B+ is the (l, r)-path (lb1 · · · bdtr), is defined analogously.

4.1 Shallow terminals of chordal asteroidal witnesses

Theorem 2.1 (restated). Let W be a large CAW of a prime graph G. We can in O(|G|) find a subgraph of G in F(L1) if the shallow terminal of W is non-simplicial in G.

Proof. Let (s : c1, c2 : l, B, r) be the CAW. Before presenting the main procedure of this proof, we introduce two subroutines. They apply to two special structures that arise frequently in the main procedure. The outcomes of both subroutines are always subgraphs in F(l1); hence the detection of either of these two structures will suffice to terminate the main procedure. Both structures involve some vertex x ∈ N(s) \ {c1, c2}. Note that c1, c2 ∈ N(s) and x /∈ B.

In the first structure x is nonadjacent to one or both of c1 and c2. We apply subroutine A (Fig. 6) when x /∈ c2, and the case x /∈ c1 can be handled in a symmetric way.

![Figure 6: Subroutine A for the proof of Thm. 2.1](image)

The second structure has a maximal sub-path (b · · · bq) of B+(x) that contains at most two vertices of B, i.e., q − p ≤ 2 and the equality can only be attained when p = 0 or q = d + 1. (Recall that b0 and bd+1 do not belong to B.) The maximality implies that if b(p, resp., bq) is not an end of B +, i.e., p ≥ 1 (resp., q ≤ d), then x /∈ b(p−1) (resp., x /∈ bq+1). We apply subroutine B (Fig. 7).

![Figure 7: Subroutine B for the proof of Thm. 2.1](image)

It is easy to verify that both subroutines correctly returns in time O(|G|) a subgraph in F(L1). Now we are ready to present the main procedure (Fig. 8).
Let us verify the correctness of the main procedure. Step 1 searches for a special large CAW in a local and greedy way. Some of its iterations might update the CAW \( W \), and when it ends, the following conditions are satisfied by \( W \): (1) its shallow terminal is still \( s \); (2) its base is a subset of the base of the original CAW given in the input; and (3) if a vertex \( x \in N(s) \) is adjacent to the base, then it is a common neighbor of it. In the progress of this step, (1) and (2) will always be satisfied by the current CAW, while (3) is satisfied by the set \( C \) of vertices in \( N(s) \) that have been explored: in the progress, \( C \) is the set of common neighbors of \( s \) and the current base. It is clear that this holds true initially, when only the center(s) of \( W \) are explored, both in \( C \). Each iteration of the for-loop explores a new vertex \( x \) in \( N(s) \). A vertex nonadjacent to \( B^+ \) satisfies all three conditions vacuously; hence omitted (step 1.1). If one of two aforementioned structures is found, the procedure calls either subroutine \( A \) (step 1.2) or \( B \) (step 1.4). Otherwise it updates \( W \) accordingly (steps 1.5 and 1.6). It is easy to verify that \( W \) is a valid CAW. Moreover, after each time \( W \) is updated, the new base is a subset of the previous one; hence (3) remains true for all explored vertices with respect to the new CAW. After step 1, every neighbor of \( s \) is either in \( C \), which is \( N(s) \cap N(B) \), or nonadjacent to \( B \).

Step 2 runs a check to ensure that \( C \) induces a clique, which is straightforward. Step 3 then applies BFS to find vertex \( x \) that is either adjacent to \( B^+ \) or nonadjacent to some vertex \( y \in C \). The existence of such a vertex can be argued by contradiction. Let \( M \) be the component of \( G - C \) that contains \( s \). Suppose, for contradiction, that \( M \) is completely adjacent to \( C \), then \( M \) is a module of \( G \). Since \( G \) is prime, we must have \( M = \{s\} \), and then \( s \) is simplicial in \( G \), contradicting the assumption. Clearly, \( x \) cannot be \( s \); hence \( \text{prev}(x) \) is well-defined. This verifies step 3.

Based on which condition \( x \) satisfies, the procedure enters one of steps 4 and 5. Note that \( x \) is not in \( C \); hence if it is adjacent to \( B^+ \), then \( \text{prev}(x) \) cannot be \( s \). In other words, \( u \) is defined in step 4. By assumption, \( v \) is adjacent to every vertex in \( C \) but nonadjacent to \( B^+ \), which means that \( (v : c_1, c_2 : l, B, r) \) is a CAW isomorphic to \( W \). This holds for \( u \) as well in step 4. Steps 4.1-4.3 and 5.1-5.2 are straightforward. For step 4.4, note that \( u \neq x \).

We now analyze the runtime of the main procedure. Note that subroutines \( A \) and \( B \) can be called at most once, which terminate the procedure. The dominating step in the for-loop of step 1 is finding the sub-path (step 1.3), which takes \( O(d(x)) \) time for each \( x \) (Lemma 3.1). In total, step 1 takes \( O(|G|) \) time. The condition of step 3, i.e., whether \( x \) is adjacent to \( B^+ \) and a common neighbor of \( C \), can be checked in \( O(d(x)) \) time; hence step 3 can be done in \( O(|G|) \) time. Steps 2, 4, and 5 are straightforward and all can be done in \( O(|G|) \) time. This completes the runtime analysis and the proof.
As a result, if a prime locally interval graph $G$ is chordal, then $G = SI(G)$ is an interval graph.

### 4.2 Holes

This subsection is focused on holes, and more specifically, their relation with other vertices in a prime graph. Let $H$ be a given hole of a prime graph $G$; we may assume $|H| \geq 6$. We start from characterizing $H(v)$ for every $v \in G$: we specify some forbidden structures not allowed to appear in a prime locally interval graph, and more importantly, we show how to find a subgraph of $G$ in $\mathcal{F}_{11}$ if one of these structures exists.

**Lemma 4.1.** For every vertex $v$, we can in $O(d(v))$ time decides whether or not $H(v)$ induces a (possibly empty) sub-path of $H$, and if yes, in the same time find the ends of the path. Otherwise, we can in $O(|G|)$ time find a subgraph of $G$ in $\mathcal{F}_{11}$.

**Proof.** We call Lem. 3.1 to obtain the ordered list IND of indices of $H(v)$ in $H$. If IND is empty, i.e. $v \not\sim V(H)$, then we return a empty path with no vertex. Hereafter IND is assumed to be nonempty. We consider first the case where IND contains at most $|H| - 1$ elements. Let $p$ and $q$ be the first and last elements, respectively, of IND. Starting from the first element $p$, we traverse IND to the end for the first $i$ such that IND$[i] > IND[i] + 1$. If no such $i$ exists, then we return $(h_p, \ldots, h_q)$ as the path $P$. In the remaining cases, we may assume that we have found the $i$; let $p_1 = IND[i]$ and $p_2 = IND[i + 1]$. We continue to traverse from $i + 1$ to the end of IND for the first $j$ such that IND$[j + 1] > IND[j] + 1$. This step has three possible outcomes: (1) if $j$ is found, then $p_3 = IND[j]$ and $p_4 = IND[j + 1]$; (2) if no such $j$ is found, and at least one of $q < |H| - 1$ and $p > 0$ holds, then $p_3 = q$ and $p_4 = p + |H|$; and (3) otherwise ($p = 0$, $q = |H| - 1$, and $j$ is not found).

In the third case, we return $(h_p, h_{p - 1}, \ldots, h_0)$ as the path induces by $H(v)$. In the first two cases, $p_3$ and $p_4$ are defined, and $p_4 > p_3 + 1$. In other words, we have two nontrivial sub-paths, $(h_{p_1} h_{p_2} \ldots h_{p_3}, h_{p_4})$ of $H$ such that $v$ is adjacent to their ends but none of their inner vertices. We then call subroutine A (Fig. 9), whose correctness is straightforward.

![Figure 9: Subroutine A for the proof of Lem. 4.1](image)

Assume now that IND contains all $|H|$ elements $\{0, 1, \ldots, |H| - 1\}$. Let $C$ be the set of common neighbors of $V(H)$, which is nonempty. We find $C$ by using Lem. 3.1 to check each vertex for its neighbors in $H$. Starting from $H$, we apply BFS in $G - C$ to find the first vertex $u$ such that $u \not\sim x$ for some vertex $x \in C$. The existence of such a pair of vertices can be argued by contradiction. Let $M$ be the component of $G - C$ that contains $H$. Suppose, for contradiction, that $M$ is completely adjacent to $C$, then $M$ is a nontrivial module of $G$, which is impossible.

Let $(u_0 \ldots u_q)$ be the searching path that leads from $u_0 \in H$ to $u_q = u$; that is, $u_i = \text{prev}(u_{i+1})$ for $0 \leq i < q$. Note that $u_1 \not\in C$. We find the path induced by $H(u_1)$, which is nonempty and proper; otherwise we can use the previous case. Based on the value of $q$, we proceed as follows (Fig. 10).

![Figure 10: Subroutine B for the proof of Lem. 4.1](image)

The list IND can be constructed in $O(d(v))$ time using Prop. 3.1. We can traverse it and find its ends in the same time if it induces a path. We now consider the other situations, and analyze the runtime of finding subgraphs in $\mathcal{F}_{11}$. When $|H(v)| < |H|$ the detection of a subgraph in $\mathcal{F}_{11}$ can also be done in $O(d(v))$ time:
the main step is to traverse IND to obtain the indices \(p_1, \ldots, p_4\), which can be done in \(O(d(v))\) time, while the rest uses constant time. The dominating step of the last case is the construction of \(C\), which takes \(O(|\|G|||)\) time: the test of \(|H(u)| = |H|\) for each vertex (again using Prop. 3.1) takes \(O(d(u))\) time. All other steps use constant time. This concludes the time analysis and completes the proof.

Now let \(v\) be a vertex such that \(H(v)\) induces a path \(P\). We can assign a direction to \(P\) in accordance to the direction of \(H\), and then we have clockwise and counterclockwise ends of \(P\). For technical reasons, we assign canonical indices to the ends of the path \(P\) as follows.

**Definition 3.** For each vertex \(v\) with nonempty \(H(v)\), we denote by first \((v)\) and last \((v)\) the indices of the counterclockwise and clockwise, respectively, ends of the path induced by \(H(v)\) in \(H\) satisfying

- \(|H| < \text{first}(v) \leq 0 \leq \text{last}(v) < |H|\) if \(h_0 \in H(v)\); or
- \(0 < \text{first}(v) \leq \text{last}(v) < |H|\), otherwise.

It is possible that \(\text{last}(v) = \text{first}(v)\), when \(|H(v)| = 1\). In general, \(\text{last}(v) - \text{first}(v) = |H(v)| - 1\) and \(v = h_i\) or \(v \sim h_i\) for each \(i\) with \(\text{first}(v) \leq i \leq \text{last}(v)\). The indices \(\text{first}(v)\) and \(\text{last}(v)\) can be easily retrieved from Lem. 4.1 and with them we can check the adjacency between \(v\) and any vertex \(h_i \in H\) in constant time, even when \(v \not\sim H\). (For example, with the definition of \(\text{first}(v)\) and \(\text{last}(v)\), we may represent the fact \(v \not\sim V(H)\) by \(\text{first}(v) > \text{last}(v)\).) We also remark that \(\text{last}(h_1)\) is unnecessarily \(i + 1\).

If \(v\) is adjacent to \(|H| - 2\) or \(|H| - 1\) vertices in \(H\), then it is trivial to find a short hole \(V(H) \cup \{v\}\). In the rest of this paper, whenever we meet a hole and a vertex such that \(|H(v)| < |H| - 2\) or \(H(v)\) is nonconsecutive, we either return a short hole or call Lem. 4.1. To avoid making the paper unnecessarily ponderous, we will tacitly assume otherwise. We now turn to the vertices that are nonadjacent to \(V(H)\), which, together with Lem. 4.1 and discussion above, concludes the proof of Thm. 2.2.

**Lemma 4.2.** Given a non-simplicial vertex \(v\) that is nonadjacent to \(H\), we can in \(O(|\|G|||)\) time find a subgraph of \(G\) in \(\mathcal{I}_L\).

**Proof.** We may assume, without loss of generality, that some neighbor \(u\) of \(v\) is adjacent to \(V(H)\): otherwise we can find (by BFS) a shortest path from \(v\) to \(H\) and take the last two inner vertices from this path as \(v\) and \(u\), respectively; in particular, as an inner vertex of a chordless path, the new vertex \(v\) is necessarily non-simplicial. We return long claw \(\{v, u, h_{\text{first}(u) - 2}, \ldots, h_{\text{first}(u) + 2}\}\) if \(|H(u)| = 1\); or net \(\{v, u, h_{\text{first}(u) - 1}, h_{\text{first}(u)}, h_{\text{last}(u)}, h_{\text{last}(u) + 1}\}\) if \(|H(u)| = 2\). Otherwise, \(|H(u)| \geq 3\), and we can call Thm. 2.1 with large CAW \(\langle v : u, u : h_{\text{first}(u) - 1}, h_{\text{first}(u)}, \ldots h_{\text{last}(u)}, h_{\text{last}(u) + 1}\rangle\). Here we are using the assumption that \(H(u)\) induces a path of at most \(|H| - 3\) vertices and the fact that \(v\) is not simplicial in \(G\). The dominating step is finding the appropriate vertices \(v, u\), which takes \(O(|\|G|||)\) time.

Now consider the neighbors of more than one vertices in \(H\).

**Lemma 4.3.** Given a pair of adjacent vertices \(u, v\) such that \(H(u)\) and \(H(v)\) are both nonempty and disjoint, we can in \(O(|\|G|||)\) time find a subgraph of \(G\) in \(\mathcal{I}_L\).

|   |   |
|---|---|
| 1 | if \(v \sim (h_{\text{last}(u) - 1}, h_{\text{last}(u) + 2})\) then return a short hole; |
| 2 | if \(v \sim (h_{\text{first}(u) - 2}, h_{\text{first}(u) - 1})\) then return a short hole; |
| 3 | if \(|H(u)| = 1\) then return long claw \(\{v, u, h_{\text{first}(u) - 2}, \ldots, h_{\text{first}(u) + 2}\}\); |
| 4 | if \(|H(u)| = 2\) then return net \(\{v, u, h_{\text{first}(u) - 1}, h_{\text{first}(u)}, h_{\text{last}(u)}, h_{\text{last}(u) + 1}\}\); |
| 5 | call Thm. 2.1 with \(\langle v : u, u : h_{\text{first}(u) - 1}, h_{\text{first}(u)}, \ldots h_{\text{last}(u)}, h_{\text{last}(u) + 1}\rangle\). |

**Proof.** Clearly, neither of \(u\) and \(v\) can be in \(H\). We use the procedure above. Steps 1-3 are straightforward. For step 4, note that \(v\) is non-simplicial because its neighbors in \(H\) and \(u\) are nonadjacent.

In other words, for any pair of adjacent vertices, if neither \(H(u)\) nor \(H(v)\) is a subset of the other, then at least one of \(h_{\text{last}(v)}\) and \(h_{\text{first}(v)}\) needs to be in \(H(u)\). Lem. 4.3 can be extended to more than two vertices as follows. Here \(H(U) = \bigcup_{v \in U} H(v)\).

**Corollary 4.4.** Given a set of vertices \(U\) such that \(G[U]\) is connected and \(H(U)\) is not consecutive in \(H\), we can in \(O(|\|G|||)\) time find a subgraph of \(G\) in \(\mathcal{I}_L\).
Proof. For two disjoint sub-paths of $H(\cup)$, we can find a pair of vertices $u_1, u_2 \in U$ such that they are adjacent to the paths respectively; then traversing a $(u_1, u_2)$-path in $G[\cup]$ will give a pair of vertices satisfying Lem. 4.3.

The last lemma of this subsection is related to the Helly property, which is crucial for the olive-ring decomposition of next section.

Lemma 4.5. Given a set $U$ of two or more pairwise adjacent vertices such that $H(U) = V(H)$, we can in $O(|G|)$ time find a subgraph of $G$ in $F_{1,1}$.

Proof. We start from an arbitrary vertex $u_1$ of $U$. Without loss of generality, we may assume $u_1 \sim H$, and in particular, Lem. 4.3 returns a proper sub-path; otherwise we are done. There must be another vertex in $U$ that is adjacent to $h_{\text{last}(u_1)}+1$; let it be $u_2$. We proceed as Fig. 11. The correctness and runtime of this procedure are straightforward.

5 Characterization and decomposition of prime locally interval graphs

Recall that if $G$ is a prime locally interval graph, then by Thm. 2.1 any non-interval subgraph of $G - SI(G)$ is a hole. If $G$ is chordal, then it suffices to check whether $G - SI(G)$ is an interval graph or not: if not, we can use Thm. 2.1 to find a subgraph in $F_{1,1}$; otherwise, we jump directly to Section 5.3 to build the caterpillar decomposition. Therefore, in the first two subsections, we are exclusively concerned with non-chordal graphs. Recall that in a non-chordal graph, a hole can be found in linear time [82]. Section 5.1 gives the details on the construction of the auxiliary graph $\bar{U}(G)$ for a non-chordal graph $G$. Section 5.2 proves Thm. 2.3. Section 5.3 exploits $\bar{U}(G)$ to build the olive-ring decomposition for $G$.

5.1 The auxiliary graph $\bar{U}(G)$

If a vertex $v$ is adjacent to four or more consecutive vertices in a hole $H$, i.e., $\text{last}(v) - \text{first}(v) > 2$, then $v \notin H$. We can thus use $(h_{\text{first}(v)} \cup h_{\text{last}(v)})$ as a short cut for the sub-path induced by the neighbors of $v$ in $H$, thereby yielding a strictly shorter hole. To simplify the later presentation, we would like that $h_0$ cannot be bypassed as such. The following lemma formally states this condition and gives a procedure for finding a hole satisfying it.

Lemma 5.1. In $O(|G|)$ time, we can find either a subgraph of $G$ in $F_{1,1}$, or a hole $H$ such that for every vertex $v \in N[h_1] \cap N[h_2]$ it holds that (i) $v$ is adjacent to neither $h_{-2}$ nor $h_2$; and (ii) $N[v] \subseteq N[H]$.

Proof. We apply the procedure given in Fig. 12. Several steps of it might end with a subgraph in $F_{1,1}$, and in this case, we terminate this procedure by returning this subgraph (remark *). Step 1 greedily searches for a pair of indices $a$ and $b$ such that $a < 0 < b$ and $[h_a, h_{a+1}, \ldots, h_b]$ is not properly contained in $N[v]$ for any vertex $v$. The set $C$ stores all vertices $v$ satisfying $\text{first}(v) = a$ and $\text{last}(v) = b$. Initially, $a = -1$ and $b = 1$, while $C$ consists of only $h_0$. Vertices in $H$ are considered explored before step 1, and each iteration of step 1 checks a vertex $v$ that has not been explored. If either condition of step 1.2 is satisfied, then $N[v]$ properly contains $[h_a, h_{a+1}, \ldots, h_b]$, and $a$ and $b$ are updated to $\text{first}(v)$ and $\text{last}(v)$ respectively. Note that the values of $a$ and $b$ are non-increasing and nondecreasing respectively. After each update of $a$ or $b$, no vertex in $C$ is adjacent to all of $[h_a, \ldots, h_b]$, and hence they are purged from $C$. No previously explored vertex can be adjacent to all of $[h_a, h_{a+1}, \ldots, h_b]$ either. Therefore, now $C$ consists of only $v$. Step 1.3 puts into $C$ those vertices whose closed neighborhood in $H$ is precisely $[h_a, h_{a+1}, \ldots, h_b]$. At the end of step 1,
taking any \( h \in C \), the hole \((h_h, h_{b+1}, \ldots, h_a)\) will satisfy (i). If one or both of \( a \) and \( b \) have been updated, then this hole is strictly shorter than \( H \). Step 2 makes sure that the length of this hole is at least 6.

The rest of the procedure will be devoted to finding a vertex from \( C \) that satisfies (ii) as well. Note that the final output of the procedure will be a hole \((h_0, h_{b+1}, \ldots, h_a)\) for some \( h \in C \), and the new hole will be formed in a way that the vertices \( h_0 \) and \( h_a \) are the new \( h_0 \) and \( h_1 \) respectively. The neighbors of \( h \) in the new hole will be \((h_0, h_a)\), and \( C = \{h_0\} \cup \{h_a\} \). We hence verify (ii) for all vertices in \( C \). For each \( v \in C \), let \( N'(v) \) denote the set of vertices that are adjacent to \( v \) but not to \( \{h_0, h_{b+1}, \ldots, h_a\} \); a vertex \( u \in N'(v) \) is nonadjacent to \( V(H) \), as otherwise we can apply Lem. 4.3 to \( u \) and \( v \).

Before the for-loop of step 4, \( \text{old} = 0 \), and \( h \) is undefined. After the first vertex \( v \) such that \( N'(v) \) is nonempty is checked, the value of old becomes positive and the vertex \( h \) is set to \( v \); thereafter, the value of old remains positive (it is nondecreasing) and \( h \) remains defined. In other words, step 4 maintains the following invariants: the value of \( \text{old} \) is 0 or \( |N'(h)| \), and when \( h \) is defined, \( N'(h) \neq \emptyset \) and \( N'(u) \subseteq N'(h) \) for every previously explored vertex \( u \).

Each iteration of step 4 checks one vertex \( v \in C \), which sets two variables. At the end of this iteration, \( \text{new} = |N'(v)| \), and \( \delta = |N'(v)| \cdot |N'(h)| \) (or \(|N'(v)| \) when \( h \) has not been defined). If \( \delta = 0 \), then the invariant remains true and the procedure proceeds to next vertex in \( C \). Otherwise, \( N'(v) \setminus N'(h) \) is nonempty, and then it enters step 4.3, which has two cases. If \( N'(h) \setminus N'(v) \) is also nonempty (i.e., \( \text{new} < \text{old} + \delta \)), then a net \( \{h_{b-1}, h_a, v, x, y\} \) can be found by taking \( x \in N'(h) \setminus N'(v) \) and \( y \in N'(v) \setminus N'(h) \). In the remaining case, \( N'(h) \subseteq N'(v) \), and the vertex \( h \) and the value of \( \text{old} \) are updated to \( v \) and \( \text{new} \), respectively.

After step 4, if \( h \) is defined, then \( \bigcup_{v \in C} N'(v) \subseteq N'(h) \), and the hole given in step 5 satisfies (ii). Otherwise, \( \bigcup_{v \in C} N'(v) \) is empty, and any vertex from \( C \) can be used to make the hole.

Let us now analyze the runtime. What dominates step 1 is finding first(v) and last(v) for all vertices (step 1.1), which takes \( O(d(v)) \) time for each vertex \( v \) and \( O(|G|) \) time in total. Steps 2 and 3 take \( O(1) \) and \( O(|G|) \) time respectively. Step 4.2 dominates step 4, and it takes \( O(|G|) \) time in total. Therefore, the whole procedure can be implemented in \( O(|G|) \) time.

Recall that \( T = N[h_0] \) and \( \bar{T} = V(G) \setminus T \). For a pair of adjacent vertices \( v \in T \) and \( u \in \bar{T} \), the direction of edge \( uv \) is determined as follows. We may assume that neither of Lemms. 4.1 and 4.3 applies to \( v \) and \( u \), as otherwise we can terminate the algorithm safely. If \( v \) is adjacent to neither \( h_{b-1} \) nor \( h_1 \), then we must have \( u \not\sim H \) (otherwise Lem. 4.3 applies), which means that \( \{v, u, h_{b-2}, h_{b-1}, h_0, h_1, h_2\} \) forms a long claw. Therefore, we may assume that \( v \) is adjacent to at least one of \( h_{b-1} \) and \( h_1 \), which gives three cases. Observer also that \( u \not\in N[h_0] \), and hence cannot be adjacent to both \( h_{b-1} \) and \( h_1 \) (Lemms. 4.1 and 4.5).
In this proof we consider the set \( \text{the auxiliary graph} \) of \( h \). Here the only uncertain adjacencies are between \( u \) and \( v \). For (1), consider first that \( uv \sim h \). Then by definition of \( E \) and \( E \), it holds that \( v \sim h_1 \) and \( v_2 \not\sim h \). We can return net \( \{u, v_1, v_2, h_1, h_1 \} \) if \( v_1 \sim v_2 \) or 4-hole \( (uv_1h_0v_2u) \) otherwise.

For (2), consider first that \( v_1 = v_2 \). Then by definition of \( E \) and \( E \), \( v_1 \) is adjacent to both \( h \) and \( h_1 \), and it follows that \( u_1 \sim h_1 \) and \( u_2 \sim h_1 \). Observing that \( u_1 \sim u_2 \) but neither of \( u_1 \) and \( u_2 \) is adjacent to \( h_0 \), at least one of Lems. [4.1] and [4.5] applies. Assume now that \( v_1 \neq v_2 \), and without loss of generality, \( u_1 \neq v_2 \) and \( u_2 \neq v_1 \). Then we can return \( (u_1v_1v_2u_2u_1) \) or \( (u_1v_1h_0v_2u_2u_1) \) as a short hole.

Let \( T \) denote the subset of vertices of \( T \) that are incident to edges in \( E \) (resp., \( E \)). Note that \( \{E_0, E \} \) partitions edges between \( T \) and \( T \), but a vertex in \( T \) might belong to both \( T \) and \( T \), or neither of them. By definition, edges \( h_1h_2 \in E \) and \( h_1h_2 \in E \); while \( T \subset N[h] \) and \( T \subset N[h] \). We have the following observation on \( T \) and \( T \).

**Lemma 5.3.** Given a pair of nonadjacent vertices \( u, x \in T \) (or \( T \)), we can in \( O(|G|) \) time find a subgraph of \( G \) in \( T \).

**Proof.** In this proof we consider the set \( T \), and a symmetrical argument applies to \( T \). By definition, we can find edges \( uv, xy \in E \), where \( v, y \in \). We have three (possibly intersecting) chordless paths \( h_0h_1h_2 \), \( h_0uv \), and \( h_0xy \). Since \( u \sim y \), they cannot be both adjacent to \( h_1 \) (Lem. 5.1). Hence we may assume, without loss of generality, \( u \not\sim h_1 \).

Assume first that \( u \sim h_1 \). We consider the subgraph induced by \( X_1 = \{h_0, h_1, h_2, u, v, x\} \), which is clearly distinct. Here the only uncertain adjacencies are between \( v, x, h_2 \); by assumption, \( h_0h_1, h_1h_2 \), and \( u \) are pairwise adjacent; \( x \) is adjacent to neither \( u \) nor \( h_1h_2 \); \( h_0h_1 \) and \( u \) are both adjacent to \( h_1 \) (because Lem. 5.1 by assumption, \( N[u] \cap V(H) = \{h_1, h_0, h_1\} \) and \( v \not\sim h \)); and \( v \) is adjacent to neither \( h_0 \) nor \( h_1 \). If \( v, x, h_2 \), they are pairwise nonadjacent, then we return \( G[X_1] \) as a net. Otherwise, there is at least one edge among \( v, x, h_2 \), then we return a 4-hole, e.g., \( (uxh_0uv) \) when the edge is \( vx \).

In the remaining cases, \( u, x, h_1 \), and \( h_2 \) are pairwise nonadjacent. If any two of \( v, y, h_2 \) are identical or adjacent, then we return a 4- or 5-hole, e.g., \( (h_0uxh_0) \) or \( (h_0uvxyh_0) \) when \( v = y \) or \( v \sim y \) respectively. Otherwise, \( v, y, h_2 \), and \( h_2 \) are distinct and pairwise nonadjacent, and we return long claw \( (h_0h_1h_2, u, v, x, y) \).

Edges \( uv \) and \( xy \) can be found in \( O(|G|) \) time, and only a small constant number of adjacencies are checked in this procedure; it thus takes \( O(|G|) \) time in total. □
Lemma 5.4. The order and size of $\Omega(G)$ are upper bounded by $2|G|$ and $2||G||$, respectively. Moreover, an adjacency list representation of $\Omega(G)$ can be constructed in $O(||G||)$ time.

Figure 13: Procedure for constructing $\Omega(G)$ (Lem. 5.4).

Proof. The vertices of the auxiliary graph $\Omega(G)$ include $T$, two copies of $T$, and $w$, i.e., $||\Omega(G)|| = 2|T| + |T| + 1 = |G| + |T| + 1 \leq 2|G|$. In $\Omega(G)$, there are two edges derived from every edge of $G[T]$ and one edge from every other edge of $G$. All other edges are incident to $w$, and there are $T_{cc}$ of them. Therefore, $||\Omega(G)|| = ||G|| + ||G[T]|| + |T_{cc}| \leq ||G|| + ||G|| + |E_{cc}| < 2|G||. This concludes the first assertion.

For the construction of $\Omega(G)$, we use the procedure described in Fig. 13 (some bookkeeping details are omitted). Step 1 adds vertex sets $L$ and $R$ (step 1.1) as well as those edges induced by them (step 1.2.1), and finds $N(T)$ (step 1.2.2). Step 2 adds edges in $E_{cc}$ and $E_{c}$, and detect $T_{cc}$ and $T_{c}$. Steps 2.2, 2.3, and 3 verify that neither of Lemmas 5.2 and 5.3 applies; information required in these verifications can be obtained in step 2.1 and stored. Steps 4 and 5 add vertex $w$ and edges incident to it. Step 6 cleans $T$. The main steps are 1 and 2, each of which checks every edge at most once, and hence the total time is $O(||G||)$.

By steps 2.2 and 2.3 (Lem. 5.2 as well as the discussion preceding it), and step 3 (Lem. 5.3), a posteriori, the following properties hold for $\Omega(G)$.

Proposition 5.5. In the auxiliary graph $\Omega(G)$, any path between $L$ and $R$ has length at least 4, and the vertex $w$ is simplicial.

5.2 Detection of subgraphs in $\mathcal{F}_{LI}$

Each vertex $x$ of $\Omega(G)$ different from $w$ is uniquely defined by a vertex of $G$, which is denoted by $\phi(x)$. We say that $x$ is derived from $\phi(x)$ and $\phi(x)$ is the origin of $x$. For example, $\phi(v^1) = \phi(v^2) = v$ for $v \in T$. By abuse of notation, we will use the same letter for a vertex $u \in T$ of $G$ and the unique vertex of $\Omega(G)$ derived from $u$; its meaning is always clear from the context. Therefore, $\phi(u) = u$ for $u \in T$, and in particular, $\phi(h_i) = h_i$ for $i = 2, \ldots, |H| - 2$. We can mark $\phi(x)$ for each vertex during the construction of $\Omega(G)$. The function $\phi(\cdot)$ is also generalized to a set $U$ of vertices that does not contain $w$, i.e., $\phi(U) = \{\phi(v) : v \in U\}$. We point out that possibly $|\phi(U)| \neq |U|$.

By construction of $\Omega(G)$, if a pair of vertices $x$ and $y$ (different from $w$) is adjacent in $\Omega(G)$, then $\phi(x)$ and $\phi(y)$ must be adjacent in $G$ as well. The converse is unnecessarily true, e.g., for any vertex $v \in T$ and...
edge \( uv \in E_c \), we have \( u \not\sim v^* \), and for any pair of adjacent vertices \( u, v \in T \), we have \( u^1 \not\sim v^* \) and \( u^* \not\sim v^1 \). We say that a pair of vertices \( x, y \) of \( \mathcal{U}(G) \) is a bad pair if \( \phi(x) \sim \phi(y) \) in \( G \) but \( x \not\sim y \) in \( \mathcal{U}(G) \). By definition, \( w \) does not participate in any bad pair, and at least one vertex of a bad pair is in \( L \cup R \).

**Proposition 5.6.** Given a bad pair of distance 3 or 4, we can in \( O(||G||) \) time find a subgraph of \( G \) in \( \mathcal{T}_{L_1} \).

**Proof.** Let \( x, y \) be a bad pair and without loss of generality, let \( x \in L \cup R \). Consider a set \( P \) of vertices that induces a shortest \((x, y)\)-path. We may assume that \( P \) contains no other bad pair. If \( x = v^1 \) for some \( v \in T \), then \( y \) is adjacent to \( v^* \); by Prop. 5.5, \( |P| \) is either 4 or 5. We return \( G[\phi(P)] \) as a short hole.

**Proposition 5.7.** Let \( X \) be a set of vertices of \( \mathcal{U}(G) \) that does not contain \( w \) or \( \{v^1, v^*\} \) for any \( v \in T \). Then \( \mathcal{U}(G)[X] \) is a subgraph of \( G[\phi(X)] \), and they are isomorphic if and only if \( X \) contains no bad pair.

**Proof.** By assumption, there is a one-to-one mapping between \( X \) and \( \phi(X) \). If \( X \) is free of bad pair, then this mapping also gives an isomorphism between \( \mathcal{U}(G)[X] \) and \( G[\phi(X)] \). On the other hand, if \( X \) contains bad pairs, then \( \mathcal{U}(G)[X] \) has strictly less edges than \( G[\phi(X)] \), and thus they cannot be isomorphic.

This observation enables us to prove the following lemma, which is crucial for the identification of simplicial vertices of \( G \). We use \( L_{cc} \) and \( L_c \) to denote the subset of vertices of \( L \) derived from \( T_{cc} \) and \( T_c \), respectively, i.e., \( L_{cc} = \{v^1 : v \in T_{cc}\} \) and \( L_c = \{v^1 : v \in T_c\} \).

**Lemma 5.8.** A vertex \( x \) different from \( \{w\} \cup R \) is simplicial in \( \mathcal{U}(G) \) if and only if \( \phi(x) \) is simplicial in \( G \).

**Proof.** Every vertex in \( L_{cc} \) is adjacent to both \( h_0^1 \) and \( w \), and thus cannot be simplicial in \( \mathcal{U}(G) \). Likewise, a vertex in \( T_{cc} \) is adjacent to \( h_0 \) and \( T \), and thus cannot be simplicial in \( G \). Therefore, we may assume \( x \not\in L_{cc} \), hence \( x \not\sim w \) and \( \phi(x) \not\in T_{cc} \). For such a vertex \( x \), any edge of \( G \) incident to \( \phi(x) \) has a corresponding edge of \( \mathcal{U}(G) \) incident to \( x \). In other words, there is a one-to-one mapping between the neighbors of \( x \) in \( \mathcal{U}(G) \) and the neighbors of \( \phi(x) \) in \( G \). By Prop. 5.7, if \( N_{\mathcal{U}(G)}(x) \) induces a clique (noting that it contains no \( \{v^1, v^*\} \) for any \( v \in T \) as they are nonadjacent), then \( G[\phi(X)] \) induces a clique as well. This verifies the “only if” direction.

Suppose that the “if” direction is false, then \( x \) must be adjacent to some bad pair; let it be \( y, z \). By Prop. 5.5, \( y \) and \( z \) cannot be both in \( L \cup R \) without loss of generality, let \( y \in T \). Then \( x \) is in \( T \) as well; otherwise, \( x \not\in L \) and \( \phi(x) \) is adjacent to two nonadjacent vertices \( y \) and \( h_0 \). Now \( \phi(z) \) is adjacent to both \( x \) and \( y \). The fact \( y \not\sim z \) in \( \mathcal{U}(G) \) implies \( \phi(z) \not\sim \phi(y) \) and \( \phi(y) \not\sim \phi(z) \) in \( E_{cc} \) and \( E_c \) but not the same. Without loss of generality, let \( \phi(z) \in E_{cc} \) and \( \phi(y) \in E_c \). Thus, \( z \in T_c \cap T_{cc} \), and \( x \) and \( y \) are adjacent to \( h_1 \) and \( h_1 \), respectively. However, \( \phi(x) \) has a pair of neighbors \( h_1 \) and \( y \) that is nonadjacent to each other. This contradiction concludes the “if” direction and the proof.

It is worth noting that even a vertex \( v \in T \) is not simplicial in \( G \), it is still possible that \( v^* \) is simplicial in \( \mathcal{U}(G) \). Such a vertex has to be in \( T_{cc} \), while the existence of \( w \) prohibits such a vertex in \( L \); for every vertex \( v \in T_{cc} \), the vertex \( v^* \) is adjacent to both \( h_0^1 \) and \( w \).

**Lemma 5.9.** Let \( X \) be a set of vertices of \( \mathcal{U}(G) \) that contains a bad pair and induces a connected subgraph. If there exists a vertex \( h \in H \) nonadjacent to \( \phi(X) \), then we can in \( O(||G||) \) time find a subgraph of \( G \) in \( \mathcal{T}_{L_1} \).

**Proof.** It suffices to assume that \( X \) is minimal, that is, it induces a path whose ends \( x, y \) are the only bad pair in \( X \). Then \( w \not\in X \): it does not participate in any bad pair, hence not an end of the path, and it is simplicial in \( \mathcal{U}(G) \), hence not an inner vertex of any shortest path. Recall that at least one of \( x \) and \( y \) is in \( L \cup R \); without loss of generality, let \( x = v^1 \in L \), then \( y \) is adjacent to the vertex \( v^* \). By Prop. 5.5, \( |X| \geq 4 \). As \( \{x, y\} \) is the only bad pair, \( \phi(X) \) induces a hole of \( G \), which enables us to call Lem. 4.2 with \( \phi(X) \) and \( h \).

Now we are ready to prove the rest of Thm. 2.3 which is separated into two statements.

**Lemma 5.10.** If \( \mathcal{U}(G) \) is not chordal, we can in \( O(||G||) \) time find a subgraph of \( G \) in \( \mathcal{T}_{L_1} \).

**Proof.** We find a hole \( C \) of \( \mathcal{U}(G) \). Recall that \( w \) is simplicial in \( \mathcal{U}(G) \) and participate in no hole. Let us first take care of two trivial cases. In the first case, \( C \) is disjoint from both \( L \) and \( R \), and \( \phi(C) \) is a hole of \( G \) (Prop. 5.7). This hole is nonadjacent to \( h_0 \) in \( G \), which enables us to call Lem. 4.2. In the other case, all vertices of \( C \) are from \( L \) or \( R \), and \( \phi(C) \) is a hole of \( G \). This hole has a common neighbor \( h_0 \), which enables us to call Lem. 4.1. Since \( L \) and \( R \) are nonadjacent, if \( C \) is disjoint from \( T \), then the second case must apply. Henceforth we assume that \( C \) intersects \( T \) and, without loss of generality, \( L \); it might intersect \( R \) as well, but this fact is irrelevant in the following proof.

By definition of \( \mathcal{U}(G) \) and Lem. 5.3, \( L_c \) is a clique separator of \( L \setminus L_c \) and \( T \). Therefore, \( C \) contains at most two vertices of \( L_c \) and is disjoint from \( L \setminus L_c \). We define two configurations based on whether \( C \) intersects \( L_{cc} \) or not (recall that a vertex can be in both \( L_c \) and \( L_{cc} \)):
I. If there exists \( x \in C \cap L_x \) (pick either one if \( |C \cap L_x| = 2 \)), then \( \text{first}(\phi(x)) = -1 \) and \( \text{last}(\phi(x)) = 1 \).

Starting from \( x \), we traverse \( C \) (in either direction) till the first vertex \( y \) that is adjacent to \( h_{-2} \).

II. Otherwise, \( \text{first}(\phi(x)) = 0 \) and \( \text{last}(\phi(x)) \geq 1 \) hold for both \( x \in C \cap L_x \). We pick \( x \) to be the vertex in \( C \cap L_x \) such that \( \text{last}(\phi(x)) \) is smaller. Starting from \( x \), we traverse \( C \) (in either direction) till the first vertex \( y \) that is adjacent to \( h_{-1}^x \).

We need to explain what to do if the vertex \( y \) is not found, which means that \( h_{-2} \) or \( h_{-1}^x \) is nonadjacent to the hole \( C \). We check whether \( C \) contains a bad pair or not. If yes, then we call Lem. 5.9; otherwise, \( \phi(C) \) induces a hole in \( G \) (Prop. 5.7). This hole is nonadjacent to \( h_{-2} \) (I) or \( h_{-1}^x \) (II), which allows us to call Lem. 4.2. In the following we may assume that we have found such a vertex \( y \).

Let \( a = \text{last}(\phi(x)) \); note that \( a < |H| - 2 \) (otherwise \( \phi(x) \) is adjacent to at least \( |H| - 2 \) vertices in \( H \)). If \( y \sim h_a^x \) (when \( a = 1 \)) or \( y \sim h_a \) (when \( a > 1 \)), then we have an \((h_d, h_r^x)\)-path \( h_d h_r^x y h_{-2}^x h_r^y \) (I) or \( h_d x h_a y h_{-1}^x h_r^y \) (II), which enables us to call Prop. 5.6.

Starting from \( x \), we traverse both directions of the hole \( C \) till the first vertices that are adjacent to \( h_{a+1} \). Let them be \( x_1 \) and \( x_2 \), and let \( P_1 \) and \( P_2 \) be the paths \((x \cdots x_1) \) and \((x \cdots x_2) \), respectively. If Lem. 4.1 or 4.3 applies in the traversal, then we are done. Otherwise, for every inner vertex in \( P_1 \) and \( P_2 \), its neighbors in \( \{h_{-3}^1, h_{-2}^1, \ldots, h_{-1}^x, h_r^y\} \) are subsets of \( \{h_{-1}^1, h_{-2}^1, \ldots, h_{-1}^x\} \), and both \( x_1 \) and \( x_2 \) are adjacent to \( h_a \). Therefore, \( y \) appears in neither path, which implies that \( x_1 \neq x_2 \) and \( x_1 \neq x_2 \). We take the hole \( C' = (x \cdots x_1 h_{a+1} x_2 \cdots x) \). Every vertex in \( C' \) is adjacent to \( h_r^x \) (or \( h_a^x \)), there is no bad pair in \( C' \), and thus \( \phi(C') \) is a hole of \( G \) (Lem. 5.7).

Noting that \( h_a \) is a common neighbor of this hole, we can call Lem. 4.1.

The main step is traversing \( C \), which can be done in \( O(|G|) \) time. This concludes this proof.

Now we may assume that \( \overline{U}(G) \) is chordal, and we use Lem. 3.4 to find the set \( S \) of simplicial vertices of \( \overline{U}(G) \). According to Lem. 5.3 \( \phi(S \setminus \{w \cup R\}) \) gives the set of simplicial vertices of \( G \). It is easy to verify that \( H \) remains a hole of \( G - SI(G) \), and more importantly, the conditions of Lem. 5.1 remain satisfied by \( H \) in \( G - SI(G) \). Therefore, \( \overline{U}(G - SI(G)) \) is also well-defined with respect to \( H \), and is an induced subgraph of \( \overline{U}(G) \). The graph can be constructed by removing all vertices derived from \( SI(G) \) from \( \overline{U}(G) \). All results in this subsection apply to \( \overline{U}(G - SI(G)) \); in particular, \( w \) remains simplicial. We point out that \( \overline{U}(G - SI(G)) \) is different from \( \overline{U}(G - SI(\overline{U}(G))) \).

**Lemma 5.11.** If \( \overline{U}(G - SI(G)) \) is not an interval graph, then we can in \( O(|G|) \) time find a subgraph of \( G \) in \( \mathcal{T}_1 \).

**Proof.** By assumption, we can find a subgraph of \( \overline{U}(G - SI(G)) \) in \( \mathcal{T}_1 \). If it is a hole, then we call Lem. 5.10 (noting that \( \overline{U}(G - SI(G)) \) is an induced subgraph of \( \overline{U}(G) \)). Hence we may assume that \( \overline{U}(G - SI(G)) \) is chordal, and we have a CAW of \( \overline{U}(G - SI(G)) \); let it be induced by \( X \). Since the largest distance between any pair of vertices in a CAW is 4, if \( X \) contains a bad pair, then we can call Prop. 5.6. Now that \( X \) is free of bad pairs, if \( w \not\in X \), then by Prop. 5.7 \( \phi(X) \) induces a CAW in \( G - SI(G) \). We can return \( \phi(X) \) if it is small, or call Thm. 2.1 otherwise. Therefore, in the remaining cases, \( X \) contains \( w \). As a result, \( X \) must intersect \( L \), or more specifically, \( L_{cc} \). The nonexistence of bad pairs implies that \( X \) is disjoint from \( R \): it is connected, and if it intersects \( R \), then it contains some vertex \( v^+ \) for \( v \in T_{cc} \); since \( T_{cc} \) induces a clique, we have a bad pair.

Since \( w \) is simplicial in \( \overline{U}(G - SI(G)) \), it has to be a terminal of the CAW and has either one or two neighbors in \( X \) (see Fig. 1). We search for a vertex \( u \in T \) such that \( u \phi(x) \in E_{cc} \) for every neighbor \( x \) of \( w \) in \( X \). We break the rest of the proof into two cases based on whether there exists such a vertex.

Assume first that such a vertex \( u \) is found. (This case is very similar as the case where \( w \not\in X \) as discussed above.) Note that this is the only case when \( |N(w) \cap X| = 1 \). It is easy to verify that \( u \) and \( x \in N(w) \cap X \) make a bad pair, which means \( u \not\in X \). By assumption, \( u \) is adjacent to \( R \), and thus nonadjacent to \( L \) (Lem. 5.2). If \( X \) contains a neighbor \( u' \) of \( u \), then we consider the shortest \((x, u')\)-path \( P \) in the CAW. Since \( x \) is a neighbor of \( w \), it cannot be a terminal. From Fig. 1 it can be observed that \( P \) consists of at most 4 vertices. We can extend \( P \) by adding edge \( u'u \), and this results in an \((x, u)\)-path of length at most 4, which allows us to call Prop. 5.6. Otherwise, \( X \) is disjoint from \( N(u - SI(G)) \), let \( X' = X \setminus \{w \cup u\} \). The subgraph of \( G - SI(G) \) induced by \( \phi(X') \) must be isomorphic to \( \overline{U}(G - SI(G)) \times X \) the vertex \( u \) is adjacent only to \( \phi(X' \cap L_{cc}) \), which are in \( T_{cc} \). We can either return \( \phi(X') \) as a small CAW, or call Thm. 2.1 with it.

Assume now that \( X \) has two neighbors \( x_1 \) and \( x_2 \) in \( X \cap L_{cc} \), and we have two distinct vertices \( y_1, y_2 \in T \) such that \( \phi(x_1) \not\sim y_1, \phi(x_2) \not\sim y_2 \) in \( E_{cc} \). By assumption, \( \phi(x_1) \sim y_2 \) and \( \phi(x_2) \sim y_1 \) in \( G \) (otherwise we have already been in the previous case). Note that \( y_1 \) and \( y_2 \) are nonadjacent; otherwise, \( (y_1, y_2) \) and the counterparts of \( \{x_1, x_2\} \) in \( R \) induce a hole of \( \overline{U}(G) \), which is impossible (we have assumed that it is chordal). We proceed as follows (Fig. 14).
Henceforth we may assume without loss of generality that $y$ is connected as well. It is worth noting that either 0 or 1.

Step 1 considers the case where $y_1 \sim h_1(b(x_1)) + 1$. By construction (Lem. 4.3) and noting that $b(x_1)y_1 \in L_{cc}$, it holds that $y_1 \not\sim h_1(b(x_1))$. Thus, $(y_1b(x_1))h_1y_1$ is a hole. A symmetric argument applies when $y_2 \sim h_1(b(x_2)) + 1$. Now that the conditions of step 1 do not hold true, step 2 is clear from assumption. Henceforth we may assume without loss of generality that $last(b(x_1)) > last(b(x_2)) \geq 0$. According to Lem. 5.1, the only possibility to make this true is $last(b(x_1)) = 1$ and $last(b(x_2)) = 0$. Consequently, $last(y_1) = |H| - 1$ (Lem. 4.3). Steps 3 and 4 are clear from the assumptions above.

The runtime is clearly $O(|G||H|)$ time. The proof is now complete.  

Lems. 5.8, 5.10 and 5.11 together conclude Thm. 2.3.

5.3 The olive-ring decomposition

If we have not found a subgraph in $\mathcal{F}_{LI}$, then the graph $U(G - SI(G))$ must be an interval graph. We build a clique path decomposition for it, and then employ it to build a clique hole decomposition for $G - SI(G)$ as follows. For notational convenience, we show the following variation, from which Lemma 5.4 follows. Recall that $G$ is connected; since no inner vertex of a shortest path can be simplicial, $G - SI(G)$ must be connected as well. It is worth noting that $G - SI(G)$ is unnecessarily prime, and the following proof does not use any property of prime graphs.

**Lemma 5.12.** Let $G'$ be a connected non-chordal graph. Given a clique path decomposition for $U(G')$, we can in $O(|G'||H|)$ time construct a hole decomposition for $G'$.

**Proof.** Let $\mathcal{P}$ be the clique path decomposition for $U(G')$. By definition of clique decomposition, if $G'$ remains connected after the deletion of vertices in a bag $K$, then $K$ must be an end of $\mathcal{P}$. By construction, $N[w]$, i.e., $\{w\} \cup L_{cc}$, is a maximal clique of $U(G')$. We argue first that $U(G') - N[w]$ is connected by showing that every neighbor $x$ of $\{w\} \cup L_{cc}$ is connected to $h_0^w$ in $U(G') - N[w]$. It holds vacuously when $x \in L \setminus L_{cc}$. If $x \in T$, then it is connected to $h_0^w$ via $h_1^w$. Thus $N[w]$ is an end bag of $\mathcal{P}$; without loss of generality, let it be $K_0$. On the other hand, $\{v^* : v \in T_{cc}\}$ is a minimal separator for $L \cup T \cup \{w\}$ and $R \setminus \{v^* : v \in T_{cc}\}$. Thus, a bag contains $h_0^w$ only if it appears to the right of this separator. Let $\ell$ be the largest index such that $h_0^w \not\subseteq K_\ell$. We build the hole decomposition $\mathcal{C}$ as follows: (1) take the sub-path $\mathcal{P}' = \{K_1 \cdots K_\ell\}$ of $\mathcal{P}$, (2) replace every vertex $x \in K_i$ with $1 \leq i \leq \ell$ by $\phi(x)$, and (3) add an edge to connect bags $K_i$ and $K_{\ell}$.

The rest of the proof is to show that $\mathcal{C}$ is a clique decomposition for $G'$. We verify first that the bags of $\mathcal{C}$ are precisely the set of maximal cliques of $G'$, that is, (1) every maximal clique of $G'$ appears exactly once in $\mathcal{C}$, and (2) every bag of $\mathcal{C}$ is a maximal clique of $G'$. For any $1 \leq i \leq \ell$, the bag $K_i$ does not contain $w$,
and thus there exists some maximal clique $X$ of $\bar{\varnothing}(G')$ such that $\phi(X) = K_i$. By Prop. 5.7, $K_i$ is a clique of $G'$. Therefore, for condition (2), it suffices to show that every bag of $\mathcal{C}$ is maximal. Hence we show

**Claim 1.** Let $K$ be a maximal clique of $G'$. It holds that (1) $K$ appears exactly once in $\mathcal{C}$, and (2) for any set $X$ of vertices of $\bar{\varnothing}(G')$ with $\phi(X) \subset K$, it does not induce a maximal clique of $\bar{\varnothing}(G')$.

**Proof.** We consider the intersection between $K$ and $T$, which has three cases. In the first case, $K$ is disjoint from $T$. Then $K$ induces a clique of $\bar{\varnothing}(G')$ as well, and both conditions hold trivially. In the second case, $K \subseteq T$. The maximality of $K$ implies $h_0 \in K$. Then $\{v^i : v \in K\}$ and $\{v^r : v \in K\}$ induce two disjoint cliques in $\bar{\varnothing}(G')$, which are subsets of $L$ and $R$, respectively. The bag in $L$ appears in $\mathcal{C}$, while the bag in $K$ contains $h_0$, and thus does not appear in $\mathcal{C}$. Therefore, $K$ appears only once in $\mathcal{C}$, and (1) is satisfied. For (2), if $X$ intersects both $L$ and $R$, then it does not induce a clique; otherwise it is a proper subset of $\{v^i : v \in K\}$ or $\{v^r : v \in K\}$, and thus induces a clique of $\bar{\varnothing}(G')$ which is not maximal.

In the remaining case we assume that $K$ intersect both $T$ and $R$. The set $K'$ of vertices of $\bar{\varnothing}(G')$ derived from $K$ is $(K \setminus T) \cup \{v^i, v^r : v \in K \cap T\}$. By construction of $\bar{\varnothing}(G')$ (and noting Lem. 5.2), all edges between $K \cap T$ and $K \setminus T$ must be either in $E_{cc}$ or $E_{cc}$. Consider first that these edges are in $E_{cc}$, then $K \cap T \subseteq E_{cc}$ and $K \setminus T$ is nonadjacent to $L$ in $\bar{\varnothing}(G')$. The only maximal clique of $\bar{\varnothing}(G')$ contained in $K'$ is $X = (K \setminus T) \cup \{v^r : v \in K \cap T\}$, and $\phi(X) = K$. Thus, both (1) and (2) are satisfied. A symmetric argument applies when all edges between $K \cap T$ and $K \setminus T$ are in $E_{cc}$.

It remains to verify that the hole $\mathcal{C}$ satisfies the other condition of the definition of clique decomposition, i.e., for every $v \in G'$, the bags containing $v$ induces a sub-path of $\mathcal{C}$. Recall that every vertex of $\bar{\varnothing}(G')$ appears in consecutive bags of $P$. For $v \in T$, all bags containing $v$ remain in $P'$, and are consecutive. For $v \not\in T \setminus T_{cc}$, a bag of $\mathcal{C}$ contains $v$ if and only if the corresponding bag in $P$ contains $v^i$ and $h_0$, which are consecutive. Assume now that $v \in T_{cc}$. On the one hand, $T_{cc} \subseteq K_1$; hence, $v \in K_1$, and the bags of $P'$ obtained from bags of $P$ containing $v^i$ appear consecutively in the left end of $P'$. On the other hand, if $P'$ contains bags of $P$ containing $v^r$, then $v \in K_1$, and they appear in the right end of $P'$. After the edge is added between $K_1$ and $K_2$, these bags are connected into a sub-path in $\mathcal{C}$. Observing $|H| \geq 6$, the decomposition $\mathcal{C}$ is clearly a hole, which concludes the proof.

Let $\mathcal{C}$ be the hole decomposition of $G - SI(G)$. For any pair of adjacent vertices, bags containing them are consecutive in $\mathcal{C}$, and by Lem. 5.2, they induce a proper sub-path of $\mathcal{C}$. Note that bags containing $h_0$ and $K_1$ are not subsets of each other. We have thus a left-right direction for $\mathcal{C}$: bags containing $K_1$ is to the left (resp., right) of $K_0$. With this numbering, a circular-arc model can be derived by Prop. 3.6. One may want to verify that arcs for vertices of $L$ appear clockwise in this model.

There cannot be three or less arcs covering the entire circle; otherwise, Lem. 5.2 must have been applied. Therefore, the circular-arc model is normal and Helly, and an inclusive-wise minimal set of arcs cover the whole circle if and only if it defines a hole. This observation allows us to decide the length of the shortest holes. Recall that every hole of $G$ is contained in $G - SI(G)$, whereupon we work on the hole decomposition $\mathcal{C}$ constructed in Lem. 2.4. For each vertex $v$, we can easily calculate $\text{left}(v)$ and $\text{right}(v)$. For each bag $K$ in the main cycle, we can also in linear time find the vertices $v_1$ and $v_2$ that achieves the minimum value for $\text{left}(v_1)$ and the maximum value for $\text{right}(v_2)$, respectively.

**Lemma 5.13.** We can in $O(|G|)$ time find a shortest hole of $G$.

---

**Figure 16:** Finding a shortest hole in an olive-ring decomposition

**Proof.** Given any vertex $v$, we can find a shortest hole through $v$ as follows. Let $v_0 = v$; for each $i = 0, \ldots$, we take the vertex in $K_{\text{right}(v_i)}$ that reaches the rightest bag as the next vertex $v_{i+1}$; the process stops when

```plaintext
0 build a list $L$ of $|K_0|$ slots, each of which contains $(u, u)$ for a distinct vertex $u \in K_0$;
1 order $L$ such that $\text{left}(u)$ is nondecreasing; reached = 0;
2 for each $(u, v) \in L$ do
2.1 if $v \sim u$ and $v$ is to the left of $u$ then return $u$;
2.2 if $v \not\in K_{\text{reached}}$ or $\text{right}(v) = \text{reached}$ then remove $(u, v)$ from $L$; goto 2;
2.3 reached = $\text{right}(v)$; $v$ = the vertex of $K_{\text{reached}}$ that maximizes $\text{right}(v)$;
3 goto 2.
```

\[\text{a shortest hole found.}\]

27
the first vertex \( v_1 \) that is adjacent to \( v_0 \) again, i.e., \( \text{left}(v_0) \in [\text{left}(v_1), \text{right}(v_1)] \). It returns the \((j + 1)\)-hole \((v_0 \ldots v_jv_0)\). The removal of all vertices in any bag from the graph will make it chordal, which means that every hole has to intersect every bag in \( \mathcal{C} \). Therefore, it suffices to find a vertex in bag \( K_0 \) that is in some shortest hole of \( G \). We use the procedure in Fig. 16, which uses the aforementioned idea to find a vertex in \( K_0 \) that appears in some shortest hole. A trivial implementation will take super-linear time, and thus the main focus will be an efficient implementation.

Each iteration of step 2 finds the next vertex for the shortest hole starting from a vertex in \( K_0 \). For the sake of simplicity, only the current vertex of this hole is recorded: let \((u, v)\) be a pair in \( U \) at the end of the ith step, then \( v \) is the ith vertex of the shortest hole through \( u \) found by previous procedure. Denote by \( P[u] \) the set of vertices that have been associated with \( u \) during the execution of this algorithm. Step 2.3 replaces \( v_i \) by the next vertex \( v_{i+1} \) for the starting vertex \( v_0 = u \). Therefore, if the condition of step 2.1 is satisfied, then \( P[u] \) will induce a shortest hole \( C \) via \( u \). Clearly, for any \( v \) that remains in \( U \), any shortest hole that passes \( u \) is no shorter than \( C \). Thus, it suffices to show that this also holds true for vertices that have been removed from \( U \) in step 2.2. Suppose there is a shortest hole \( C' \) through \( u' \). Then there is such a hole through all vertices in \( P[u'] \). Since the vertex of \( u \) preceding \( u' \) satisfies \( \text{left}(u) \leq \text{left}(u') \), replacing \( P[u'] \) by the first \( |P[u']| \) vertices of \( P[u] \) in \( C' \), we must obtain a cycle such that the arcs for its vertices cover the whole circle. Thus we obtain a hole through \( u \) that is no longer than \( C' \), i.e., strictly shorter than \( C \). This contradiction justifies the correctness of the algorithm.

Using a circular linked list for storing \( U \), the algorithm can be easily implemented in \( O(|G|) \) time: each bag is checked at most once.

If the hole found by Lem. 5.13 is short, i.e., having length 4 or 5, then we return it. Otherwise we add back \( SI(G) \) to build the olive-ring decomposition as follows. As promised, here we need to take into consideration the case where \( G \) is chordal. Recall it is prime and thus connected. We find \( SI(G) \) and check \( G – SI(G) \). If it is not an interval graph, we can either find and return a small CAW, or obtain a large CAW and call Thm. 2.1. If \( G – SI(G) \) is a clique, then we have a clique tree decomposition for \( G \) that is a star, which is a trivial caterpillar. Otherwise, the clique path decomposition for \( G – SI(G) \) has at least two bags, and we can extend it by appending two bags to its ends respectively. This gives a path of at least 4 bags, which can be treated as a hole decomposition by adding an edge connecting the end bags of the path. This allows us to handle the chordal case and non-chordal case, i.e., the construction of caterpillar decomposition and olive-ring decomposition, in a unified approach.

**Theorem 2.5 (restated).** Given a clique hole decomposition for \( G – SI(G) \), we can in \( O(|G|) \) time construct a clique decomposition for \( G \) that is an olive-ring.

**Proof.** We traverse all bags, and record \( \text{left}(v) \) and \( \text{right}(v) \) for each vertex \( v \in V(G) \setminus SI(G) \). We also record for every \( i = 0, \ldots, |\mathcal{C}| – 1 \), the size of \( K_i \cap K_{i+1} \). This can be done as follows: for each vertex \( v \), we add one to \( |K_i \cap K_{i+1}| \) (modulo \( |\mathcal{C}| \) when \( i < 0 \)) with \( i \in [\text{left}(v), \text{right}(v)] \).

For every simplicial vertex \( s \in SI(G) \), we check the vertices in \( N(s) \), and find \( i = \max_{v \in N(s)} \text{left}(v) \) and \( j = \min_{v \in N(s)} \text{right}(v) \). If \( i = j \) and \( K_i = N(s) \), then we add \( s \) into \( K_i \). If there is an 1 such that \( l \in [i, j – 1] \) and \( |K_l \cap K_{l+1}| = |N(s)| \), then we will insert \( N(s) \) as a new bag between \( K_l \) and \( K_{l+1} \). Otherwise we add \( N(s) \) as a bag pendant to \( K_i \).

We remark that the insertion of a new bag into the hole (the second case) will change the indices. Thus, for implementation, we may mark the position for each \( N[s] \) and add them after all the positions for them have been decided. This is justified because the insertion of a new bag in between does not change the \( K_i \cap K_{i+1} \).

It is easy to verify that the obtained decomposition is a clique decomposition. The first stage, construction of \( \text{left}(v) \) and \( \text{right}(v) \) for all vertices, can be done in one run, which takes \( O(|G|) \) time. With these indices, we can in \( O(|G|) \) time calculate all the sizes \( |K_i \cap K_{i+1}| \). The detection of the position for a simplicial vertex \( s \) takes \( O(d(s)) \) time, and hence \( O(|G|) \) time in total. In summary, the algorithm takes \( O(|G|) \) time.

In particular, in the olive-ring decomposition obtained as above for a chordal graph \( G \), there must be two consecutively bags that are disjoint. By detaching them, we get a caterpillar decomposition for \( G \).

**Corollary 5.14.** Let \( G \) be a prime graph. Given a clique path decomposition for \( G – SI(G) \), we can in \( O(|G|) \) time construct a clique decomposition \( \mathcal{C} \) for \( G \) that is caterpillar.

As aforementioned, the existence of the olive-ring decomposition is only a necessary but not sufficient condition for a prime graph to be a locally interval graph. It might still contain some small CAW, but it cannot have short holes.

28
Corollary 5.15. The shortest hole in the olive-ring decomposition for G has length 6.

We use SP(G) to denote those vertices contained only in pendant bags. Clearly SP(G) ⊆ SI(G), i.e., every vertex is SP(G) is simplicial.

6 Linear-time algorithms for recognizing almost unit interval graphs

This section presents the algorithms for recognizing unit-interval+kv, unit-interval+ke, and unit-interval−ke. We remark that the filling of small CAWs and the disposal of holes by deletion will be also used in the algorithm for almost interval graphs, and thus Prop. 6.2, Prop. 6.6, and Lem. 6.7 and presented in a general setting.

6.1 The completion and recognition of unit-interval−ke

It is now well known that holes can be easily filled in. For completion problems, the existence of a hole of more than k + 3 vertices will immediately imply “NO.” A hole of bounded length has only a bounded number of minimal ways to fill [50, 14], of which an interval supergraph must contain one.

Lemma 6.1. A minimal set of edges that fills a hole H has size |H| − 3, and the number of such sets is upper bounded by 4^{|H|−3}. Moreover, they can be enumerated in \(O(4^{|H|−3} \cdot |G|)\) time.

In a small CAW, the number of edges is 6, 9, 12, 6, or 10, and the number of edges in its complement is 9, 6, 9, 15, or 11, respectively. A trivial branching scheme for adding edges to fix a small CAW will thus fork into 15 directions. We observe that not all of them are needed, and 6 directions will suffice.

Proposition 6.2. For each small CAW in a graph G, there is a set of at most 6 edges such that any interval supergraph \(\hat{G}\) of G contains at least one of them.

![Figure 17: An interval supergraph must contain some dashed edge.](image)

Proof. These edges are depicted as dashed in Fig. [17]. We now argue their correctness by contradiction: We show that if \(\hat{G}\) does not contain any of the dashed edges, then \(\hat{G}\) contains a non-interval subgraph. Let W be the vertex set of the small CAW. We consider the edge(s) of \(\hat{G}\)[W] that are not in G[W], which must contain one of edges not appearing in Fig. [17] (solid or dashed), called omitted edges.

For net and sun, the only three omitted edges are among three terminals. If \(\hat{G}\) contains any of them, then there is 4-hole, e.g., \((v_1v_2t_2t_1)\) or \((v_1v_2v_2v_1t_1)\) if \(t_1t_2\) is an edge in \(\hat{G}\).

For rising sun, the omitted edges include the three among terminals, and \(t_2v_1, t_3v_0\). Similar as above, \(\hat{G}\) cannot contain any edge connecting two terminals. Suppose \(\hat{G}\) contains \(t_2v_1\), then \(\hat{G}[\hat{G}\setminus \{v_0\}]\) is a sun, and it reduces to the case we discussed above. A symmetric argument applies to \(t_3v_0\).

For long claw, there are nine omitted edges, three of which are among terminals, and each of the other six is between a terminal and a non-terminal vertex at distance 3 to it. Suppose first that \(\hat{G}\) contains \(t_1v_2\). Since neither \(v_1v_2\) nor \(t_1c\) is in \(\hat{G}\), there is a 4-hole \((t_1v_1v_2c_1)\). By symmetry, there is a 4-hole if \(\hat{G}\) contains any of \((t_1v_3, t_2v_4, t_2v_3, t_3v_1, t_3v_2)\). If none of these six edges is in \(\hat{G}\), then it must contain an edge connecting two terminals, and there is a 5-hole.

For whipping top, the omitted edges include \(\{t_1v_2, t_1, v_3, t_2v_3, t_3, v_2\}\) and three among terminals. Both \(t_1u\) and \(t_2c\) are dashed, they are not in \(\hat{G}\). Thus, \(\hat{G}\) contains \(t_1t_2\) as an edge; otherwise there is a 4-hole \((t_1c_1v_2t_1)\). By symmetry, \(t_1t_3\) is not an edge of \(\hat{G}\). For the same reason, \(\hat{G}\) cannot contain \(t_2v_3\) or \(t_3v_2\). Then it does not contain \(t_2t_3\) either; otherwise there is a 5-hole \((t_2v_2v_3t_1t_2)\). Since \(t_1c\) is dashed and not an edge, \(\hat{G}\) cannot contain both \(t_1v_2\) and \(t_1v_3\). If \(\hat{G}\) contains \(t_1v_2\), then \(\hat{G}[\hat{G}\setminus \{t_1\}]\) is a sun, and it reduces to the case we discussed above. A symmetric argument applies to \(t_1v_3\). \(\square\)
We now have all the details for the algorithm for unit interval completion, i.e., the recognition of unit-interval-ke. Recall that we can in linear time find a subgraph in $F_{UI}$ by calling Hell and Huang’s algorithm [47]. In particular, it returns a claw, net, or sun if the graph is chordal. Since a unit interval supergraph of $G$ is an interval supergraph of $G$, Lem. [6.1] and Prop. [6.2] apply as well, which are already sufficient to give the following algorithm.

**Corollary 6.3.** There is an $O(6^k \cdot ||G||)$-time algorithm for recognizing unit-interval–ke.

The constant $6$ of Cor. [6.3] is decided by Prop. [6.2], which is not tight for this problem and can be improved as sketched here. A claw has only three missing edges. According to Prop. [6.2], we only need to consider three edges for a sun. In Fig. [17], although there are six dashed edges in a net, the addition of every of them (they are symmetric) will introduce a claw. Then we need to add more edges. This induces a chain of edge additions, and it can be shown to be equivalent to a 4-way branching. Therefore, we have the following improved algorithm.

**Theorem 6.4.** There is an $O(4^k \cdot ||G||)$-time algorithm for recognizing unit-interval–ke.

### 6.2 Holes in a normal Helly circular-arc graph

In any circular-arc model, the arcs corresponding to vertices of a hole have to cover the entire circle (i.e., every point on the circle is contained in at least one arc of them). The converse holds true as well if the model is normal and Helly: a minimal set of arcs that cover the circle has size at least four. We will need the following observation from Lin et al. [60]:

**Proposition 6.5.** Let $G$ be a normal Helly circular-arc graph. If $G$ is not chordal, then every circular-arc model of $G$ is normal and Helly.

The following observation is immediate from the definition of clique decomposition. In particular, when the graph is already chordal, the only minimal hole cover is empty, which can be chosen as the two end bags.

**Proposition 6.6.** Let $\mathcal{C}$ be the clique hole decomposition of a normal Helly circular-arc graph $G$. For any minimal hole cover $V_-$ of $G$, there is a bag $K_t$ such that $V_- = K_t \cap K_{t+1}$.

Observing that there are at most $|G|$ bags, a minimum hole cover can be found in linear time. Therefore, Prop. [6.6] implies Prop. [2.6]. We now characterize minimal edge hole covers: the following is the technical version of Lem. [2.7] and implies it.

**Lemma 6.7.** Let $\mathcal{C}$ be the clique hole decomposition of a normal Helly circular-arc graph $G$. For any minimal edge hole cover $E_-$ of $G$, there is a bag $K_t$ and a partition $(X, Y)$ of $K_t$ such that $E_-$ is

$$E_{X,Y} = X \times Y \cup \{vu : v \in X, \text{left}(u) \in [l, \text{right}(v)] \} \cup \{vu : v \in Y, \text{right}(u) \in [\text{left}(v), r]\}. \quad (1)$$

Moreover, the deletion of $E_-$ will not introduce new claws to the graph.

**Proof.** We show first the following statement.

**Claim 2.** For any bag $K$ of $\mathcal{C}$ and any partition $(X, Y)$ of $K$, the set $E_{X,Y}$ gives an edge hole cover of $G$.

**Proof.** Let $G = G - E_{X,Y}$; we show that $G$ is an interval graph by building an interval model for it. Without loss of generality, we may renumber the bags such that the bag $K$ is $K_0$, then $E_{X,Y}$ can be written as

$$X \times Y \cup \{vu : v \in X, \text{left}(u) \leq \text{right}(v)\} \cup \{vu : v \in Y, \text{right}(v) \leq \text{right}(u)\}. \quad (2)$$

We start from the circular-arc model $\mathcal{A}$ for $G$ given by Prop. [3.6]. This model is normal and Helly, and $0 \in A_v$ if and only if $v \in K_0$. Let $c$ denote the length of the interval in the model. We use the following intervals:

$$I_v = \begin{cases} [0, \text{rp}(v)] & \text{if } v \in Y, \\ [\text{lp}(v), \text{rp}(v)] & \text{otherwise } (v \notin K_0). \end{cases} \quad (3)$$

We now verify this set of intervals represents $G = G - E_{X,Y}$. Let $u, v$ be a pair of vertices of $V(G)$. Consider first that $u, v \notin K_0$, then $I_u = A_u$ and $I_v = A_v$. Arcs $A_u$ and $A_v$ intersect if and only if $uv$ is an edge of $G$; (as $uv \notin E_{X,Y}$) intervals $I_u$ and $I_v$ intersect if and only if $uv$ is an edge of $G$. Consider second the case
u,v \in X$, then $uv \in E(G)$ and $c \in I_u \cap I_v$; likewise, if $u,v \in Y$, then $uv \in E(G)$ and $0 \in I_u \cap I_v$. Assume now that $u \in X$ and $v \in Y$, then $uv \notin E(X,Y)$ and is not an edge of $G$. On the other hand, since the model $M$ is normal, $1p(u) > r_p(v)$, which implies $I_u \cap I_v = \emptyset$. A symmetric argument applies when $u \in Y$ and $v \in X$.

In the remaining cases, exactly one of $u,v$ is in $K_0$; without loss of generality, assume $u \in X$ and $v \in V(G) \setminus K_0$, then we must have $0 < 1p(v)$. If $vu \notin E(G)$, i.e., $r_p(u) < 1p(v) < r_p(v) < 1p(u)$, then $I_u$ and $I_v$ do not intersect. If $1p(v) < r_p(u)$, then $vu \notin E(X,Y)$, and since the model is normal, $r_p(v) < 1p(u)$, the intervals $I_0$ and $I_0$ do not intersect. Likewise, if $1p(u) < r_p(v)$, then $r_p(u) < 1p(v)$ and $vu \notin E(X,Y)$; as a result, $I_v$ and $I_u$ intersect. A symmetric argument applies when $u \in Y$ and $v \in V(G) \setminus K_0$.

For the first assertion of this lemma, it suffices to show that any minimal hole cover $E_-$ contains some set of edges given by $\{1\}$, and then the equality follows from the minimality of $E_-$. Suppose, for contradiction, that for every bag $K$ of $G$ and any partition $(X,Y)$ of $K$, the set $E_-$ does not contain $E_{X,Y}$. We find a hole of $G - E_-$ as follows. Let $H$ be a shortest hole of $G$. We take two nonadjacent vertices $h_1$ and $h_2$ from $H$. We consider the set of vertices whose arcs lie totally between $[1p(h_1), r_p(h_2)]$. Let $G_1$ the subgraph of $G - E_-$ induced by them. By the assumption above, we can find an induced $(h_1, h_2)$-path in $G_1$. The arcs of inner vertices of this path covers the arc $[r_p(h_1), 1p(h_2)]$. Likewise, we can find a subgraph $G_2$ of $G - E_-$ induced by vertices whose arcs lie totally between $[1p(h_2), r_p(h_1)]$. We find an induced $(h_2, h_1)$-path in $G_2$, and arcs of inner vertices of this path covers the arc $[r_p(h_2), 1p(h_1)]$. Combining these two paths gives a cycle of $G - E_-$. If the two neighbors of $h_1$ (resp., $h_2$) are adjacent, then we remove $h_1$ (resp., $h_2$). This gives an induced cycle of $G - E_-$; since arcs corresponding to these sets of vertices still cover the circle, its size is at least 4. Therefore, it is a hole of $G - E_-$, contradicting that $E_-$ is an edge hole cover.

Suppose, for contradiction of the second assertion, there is a new claw. Then there must be three pairwise adjacent vertices $u, v, x$ in $G$ such that precisely one edge among them is in $E_-$. Since $E_- \neq \emptyset$, the graph $G$ cannot be chordal. According to Prop. 6.5 the model given by Prop. 3.6 for $G$ is normal and Helly. However, the arcs for $u, v, x$ contradict the Helly property. This concludes this proof. $\square$

We remark that both statements can be adapted for a graph with an olive-ring decomposition. Such a graph can be viewed as adding simplicial vertices to a normal Helly circular-arc graph. In particular, Prop. 6.6 remains true, while Lem. 6.7 needs more cases: If a simplicial vertex $v$ is adjacent to both parts $X$ and $Y$, then we need to choose one side for it and add the set of edges between it and the other part into $E_-$.  

### 6.3 The recognition of unit-interval $+kv$ and unit-interval $+ke$

We now turn to the deletion problems. Our algorithms first break all small subgraphs of $G$ in $\mathcal{F}_{UI}$. Here we consider claws, nets, suns, and short holes as small. Their detection relies on the application of Thm. 1.5 to the quotient graph $G_Q$ of $G$. Observe that every long claw, whipping top, or rising sun contains a claw.

**Lemma 6.8.** We can in $O(||G||)$ time either detect a claw, net, sun, or short hole in $G$, or assert its nonexistence.

**Proof.** No generality will be lost by assuming that $G$ is connected, on which we apply the algorithm described in Fig. 18. Steps 0 and 1 are straightforward, and after that, the graph $G$ must be non-chordal. We start by finding all the maximal strong modules and constructing the quotient graph $G_Q$ of $G$. The algorithm enters step 2 when $G_Q$ is a clique. Step 2.1 is clear, and if it is not the case, then there must be some maximal strong module $M$ that induces a non-chordal graph. Step 2.2 finds a hole $H$ from $G[M]$. If $|H| < 6$, then step 2.3 returns it. Otherwise, $H$ contains three pairwise nonadjacent vertices, which, together with any vertex $v \in N(M)$, make a claw (step 2.4).

Now the algorithm has passed step 2, $G_Q$ must be prime; it enters step 3 when some maximal strong module $M$ of $G$ does not induce a clique. Step 3.2 is clear; otherwise, we find a claw as follows. Let $u$ be a neighbor of $v$ in $G_Q$. By definition, $u$ has a neighbor $x \notin N(v)$; otherwise $u$ and $x$ make a module of $G_Q$, contradicting that $G_Q$ is prime. We pick any pair of vertices from the modules corresponding to $u$ and $x$, respectively; they, together with two nonadjacent vertices in $M$, give a claw.

Hereafter every maximal strong module $M$ of $G$ induces a clique and $G_Q$ is not chordal.

Step 4 applies the algorithm for Thm. 1.5 to $G_Q$, which has two possible outcomes. The algorithm enters step 5 if the output is a short hole or small CAW. Note that every long claw, whipping top, or rising sun contains a claw, which can be easily identified and returned. Now that $K$ is an olive-ring decomposition of the quotient graph $G_Q$, the algorithm enters step 6 or 7 based on whether $K$ is a hole or not. It should be noted that all vertices and subgraphs in the rest of the algorithm are meant in the quotient graph $G_Q$; the details on how to translate a subgraph of $G_Q$ back to an isomorphic subgraph of $G$ is omitted.
Step 6 takes care of the case \( \text{SP}(G_Q) \neq \emptyset \). It starts from a vertex \( v \in \text{SP}(G_Q) \), and other vertices will be selected accordingly; their existence is guaranteed by the definition of clique decomposition. The bag \( N[v] \) has a unique neighbor \( K \) in \( \mathcal{K} \), and let \( K_L \) and \( K_R \) be the neighboring bags of \( K \) in the cycle of \( \mathcal{K} \). If some neighbor \( u \) of \( v \) appears in both \( K_L \) and \( K_R \), then the algorithm enters step 6.2, which returns a claw. Step 6.3 uses a vertex \( u \) that is in \( K \) but not \( K_L \) or \( K_R \). As the condition of step 6.2 is not satisfied, (1) \( v \) is nonadjacent to \( x, y, x', y' \); (2) \( u \) is adjacent to \( x, y \) but not \( x', y' \); and (3) \( x' \) and \( y' \) are distinct and nonadjacent. As a result, either \( \{u, x, x', y'\} \) is a claw (if \( x = y \)) or \( \{v, u, x, x', y'\} \) is a net (otherwise). In the remaining cases, a vertex in \( N[v] \) is either in \( K_L \) or \( K_R \) but not both. Since \( N[v] \) is disjoint from \( K_L \cap K_R \), it holds that \( x \neq y' \) and \( x' \neq y \). The claw or sun of steps 6.7 or 6.8 is clear.

If the algorithm reaches step 7, the decomposition \( \mathcal{K} \) is a hole of which every maximal strong module induces a clique of \( G \). Since the smallest number of arcs that cover the circle is six, the graph contains no net, sun, or short hole. A claw exists if and only if there is a pair of vertices \( v, u \) such that \( \text{left}(v) < \text{left}(u) \leq \text{right}(u) < \text{right}(v) \). The existence of such a pair can be tested easily; this completes the algorithm. It is clear that every step can be implemented in linear time.

A claw, net, sun, or short hole has no more than six vertices, and thus every of them can be broken by branching on at most six vertex deletions. Likewise, a claw, net, or short hole can be broken by branching on at most six edge deletions. However, a sun contains nine edges. We observe that six of them are used in the definition of this sun, and one of them has to be deleted; otherwise we have to delete one of the other three edges, but then the resulting graph contains a hole. Therefore, a sun can be broken by branching on six edge deletions. We are now ready to present the algorithms for \textsc{unit interval vertex deletion} and \textsc{unit interval edge deletion}.

**Theorem 6.9.** There are \( O(6^k \cdot ||G||) \)-time algorithms for recognizing unit-interval + \( kv \) and unit-interval + \( ke \).

**Proof.** The algorithms for both problems have a similar plan, which consists of two phases. Phase 1 calls Lem. 6.8 to find a claw, net, sun, or short hole of \( G \). If such a subgraph is found, then it branches into at most 6 sub-instances, each of which has one vertex or edge deleted. After all of these subgraphs have been
destroyed, the remaining graph must be a normal Helly circular-arc graph; this finishes phase 1. If the graph is already chordal, then it must be a unit interval graph, and the algorithm is concluded. Otherwise, let C be the hole decomposition, and the algorithm proceeds to phase 2 as follows. Let $k_1$ be the number of vertices or edges that have been deleted in phase 1. For vertex deletion, it takes $K_1 \cap K_{i+1}$ with the minimum size among all $0 \leq i \leq |C| - 1$. For edge deletion, it tries every bag $K$ of $C$ with $|K| \leq k - k_1 + 1$ and every partition of $K$; it deletes the edge set $E_-$ with the minimum size among them. This concludes the algorithm.

The correctness of phase 1 is straightforward. The correctness of phase 2 follows from Prop. 2.6 and Lem. 6.7 respectively. In particular, the edge set $E_-$ decided by every nontrivial partition of a bag $K$ has size at least $|K| - 1$. Since phase 2 is allowed to delete at most $k - k_1$ edges, it suffices to consider bags of size no more than $k - k_1 + 1$.

We now analyze their runtime. In each branching step, we have at most 6 directions, of which each sub-instance has one less vertex or edge, the total number of sub-instances is thus bounded by $6^{k_1}$. Therefore, at the end of phase 1, there are at most $6^{k_1}$ sub-instances. Since there are at most $|G|$ bags in $C$, for vertex deletion, phase 2 takes $O(|G|)$ time for each sub-instance. Therefore, the algorithm for UNIT INTERVAL VERTEX DELETION runs in time $O(6^k \cdot |G|)$. For edge deletion, a bag of size no more than $k - k_1 + 1$ has at most $2^{k_1+1}$ partitions. The smallest edge set $E_-$ can thus be found in $O(2^{k_1+1} \cdot |G|)$ time. Noting that $O(6^{k_1} \cdot 2^{k_1+1}) = O(6^k)$, the algorithm for UNIT INTERVAL EDGE DELETION runs in time $O(6^k \cdot |G|)$ as well.

## 7 Modules

This section is devoted to the proof of Thms. 1.6, 1.8 and Thms. 2.8, 2.10. As the properties on vertex deletions (Thms. 1.6 and 2.8) are straightforward, main efforts will be on the edge versions. Simple examples tell us that not all maximum spanning interval subgraphs and minimum interval supergraphs preserve all modules. For example, consider the graph in Fig. [19] which is obtained from a as follows: the center c is replaced by a clique of 5 vertices, and the shallow terminal is replaced by two nonadjacent vertices $s_1$ and $s_2$. As shown in the dashed edges, a minimum completion can be adding for either $s_1$ and $s_2$ an edge to connect it to some base vertex. In this minimum interval supergraph, however, $\{s_1, s_2\}$ is no longer a module. The module $\{s_1, s_2\}$ does not induce a connected subgraph, and we may alternatively connect both $s_1$ and $s_2$ to the same base vertex so that we obtain another minimum interval supergraph that preserves $\{s_1, s_2\}$ as a module. These observations turn out to be general, i.e., a module M of a graph $G$ that is not a module of a minimum interval supergraph $\hat{G}$ of $G$ has to be disconnected, and we can always modify $\hat{G}$ to another minimum interval supergraph $\hat{G}'$ of $G$ such that $M$ is a module of $\hat{G}'$.

![Module {s_1, s_2} is not preserved by a minimum interval supergraph (dashed edges are added).](image)

For the edge deletion, however, we have to break some modules. The simplest example is a 4-hole graph, which has two nontrivial modules, but its maximum spanning interval subgraph must be a simple path, which is prime. Even the connectedness does not help, e.g., consider the graph in Fig. 20a which is obtained by completely connecting two 4-paths ($v_1v_2v_3v_4$) and ($u_1u_2u_3u_4$). A maximum spanning interval subgraph has to be prime, and is the same as shown in Fig. 20a after dashed edges deleted (up to isomorphism). Both examples contain some 4-hole, which inspire us to consider 4-hole-free graphs. We show that any 4-hole-free graph has a maximum spanning interval subgraph that preserves all its modules. It is worth stressing that not all maximum spanning interval subgraphs of a 4-hole-free graph preserve all its modules, e.g., the graph and its maximum spanning interval subgraph in Fig. 20b.

The way we prove these statements will be explicitly constructing an interval graph satisfying the claimed conditions by giving an interval model. All interval models in this section are normalized. Let $P$ be a set of points that are in $[\ell_1, \ell_2]$. By projecting $P$ from $[\ell_1, \ell_2]$ to $[\ell_3, \ell_4]$ we mean the following operation:

$$p \rightarrow \frac{\ell_4 - \ell_3}{\ell_2 - \ell_1}(p - \ell_1) + \ell_3 \quad \text{for each } p \in P.$$
Before the proof of Thm. 2.9, we give a stronger result on Lemma 7.2.

That is, each point in \([\ell_1, \ell_2]\) is proportionally shifted to a point in \([\ell_3, \ell_4]\). It is easy to verify that all new points are in \([\ell_3, \ell_4]\) and this operation retains relations between every pair of points. In particular, if we project the endpoints of all intervals for \(V(G)\), the set of new intervals defines the same interval graph.

### 7.1 Modules in maximum induced interval subgraphs

The preservation of modules in maximum induced interval subgraphs is an immediate consequence of Prop. 3.2. Therefore, for Thms. 1.6 and 2.8 it suffices to prove their second assertions, which follow from the following statement. Recall that if a 4-hole contains precisely two vertices from some module \(M\), then neither \(M\) nor \(N(M)\) can induce a clique.

**Lemma 7.1.** Let \(G - V_-\) be a maximum induced interval subgraph of a graph \(G\). Let \(M\) be a module of \(G\) such that at least one of \(M\) and \(N(M)\) induces a clique. If \(M \not\subseteq V_-\), then replacing \(G[M \setminus V_-]\) by any maximum induced interval subgraph of \(G[M]\) in \(G - V_-\) gives a maximum induced interval subgraph of \(G\).

**Proof.** Suppose, for contradiction, that the new graph \(G - V'_-\) is not an interval graph. From \(G - V'_-\), we can find a subgraph \(X\) of \(F_1\), which must intersect both \(M\) and \(V(G) \setminus M\). Since at least one of \(M \setminus V'_-\) and \(N(M) \setminus V'_-\) induces a clique, \(X\) contains exactly one vertex of \(M\); let it be \(x\). By assumption that \(M \not\subseteq V_-\), there exists a vertex \(x' \in M \setminus V_-\) (possibly \(x' = x\)). However, replacing \(x\) by \(x'\) in \(X\), we obtain a subgraph of \(G - V_-\) in \(F_1\), which is impossible. \(\square\)

We point out that Thms. 1.6 and 2.8 holding for all maximum induced interval subgraphs, are stronger than their counterparts for edge versions.

### 7.2 Modules in maximum spanning interval subgraphs

Before the proof of Thm. 2.9, we give a stronger result on clique modules (i.e., modules inducing cliques) of any graph with respect to its maximum spanning interval subgraphs.

**Lemma 7.2.** A clique module \(M\) of a graph \(G\) is also a clique module of any maximum spanning interval subgraph \(G'\) of \(G\).

**Proof.** Let \(C\) be the component of \(G[M]\) such that \(N_{G}(C)\) attains the maximum value among all components of \(G[M]\). We modify any given normalized interval model \(J = \{1_v : v \in V(G)\}\) for \(G\) as follows. Let \(p = 1p(C)\) and \(q = 1p(C)\). For each \(v \in M\), we set \(1p'(v)\) to a distinct value in \((p - \epsilon, p)\), and set \(1p'(v)\) to a distinct value in \((q, q + \epsilon)\). For each \(u \in V(G) \setminus M\), we set \(1u' = 1u\). Let \(G'\) be the interval graph defined by \(J'\). By construction, a vertex \(u \in V(G) \setminus M\) is adjacent to \(C\) in \(G'\) if and only if it is in \(N_{G}(C)\). Since \(N_{G}(C) \subseteq N_{G}(M)\), we have \(G' \subseteq G\). For each \(v \in M\), it holds that

\[
|N_{G}(v) \setminus M| \leq |N_{G}(C) \setminus M| = |N_{G}(C)| = |N_{G}(v) \setminus M|.
\]

On the other hand, \(M\) induces a clique in \(G'\). They together imply \(|G| \leq |G'|\), while the equality is only attained when \(G[M]\) is a clique, hence \(C = M\), and \(M\) is completely connected to \(N_{G}(C) = N_{G}(M)\). Therefore, \(G' = G\), and this verifies the claim. \(\square\)
Theorem 2.9 (restated). Let $G$ be a graph of which every 4-hole is contained in some maximal strong module. There exists a maximum spanning interval subgraph $G'$ of $G$ such that every maximal strong module $M$ of $G$ is a module of $G'$, and replacing $G[M]$ by any maximum spanning interval subgraph of $G[M]$ in $G$ gives a maximum spanning interval subgraph of $G$.

Proof. As a consequence of Lem. 7.2, it suffices to consider non-clique modules. It is hence assumed that $G[M]$ is not a clique, and then $N_G(M)$ must induce a clique of $G$. Let $\mathcal{J} = \{I_v : v \in V(G)\}$ be a normalized interval model for $G$. We first argue that

Claim 3. For any component $C$ of $G[M]$, the set $N_{G[C]}$ induces a clique of $G$.

Proof. Suppose the contrary, we construct an interval graph $G'$ with $\subseteq G' \subseteq G$ as follows. Let $x, y$ be a pair of vertices in $N_{G[C]}$ such that $I_x \cap I_y = \emptyset$, i.e., $x \not\sim y$ in $G$. Without loss of generality, assume $rp(x) < 1p(y)$. We take an arbitrary point $\ell$ with $rp(x) < \ell < 1p(y)$, and for every $u \in N_{G[C]}$, extend the interval $I_u$ to include $\ell$: if $I_u$ is to the left of $\ell$, we set $rp'(u)$ to be a distinct point in $(\ell, \ell + \epsilon)$; if $I_u$ is to the right of $\ell$, we set $1p'(u)$ to be a distinct point in $(\ell - \epsilon, \ell)$. We use the graph defined by the set of new intervals as $G'$. To see $\subseteq G'$, note that all intervals are extended only; to see $G' \neq G'$, note that $xy$ is an edge in $G'$ but not in $G$. It remains to verify $G' \subseteq G$, for which it suffices to consider the edges in $E(G') \setminus E(G)$. Such an edge always connects two vertices of $N_{G[C]}$, which is a subset of $N_{G[M]}$, and hence exists also in $E(G)$. Therefore, $G'$ is an interval subgraph of $G$ with strictly more edges than $G$, which is impossible.

Let $G_M$ be any maximum spanning interval subgraph of $G[M]$. We modify $G$ first to make $M$ satisfy the claimed condition. Let $C$ be the component of $G[M]$ such that $|N_G(C)|$ attains the maximum value among all components of $G[M]$. We have seen that $N_G(C)$ induces a clique in $G$. The intersection of all intervals $\{I_v : v \in N_{G[C]}\}$ is thus nonempty; let it be $[p, q]$. Since $C$ is connected, $\cup_{v \in C} I_v = [1p(C), rp(C)]$. The interval $[1p(C), rp(C)]$ intersects $[p, q]$, and we can choose a common point $\ell$ in them. We construct another graph $G'$ by projecting an interval model for $G_M$ into $[\ell - \epsilon, \ell + \epsilon]$. Denote by $G'$ the graph represented by the set of new intervals.

We now verify $G' \subseteq G$ and $\|G'\| \geq \|G\|$. On the one hand, $V(G) \setminus M$ induces the same subgraph in $G'$ and $G$. On the other hand, by assumption, $G[M] = G_M$ is an interval subgraph of $G[M]$ and has no more edges than $G[M]$. Therefore, it suffices to consider edges between $M$ and $V(G) \setminus M$. For $G'$, there are edges are $M \times N_{G[C]}$. Since $N_{G[C]}$ is a subset of $N_{G[M]}$, it holds that $G' \subseteq G$. By selection of $C$ (i.e., $N_{G[C]}$ has the largest size), $\|G'\| \geq \|G\|$.

We have now constructed a maximum spanning interval subgraph of $G$ where $M$ satisfies the claimed conditions. Only intervals for vertices in $M$ are changed, and thus this operation can be successively applied on the maximal strong modules of $G$ one by one. If a module already satisfies the conditions, then it remains true after modifying other modules. Therefore, repeating this process will derive a claimed maximum spanning interval subgraph of $G$.

As explained below, this settles Thm. 1.7 as well.

Lemma 7.3. Thms. 1.7 and 2.9 are equivalent.

Proof. We show first that Thm. 1.7 implies Thm. 2.9. For each maximal strong module $M$, we replace $G[M]$ by a maximum spanning interval subgraph of $G[M]$; let $G'$ denote the obtained graph. Clearly, $G'$ is a subgraph of $G$ and has the same quotient graph as $G$. Since $G_Q$ is 4-hole-free, every non-simplicial vertex of $G_Q$ corresponds to a clique of $G'$, and every maximal strong module induces an interval graph, $G'$ is 4-hole-free. Let $G$ be a maximum spanning interval subgraph of $G'$ as specified by Thm. 1.7. By definition, it is a spanning interval subgraph of $G$ as well, and we now argue that it is minimum. Since $G'[M]$ is an interval subgraph for every $M$, Thm. 1.7 implies that $G'[M] = G[M]$, and it is a clique if and only if $G[M]$ is a clique. Therefore, every edge of $E(G) \setminus E(G')$ is among different maximal strong modules. Any maximum spanning interval subgraph of $G$ has to delete at least this amount of edges among different maximal strong modules. Therefore, $G$ a minimum maximum spanning interval subgraph of $G$ satisfying conditions of Thm. 2.9.

We now verify the other direction. We first use inductive reasoning along the modular decomposition tree to show that it holds for every strong module of $G$. The base case is trivial: the root of the modular decomposition tree has the only trivial module $V(G)$. Since every strong module $M$ induces a 4-hole-free subgraph $G[M]$, its quotient graph trivially satisfies the condition of Thm. 2.9. Thus, conditions of Thm. 1.7 hold for every maximal strong module of the subgraph $G[M]$. This settles all strong modules, and then we consider modules that are not strong. Such a module $M$ is composed of more than one strong modules, and they are either pairwise adjacent or pairwise nonadjacent. In the first case, (noting that graph contains no
7.3 Modules in minimum interval supergraphs

Before the proof of Thm. 2.10, we give a stronger result on connected modules, i.e., modules inducing connected subgraphs, of a graph with respect to its minimum interval supergraphs.

**Theorem 7.4.** Let $\hat{G}$ be a minimum interval supergraph of $G$. Every connected module $M$ of graph $G$ is a module of $\hat{G}$, and if $G[M]$ is not a clique, then replacing $G[M]$ by any minimum interval supergraph of $G[M]$ in $\hat{G}$ gives a minimum interval supergraph of $G$.

**Proof.** The statement holds vacuously if $M$ consists of a single vertex or a component; hence we may assume $|M| > 1$ and $N_G(M) \neq \emptyset$. Let $I = \{v : v \in V(G)\}$ be a normalized interval model for $\hat{G}$. We define $p = \min_{v \in M} rp(v)$ and $q = \max_{v \in M} lp(v)$. Let $x$ and $y$ be the vertices that attain the bound $p$ and $q$ respectively, i.e., $rp(x) = p$ and $lp(y) = q$; possibly $x = y$, which is irrelevant in the following argument. By assumption, $N_G(M) = N_G(M) \subseteq N_{\hat{G}}(M) \subseteq N_{\hat{G}}(M)$, and the first assertion is equivalent to $N_G(M) = N_{\hat{G}}(M)$. Suppose, for contradiction, that there exists $z \in N_G(M) \setminus N_{\hat{G}}(M)$, then we modify $I$ into another set of intervals $I' = \{I'_v : v \in V(G)\}$. We argue that the interval graph $\hat{G}'$ defined by $I'$ is a supergraph of $G$ and has strictly smaller size than $\hat{G}$. This contradicts the fact that $\hat{G}$ is a minimum interval supergraph of $G$, and thus the assertion must be true. Since $I$ is normalized, $p \neq q$.

**Case 1.** $p > q$. Then $M$ induces a clique of $\hat{G}$. We have $[q, p] \subseteq I_v$ for every $v \in M$, and $I_u \cap [q, p] \neq \emptyset$ for every $u \in \hat{N}_G(M)$. We construct $I'$ as follows. We keep $rp'(x) = p$ and $lp'(y) = q$; for $v \in M \setminus \{y\}$, we set $rp'(v)$ to a distinct value in $(q - \epsilon, q)$; and for $v \in M \setminus \{x\}$, we set $rp'(v)$ to a distinct value in $(p, p + \epsilon)$. For each $u \in V(G) \setminus M$, we set $I'_u = I_u$. In the graph $\hat{G}'$ represented by $I'$, the subgraph induced by $M$ is a clique; the subgraph induced by $V(G) \setminus M$ is the same as $\hat{G} - M$; and $M$ is completely connected to $N_G(M) \subseteq N_{\hat{G}}(M)$. This verifies $G \subseteq \hat{G}'$. On the other hand, for any $v \in M$, from $I'_v \subseteq I_v$, it can be inferred $N_G(v) \subseteq N_{\hat{G}}(v)$; it follows that $\hat{G}' \subseteq \hat{G}$. By assumption that $z \notin N_G(M)$, the interval $I'_z(= I_z)$ is either to the left of $q$ or to the right of $p$, and then from the choice of $\epsilon$ we can conclude that it is either to the left of $q - \epsilon$ or to the right of $p + \epsilon$. As a result, $z \not\in M \in \hat{G}'$ and thus $\hat{G}' \neq \hat{G}$; in other words, $\hat{G}'$ is a proper subgraph of $\hat{G}$. This contradiction verifies the first assertion for the case $p > q$.

**Case 2.** $p < q$. Then $x \neq y$ and $y \not\in \hat{G}$. We have $I_v \cap [p, q] \neq \emptyset$ for every $v \in M$, and $[p, q] \subseteq I_u$ for every $u \in \hat{N}_G(M)$. (See Figure 21) We construct $I'$ as follows. For each point $\ell \in [p, q]$, it thickness is defined by $\theta_\ell = |\{v \in V(G) \setminus M : \ell \in I_v\}|$. Let $\ell$ be a point in $[p, q]$ that attains the minimum thickness, denoted by $\ell^*$; without loss of generality, we may assume $\ell$ is different from any endpoint of intervals in $\ell$. For each $v \in M$, we set $I'_v$ by projecting $I_v$ from $[p, q]$ to $[\ell - \epsilon, \ell + \epsilon]$. For each $v \in V(G) \setminus M$, we set $I'_v = I_v$. Let $\hat{G}'$ be represented by $I'$. The subgraphs induced by $M$ and $V(G) \setminus M$ are the same as $G[M]$ and $\hat{G} - M$ respectively. Hence, we only need to consider edges between $M$ and $V(G) \setminus M$. For each $u \in N_G(M)$, the interval $I'_u(= I_u)$ contains $[p, q]$, which contains $[\ell - \epsilon, \ell + \epsilon]$ in turn. Therefore, $N_G(M) \subseteq N_{\hat{G}}(M)$, and it follows that $G \subseteq \hat{G}'$.

It remains to verify $|E(\hat{G}')| < |E(\hat{G})|$, which is equivalent to

$$|E(\hat{G}') \cap (M \times V(G) \setminus M)| < |E(\hat{G}) \cap (M \times V(G) \setminus M)|. \quad (3)$$
By the selection of $\ell$ and $\epsilon$, no interval in $\mathcal{I}$ has an endpoint in $[\ell - \epsilon, \ell + \epsilon]$. Thus, for each $u \in V(G) \setminus M$, the interval $I_u' (= I_u)$ contains $\ell$ if and only if $[\ell - \epsilon, \ell + \epsilon] \subseteq I_u'$. As a result, $N_G(M) = N_{\hat{G}}(M) = \{v \in V(G) \setminus M : \ell \in I_v\}$, and the left-hand side of (3) is equal to $|M| \cdot \theta_c$. We now consider the right-hand side of (3), i.e., the number of edges between $M$ and $V(G) \setminus M$ in $\hat{G}$. For every $v \in M$, the interval $I_v$ contains some point $x \in [p, q]$, which means $v$ is adjacent to all vertices in $\{u \in V(G) \setminus M : x \in I_u\}$. By the selection of $\ell$, it holds that $|N_{\hat{G}}(v) \setminus M| \geq |N_G(v) \setminus M| = \theta_c$. For the correctness of (3), it suffices to show that this is strict for at least one vertex in $M$.

As $z \not\in \hat{N}_{\hat{G}}(M) \cup M$, the interval $I_z' (= I_z)$ does not contain $[p, q]$ (see the thick/red edges in Figure 21). If $p < \min \{z, q\} < q$ (see $z_1$ in Figure 21), then $\theta_{zp(z)} > \theta_{zp(z)} \geq \theta_c$. As $G[M]$ is connected, there exists a vertex $v \in M$ such that $x_{zp(z)} \in I_v$, which we are done. A symmetric argument applies if $p < 1 \min \{z, q\} < q$. Hence we may assume there exists no vertex $u \in V(G) \setminus M$ such that $I_u' (= I_u)$ has an endpoint in $[p, q]$. Suppose now $x_{zp(z)} < p$ (see $z_2$ in Figure 21), and let $v \in \hat{N}_{\hat{G}}(v) \cap M$. By the selection of $p$, it holds that $p \in I_v$, and $v$ is adjacent $x_{zp(z)} \in I_p$. Therefore, $|N_{\hat{G}}(v) \setminus M| \geq 1 + \theta_p > \theta_c$. A symmetric argument applies if $1 \min \{z, q\} > q$. This verifies (3) and finishes the proof of case 2 and the first assertion.

For second assertion, we may assume that $\hat{G}[M]$ is not a clique. Then $N_G(M)$ must be a clique, and thus we can take the nonempty intersection of all intervals for $N_G(M)$; let it be $[p, q]$. Then replacing intervals $\{I_v : v \in M\}$ by an interval model of any minimum interval supergraph of $G[M]$ projected to $[p, q]$ will makes another interval model. Since $N_G(M) \subseteq N_{\hat{G}}(M)$, which remains common neighbors of $M$, the graph defined by this new model is clearly an interval supergraph of $G$. It is easy to verify that its size is no larger than $G$. This completes the proof. 

Thm. 7.4 will ensure preservation of any connected module in perpetuity. Observing that a graph $\hat{G}$ is a minimum interval supergraph of $G$ if and only if it is a minimum interval supergraph of any graph $G'$ satisfying $G \subseteq G' \subseteq \hat{G}$, it can be further strengthened to:

**Corollary 7.5.** Let $\hat{G}$ be a minimum interval supergraph of graph $G$. A connected module $M$ of any graph $G'$ satisfying $G \subseteq G' \subseteq \hat{G}$ is a module of $\hat{G}$.

We are now ready to prove Thm. 2.10 which is restated below.

**Theorem 2.10 (restated).** There is a minimum interval supergraph $\hat{G}$ of $G$ such that every maximal strong module $M$ of $G$ is a module of $\hat{G}$, and if $\hat{G}[M]$ is not a clique, then replacing $\hat{G}[M]$ by any minimum interval supergraph of any graph $G'$ satisfying $G \subseteq G' \subseteq \hat{G}$, it can be further strengthened to:

**Proof.** It suffices to consider maximal strong modules that are not connected in $G$. Let $M$ be such a module and let $C$ be a component of $G[M]$. By definition, $N_G(C) = N_G(M) = N_{\hat{G}}(M)$, and $C$ is a connected module of $G$. Thus, by Thm. 7.4 $C$ remains a module of $\hat{G}$. Let $\{I_v : v \in V(G)\}$ be a normalized interval model for $\hat{G}$.

Assume first that $G$ contains no edge between different components of $G[M]$. Let $x, y$ be the vertices in $N_{\hat{G}}(C)$ such that $r_p(x)$ and $l_p(y)$ are the smallest and largest, respectively. Suppose $x \not\in N_{\hat{G}}(C)$, then we can delete edges between $x$ and $C$ to obtain an interval supergraph of $\hat{G}$, which is strictly less edges than $\hat{G}$: an interval model for the new graph can be obtained by setting $l_p(v) = r_p(x) + \epsilon$ or $r_p(v) = l_p(y) - \epsilon$ for every $v \in C$; here we are using the fact that the model is normalized. A symmetrical argument applies to $y$, and thus both of $x$ and $y$ must be in $N_{\hat{G}}(C)$. We now argue that $N_{\hat{G}}(C)$ must induce a clique in $G$. Suppose, for contradiction, that $N_{\hat{G}}(C)$ does not induces a clique, then $x \not\sim y$ in $\hat{G}$. Since $N_{\hat{G}}(C) = N_G(M) = N_{\hat{G}}(M)$, for every $v \in M \setminus C$, the interval $I_v$ fully contain $[r_p(x), l_p(y)]$. However, $v$ is then adjacent $C$ in $\hat{G}$, which contradicts the assumption. Therefore, $N_{\hat{G}}(C)$ induces a clique of $\hat{G}$.

We construct another minimum interval supergraph $G'$ of $G$ satisfying the specified conditions as follows. Let $C_0$ be the component of of $G[M]$ such that $N_{\hat{G}}(C_0)$ has minimum size. We choose a point $\ell$ contained in all intervals for $N_{\hat{G}}(C_0)$, and project an interval model of any minimum interval supergraph of $G[M]$ to $[\ell - \epsilon, \ell + \epsilon]$. Let $\hat{G}'$ be the graph defined by this new model. By construction, $N_{\hat{G}'}(M) = N_{\hat{G}}(C_0)$, which fully contains $N_{\hat{G}}(C_0)$, and hence $G \subseteq \hat{G}'$. The selection of $C_0$ implies that $\|\hat{G}'\| \leq \|\hat{G}\|$.

Assume now that $E(\hat{G})$ contains edges between different components of $G[M]$. We may add these edges first, and then consider the resulted graph $G'$. According to Cor. 7.5 $\hat{G}$ is a minimum interval supergraph of $G'$, and this reduces to the previous case.
Therefore, if $\G[M]$ is not a clique, then it can be replaced by any minimum interval supergraph of $G[M]$. This operation only change intervals for vertices in $M$, it can be successively applied on the maximal strong modules of $G$ one by one. After that, the condition holds for every of them. □

With a similar argument as Lem. 7.3, we can derive the equivalence between Thm 1.8 and Thm 2.10.

### 8 Linear-time algorithms for recognizing almost interval graphs

Recall that we can easily reduce into smaller instance(s) when the quotient graph $G/Q$ is complete or has no edge. Hence we may assume that $G/Q$ is prime, to which Thm. 1.5 is applied. Observing that small CAWs and short holes are easily disposable, we may assume that the outcome of Thm. 1.5 is a clique decomposition. For the same reason, we can assume that every $4$-hole is contained in some maximal strong module, whereupon Thms. 2.8-2.10 apply. Moreover, for the completion problem, we can always call Lem. 6.1 to fill all holes and work on a chordal graph, while for the deletion problems, we can call prop. 6.6 and Lem. 6.7 after all CAWs have been destroyed. In accordance with these observations, we are mainly concerned with large CAWs of a prime graph.

Our algorithms find a large CAW satisfying a certain minimality condition, from which they identify a set of bounded number of vertices/edges that intersects some minimum solution. Hence they branch on deleting/adding one of these vertices/edges. The argument for the existence of such a solution will be constructive, i.e., given a solution that avoids all those vertices/edges, we construct another one intersecting it of the same size. In particular, we will use a similar reasoning as Section 7: we show the new graph is an interval graph by constructing an interval model for it, and the new model is modified from an interval model of the given interval graph.

Besides the operation “project” defined at the beginning of Section 7, we will need the following operation on circular-arc models. Given a pair of distinct positive numbers $p$, $q$ and a pair of nonadjacent vertices $u$, $v$, we can always reverse, rotate, and/or re-scale all arcs such that $1p(u) = p$ and $rp(v) = q$.

#### 8.1 Minimal frames

This subsection formally defines minimal frames, and more importantly, gives a linear-time algorithm to find a minimal frame. Let $\K$ be the olive-ring decomposition, and let $\ell$ be the unique cycle in $\K$. Recall that $SP(G)$ is the set of simplicial vertices of $G$ that does not appear in any bag of $\ell$. The following statement can be easily observed from Fig. 1 and the fact that the graph induced by $\bigcup_{K \in \ell} K = V(G) \setminus SP(G)$ is always a normal Helly circular-arc graph.

**Proposition 8.1.** Let $\K$ be an olive-ring decomposition of a prime graph $G$ obtained by Thm. 1.5. The graph is free of CAW if and only if $\K$ is a hole.

As a consequence, we may assume in this section that $SP(G)$ is nonempty. Recall that for a vertex $v \in V(G) \setminus SP(G)$, left($v$) and right($v$) are indices of the leftmost and rightmost bags in $\ell$ containing $v$. We extend this definition to vertex $s \in SP(G)$ as follows: left($s$) = $\max_{v \in N(s)}$ left($v$) and right($s$) = $\min_{v \in N(s)}$ right($v$). Therefore, for every vertex $v \in N(s)$, it holds that $[\text{left}(s), \text{right}(s)] \subseteq [\text{left}(v), \text{right}(v)]$. We have calculated left($v$) and right($v$) for every $v \in V(G) \setminus SP(G)$ during the construction of olive-ring decomposition (Thm. 2.5). They can be used to calculate left($s$) and right($s$) for every $s \in SP(G)$ in linear time. The inner vertices of a frame $(s : c_1, c_2 : l, l' ; r', r)$ is defined to be $U_1 = \bigcup_{r' < \ell < r} K_{\ell} \setminus N(s)$.

**Definition 4.** A frame $(s : c_1, c_2 : l, l' ; r', r)$ is called minimal if

1. **(C1)** no vertex in $SP(G)$ is adjacent to both $c_1$ and $c_2$;
2. **(C2)** $N[c_1] \cap N[c_2] \subseteq N[v]$ for every $v \in N(s)$;
3. **(C3)** $N[v] \subseteq N[c_1] \cap N[c_2]$ for every $v \in U_1$;
4. **(C4)** $s$ is the only vertex of the frame in $SP(G)$; and
5. **(C5)** $N[l'] \cap N[c_2] \subset N[c_1]$.

38
As a consequence of Lemma 5.11 and 5.13, in a minimal frame, every \( v \in U_1 \) is nonadjacent to \( SP(G) \). We now show how to find such a minimal frame. As we have said earlier, there might still exist small CAWs in a graph that admits an olive-ring decomposition. Therefore, our algorithm either returns a small CAW, or a minimal frame, either of which will serve our purpose. Without loss of generality, we may assume that in the olive-ring decomposition, every pendant bag \( N[s] \) for \( s \in SP(G) \) is attached to the bag \( K_{left(s)} \) in the cycle \( \mathcal{C} \). We can preprocess \( \mathcal{X} \) in linear time to make it satisfy this assumption.

**Lemma 8.2.** Given a prime graph \( G \) and an olive-ring decomposition \( \mathcal{X} \) for it, in \( O(|\mathcal{X}|) \) time we can find a small CAW, a \( + \) or \( - \) whose base consists of three vertices, or a minimal frame.

**Proof.** The algorithm is given in Fig. 22. We start from a simple claim.

**Claim 4.** Given a pair of nonadjacent vertices \( s \in SP(G) \) and \( v \in V(G) \setminus SP(G) \) such that \( [left(s), right(s)] \subseteq [left(v), right(v)] \), we can in \( O(|\mathcal{X}|) \) time find a small CAW, a \( + \) or \( - \) whose base consists of three vertices.

**Proof.** By definition, we can find \( c_1 \) and \( c_2 \) such that \( right(c_1) = left(s) \) and \( left(c_2) = right(s) \). If \( c_1 \neq c_2 \), then we can find a sun, rising sun, or \( + \) with \( |B| = 3 \). Otherwise \((c_1 = c_2)\), we can find a long claw, net, or \( - \) with \(|B| = 3\).

Recall that \( SP(G) \) is nonempty; the algorithm starts by arbitrarily picking a vertex from it. Step 1 searches for a new vertex \( s \) such that it is minimal in the sense that there is no other vertex \( s' \in SP(G) \) satisfying \( [left(s'), right(s')] \subseteq [left(s), right(s)] \). We need to argue that Claim 4 applies if there is another vertex with \( [left(s'), right(s')] = [left(s), right(s)] \) (step 1.3). Since the graph is prime, \( N(s') \neq N(s) \), and thus there must be some vertex \( v \) with \( [left(s'), right(s)] \subseteq [left(v), right(v)] \) and nonadjacent to \( s \) or \( s' \).

Therefore, we can call Claim 4 with either \((v, s')\) or \((s', v)\).

Now that such a shallow terminal \( s \) is identified, we find the rest of the minimal frame in steps 2-6. Steps 2 and 3 find center \( s, c_1, c_2 \) and \( l', r' \) respectively; their existence is clear. It is worth noting that \( c_1 \) and \( c_2 \) might refer to the same vertex, and this fact is irrelevant in the rest of the algorithm. Recall that \( K_{left(c_2)} \cap K_{left(c_1)} \) is not a subset of \( N(s) \); otherwise we should have insert \( N(s) \) as a bag in between them. Therefore, \( right(b_1) \geq left(c_2) \), which implies \( l' \sim c_2 \); likewise, \( c_1 \sim r' \). Since \( left(l') < left(s) \) and \( right(s') < right(r') \), if \( right(l') \geq right(s) \) or \( left(l') \leq left(s) \), then Claim 4 applies. Step 5 finds base terminals, which exist because \( K_{left(l')} \cap K_{left(r')} \neq \emptyset \) and \( K_{right(l')} \cap K_{right(r')} \neq \emptyset \).

To verify steps 7 and 8, we need to first check the pairwise adjacency relation. By construction, both \((l'c_2r'\sim r)\) and \((l'c_2r')\) are paths of length 4. Since the length of a shortest hole is at least 6 (Lemma 5.13), \( right(l) \leq left(c_2) - 1 \) implies that \( l \sim c_2, r' \). Likewise, we conclude \( r' \sim c_1, l' \). Therefore, if \( l' \sim r' \), then \((s, c_1, c_2, l', r', r')\) is a net (when \( c_1 = c_2 \)) or a rising sun (when \( c_1 \neq c_2 \)). This justifies step 7. Otherwise, \((l' \sim r')\) (\( s = c_1, c_2 : l', r', r' \) is a frame, witness by any \((l', r')\)-path with internal vertices in \( \bigcup_{r' < c < left(r')} K_l \) (noting that \( K_l \cap K_{c+1} \setminus N(s) \) is nonempty for every \( right(l') \leq l < left(r') \)). It remains to verify that the frame \((s = c_1, c_2 : l', r', r')\) is minimal.
Suppose to the contrary of C1 there is a vertex \( x \in \text{SP}(G) \) that is adjacency to both \( c_1 \) and \( c_2 \). By definition, \( \text{left}(x) \leq \text{left}(c_2) \leq \text{left}(s) \) and \( \text{right}(x) \geq \text{right}(c_1) \geq \text{right}(s) \). Neither inequality can be strict, as otherwise we have chosen \( x \) instead of \( s \) in step 1. However, in this case, step 1.3 should have called Claim 4. This contradiction verifies C1.

For C2 let \( x \) be any vertex in \( N[c_1] \cap N[c_2] \). As a result of C1 \( x \not\in \text{SP}(G) \), and thus is contained in \( \cup_{\text{left}(c_2) \leq v \leq \text{right}(c_1)} K_v \). By definition, \( \text{left}(v) \leq \text{left}(c_2) \) and \( \text{right}(v) \geq \text{right}(c_1) \), and thus \( v \sim x \). This verifies C2.

C3 and C4 are immediate from the selection of the center(s) \( c_1, c_2 \) and the terminal \( s \), respectively.

Suppose to the contrary of C5 there is a vertex \( x \in N[l'] \cap N[c_2] \setminus N[c_1] \). If \( x \in \text{SP}(G) \), then it should have been chosen in step 1. Otherwise, \( \text{right}(l') \geq \text{right}(s) \), and step 4 should have called Claim 4. Neither of them is possible, and thus C5 must hold.

The runtime is clearly \( O(|G|) \). This concludes this proof. \( \square \)

We point out that the frame of a CAW with minimum number of vertices is unnecessarily minimal. For example, in Fig. 23, the frame \( (s': c', c': b_1, b_2, b_8, r) \) of CAW \( (s': c', c': b_1, b_2 \cdots b_8, r) \) is minimal, though it has more vertices than \( (s: c, c: l, b_1 b_2 c' b_8, r) \).

Recall that a shortest hole has length six. A large CAW \( W \) with a minimal frame can see at most five vertices in a shortest hole. Therefore, we may number the main cycle \( C \) in a way that \( K_0 \) is disjoint from \( W \). As a result, we can assume \( \text{left}(l) \leq \text{left}(l') \leq \text{right}(l) < \text{left}(c_2) \leq \text{right}(l') < \text{left}(r') \leq \text{right}(c_1) < \text{left}(r) \leq \text{right}(r') \leq \text{right}(r) \). This will simplify our reference of indices of bags in \( C \).

\[ \text{Figure 23: } (s: c, c: l, b_1 b_2 c' b_8, r) \text{ has less vertices, but its frame is not minimal.} \]

### 8.2 The recognition of interval+kv

**Lemma 8.3.** Let \( F \) be a minimal frame \( (s: c_1, c_2 : l, l', r', r) \) in a prime graph \( G \) and let \( K \) be an olive-ring decomposition for \( G \). Let \( S_i = K_i \cap K_{i+1} \setminus N(s) \) for \( \text{right}(l') \leq i < \text{left}(r') \), and let \( l \) be an index such that \( S_l \) has the minimum size among all of them. If every maximum induced interval subgraph of \( G \) fully contains \( V(F) \), then there exists a maximum induced interval subgraph \( G - V_\sim \) such that \( V_\sim \cap U_1 = S_l \).

**Proof.** Let \( G - V^* \) be any maximum induced interval subgraph of \( G \); by assumption, \( V^* \) is disjoint from \( V(F) \). We use \( (V^* \setminus U_1) \cup S_1 \) as the claimed \( V_\sim \). To verify that \( G - V_\sim \) is an interval graph, we construct an interval model \( J \) for it. Let \( J^* = \{ I^*_v : v \in V(G) \setminus V^* \} \), where \( I^*_v = [1p^*(v), rp^*(v)] \), be a normalized interval model for the interval graph \( G - V^* \). Let \( A = \{ A_v : v \in V(G) \setminus \text{SP}(G) \} \), where \( A_v = \{ 1p^c(v), rp^c(v) \} \), be the circular-arc model for \( G - \text{SP}(G) \) obtained by using Prop. 3.6. We make the following assumptions on \( J^* \) and \( A \).

1. The interval for \( l' \) is to the left of that for \( r' \) in \( J^* \), i.e., \( 1p^*(l') < 1p^*(r') \). Since \( l' \neq r' \), this assumption is clear. As a consequence of this assumption, when \( c_1 \neq c_2 \), all endpoints of intervals for \( c_1 \) and \( c_2 \) are ordered as follows (see Fig. 4):

\[ 1p^*[c_1] < (rp^*[l] <) 1p^*[c_2] < rp^*[c_1] < (1p^*[r] <) rp^*[c_2]. \]

2. The models \( J^* \) and \( A \) are aligned such that \( [1p^*[c_2] = 1p^*[c_2] \) and \( [rp^*[c_1] = rp^*[c_1] \). We will use the same values for \( 1p^*[c_2] \) and \( rp^*[c_1] \) in \( J \). As a result, we can refer to \( 1p^*[c_2] \) and \( rp^*[c_1] \) without specifying which model.
(3) For every $v \in N(s) \setminus V^*$, the interval $I^*_v$ fully contains $[1p(c_2), rp(c_1)]$. By (C2), $N[c_1] \cap N[c_2] \subseteq N[v]$, and thus we can always extend $I^*_v$ to cover $[1p(c_2), rp(c_1)]$ without breaking the model.

(4) For every $v \in U_1 \setminus V^*$, the interval $I^*_v$ is fully contained in $[1p(c_2), rp(c_1)]$. By (C3), $N[v] \subseteq N[c_1] \cap N[c_2]$, and thus we can always shrink $I^*_v$ to satisfy this assumption.

Note that assumptions (3) and (4) do not conflict with each other.

Let $p$ be a point in the circular-arc model such that $p \in A_v$ if and only if $v \in K_{l} \cap K_{l+1}$. The existence of $p$ is clear from the construction of $A$, e.g., $l + 1/2$ in the original model given by Prop. 3.6. The centers $c_1, c_2$ belong to both $K_l$ and $K_{l+1}$, and thus $p \in [1p(c_2), rp(c_1)]$. We are now ready to present interval model $f = (I_v : v \in V(G) \setminus V^*)$ for $G - V^*$ as follows. If $v \notin V^*$ and $I^*_v$ is disjoint from or fully contains $[1p(c_2), rp(c_1)]$ then $I_v = I^*_v$; this includes both $c_1$ and $c_2$. The interval $I_v = [p - e, p + e]$. For every vertex $v \in U_1 \setminus V^*$, we use $I_v = A_v$. If a vertex does not belong to all of above, i.e., $I^*_v$ intersects only part of $[1p(c_2), rp(c_1)]$, then we set

$$1p(v) = \begin{cases} 1p^*(v) & \text{if } 1p^*(v) < 1p(c_2), \\ 1p(c_2) & \text{otherwise}, \end{cases} \quad \text{and } rp(v) = \begin{cases} rp^*(v) & \text{if } rp^*(v) > rp(c_1), \\ rp(c_1) & \text{otherwise}. \end{cases}$$

Since both the given interval model and circular-arc model represent only part of $V(G)$, we need to verify that $I_v$ is well-defined for every $v \in V(G) \setminus V^*$. That is, the construction does not use any nonexistent interval $I^*_v$ for $v \in V^* \setminus V_*$ or nonexistent arc $A_v$ with $v \in SP(G) \setminus V_*$. A vertex $v \in V^* \setminus V_*$ is in $U_1$, and thus $I_v = A_v$. By construction, $SP(G) \setminus V_* = SP(G) \setminus V^*$. For any $v \in SP(G) \setminus V_*$ that is different from $s$, the interval $I_v$ is disjoint from $[1p(c_2), rp(c_1)]$ (C4), and hence $I_v = I^*_v$.

**Claim 5.** The interval model $f$ represents $G - V^*$.

**Proof.** Note that $V^* \setminus V_* \subseteq U_1$; for any vertex $v \in V^* \setminus V_*$, the interval $I_v (= A_v)$ is contained in $[1p(c_2), rp(c_1)]$. Therefore, by construction, $I_v$ contains a point $p \notin [1p(c_2), rp(c_1)]$ if and only if $I^*_v$ contains it. Given a pair of intersecting intervals $I_u$ and $I_v$, we show $u \sim v$. Let $p$ be a point in $I_u \cap I_v$. If $p \notin [1p(c_2), rp(c_1)]$, then $I^*_v$ intersects $I^*_u$; otherwise, $A_u$ intersects $A_v$. In either case, we have $u \sim v$.

Given a pair of adjacent vertices $u, v \in V(G) \setminus V_*$, we show $I_u \cap I_v \neq \emptyset$. Assume first that neither of them is in $U_1$, then intervals $I^*_u$ and $I^*_v$ are defined and intersect. If $I^*_u \cap I^*_v$ contains a point $p \notin [1p(c_2), rp(c_1)]$, then $p$ is also contained in both $I_u$ and $I_v$. Otherwise, both $I^*_u$ and $I^*_v$ intersects $[1p(c_2), rp(c_1)]$; by (C1), $u \notin SP(G)$. Thus, $A_u$ and $A_v$ are defined and intersect at some point $p \in [1p(c_2), rp(c_1)]$, which is contained in both $I_u$ and $I_v$. Assume now that, without loss of generality, $u \in U_1$. Then $I_u = A_u \subseteq [1p(c_2), rp(c_1)]$. By (C1) and (C3), $v \notin SP(G)$, and thus $I_u \cap I_v = A_u \cap A_v$.

We have seen that $G - V_*$ is another interval subgraph, and it remains to show that $|V_*| \leq |V_*^*|$. By construction, this is equivalent to that $V^* \cap U_1$ contains at least $|S|$ vertices. Suppose this is not true, then by the fact that $|S|$ is minimum, for every $v$ with $\text{right}(l') \leq l < \text{left}(r')$, at least a vertex in $S$ remains in $G - V^*$. Therefore, there exists an $(l', r')$-path with all inner vertices from $U_1$, which is impossible as it together with $F$, makes a CAW of $G - V^*$. Therefore, it must hold that $|V_*| \leq |V_*^*|$, and then $G - V_*$ is a maximum induced interval subgraph of $G$.

It is worth noting that Lem. 3.3 is stronger than a similar result presented in our previous work [17]. The improvement is not only on the constant (decreased from 10 to 8), but more importantly, the new proof does not require the graph $G$ to be chordal. This relaxation permits us to attend to large CAWs in a non-chordal graph. We are now ready to present the algorithm for the INTERVAL VERTEX DELETION problem in Fig. 24.

Recall that we are after a maximum induced interval subgraph of the input graph $G$; and for any maximal strong module $M$, we either delete it, or find a maximum induced interval subgraph of $G[M]$.

**Theorem 8.4.** Algorithm interval-vertex-deletion solves INTERVAL VERTEX DELETION in $O(8^k \cdot ||G||)$ time.

**Proof.** Let us verify first its correctness. Step 0 gives two trivial exit conditions. Steps 1 and 2 take care of the special case where the quotient graph $G_Q$ is already an interval graph. It might be a graph with no edge, a clique, or a prime interval graph. The algorithm enters step 1 if $G_Q$ is a clique. Step 1.1 is straightforward, and if it is not the case, then all vertices in $V(G) \setminus M$ are universal, and it suffices to solve $(G[M], k)$; this module $M$ exists because $G$ is not an interval graph. Otherwise, $G_Q$ is an interval graph but not a clique, and the algorithm enters step 2. Step 2.1 is clear, and the correctness of 2.2 can be argued by inductive reasoning and Thm. 2.8. In particular, if at least $k + 1$ vertices need to be deleted from $G[M]$ to make it an interval graph, then $(G, k)$ must be a “NO” instance. Otherwise, by inductive hypothesis, we may assume...
Lemma 8.6. Let \((\mathcal{G}, \mathcal{H})\) be a minimal frame in a prime graph \(G\) such that the shortest \((l', r')\)-path in \(G[U \cup \{l', r'\}]\) has length more than \(6k + 6\). Let \(\ell \in \{\text{right}(l'), \text{left}(r')\}\) and \((X, Y)\) be a partition.
of $K_t \setminus N(s)$ such that $E_{t,X,Y}$ has the minimum size among all bags and partitions. If the maximum spanning interval subgraphs of $G$ contains at least $|G| - k$ edges and every of them fully contains $\{sc_1, sc_2, l′c_1, r′c_2\}$ as well as every edge $b_1b_{i+1}$ with $i \in [0, \ldots, k + 1] \cup \{d - k - 2, \ldots, d\}$, then there is a maximum spanning interval subgraph $\mathcal{G}$ of $G$ that is disjoint from $E_{t,X,Y}$.

**Proof.** We use $P_k$ and $P_{k+1}$ to denote the paths $(b_0 \cdots b_{k+1})$. Let $\mathcal{G}^*$ be a maximum spanning interval subgraph of $G$. By assumption, $|E(G) \setminus E(\mathcal{G}^*)| \leq k$; and $\mathcal{G}^*$ fully contains $\{sc_1, sc_2, l′c_1, r′c_2\}$ and every edge of paths $P_k$ and $P_{k+1}$. We build another maximum spanning interval subgraph of $G$ satisfying the claimed condition by modifying $\mathcal{G}^*$. Let $J^* = \{I^*_v \in V(G) : |I^*_v| ≥ k\}$, where $I^*_v = \{l^*_p(v), r^*_p(v)\}$, be a normalized interval model for $\mathcal{G}^*$. Let $A = \{A_v : v \in V(G) \setminus E(\mathcal{G}^*)\}$, where $A_v = \{l^*_p(v), r^*_p(v)\}$, be the circular-arc model for $G - E(\mathcal{G}^*)$ obtained by using Prop. 3.6. We make following assumptions on $J^*$ and $A$.

1. The interval for $l'$ is to the left of that for $r'$ in $J^*$, i.e., $r^*_p(l') < 1p^*(r')$. Since $l' \neq r'$, this assumption is clear. Moreover, as $l \in N(l') \setminus N(c_2)$, it follows that $1p^*(l') < 1p^*(l) < 1p^*(c_2)$.

2. If $c_1 \neq c_2$ then $1p^*(c_1) < 1p^*(c_2)$. Suppose the contrary, then we can always extend $I^*_c$ to the left by setting $1p^*(c_1) = 1p^*(c_2) - e$ to satisfy this assumption. The safeness of this extension can be argued using (C2): since $\{l^*_p(c_2), r^*_p(c_1)\}$ is a subset of $I^*_c$, and by the selection of $e$, any interval $I^*_v$ that intersects $\{l^*_p(c_2) - e, 1p^*(c_1)\}$ only if $v \sim c_1$. An alternative interpretation of this assumption is $1p^*(c_1) ≤ 1p^*(c_2)$.

3. For every $v \in N(s)$, the interval $I^*_v$ either fully contains $I^*_c \cap I^*_s$, or is disjoint from it. By (C2), $N[c_1] \cap N[c_2] \subseteq N[v]$, and thus we can always extend $I^*_c$ to cover $I^*_c \cap I^*_s$ without breaking the model.

4. For every $v \in U_1$, the interval $I^*_v$ is either fully contained in, or disjoint from $\{1p(c_2), r^*_p(c_1)\}$. By (C3), $N[v] \subseteq N[c_1] \cap N[c_2]$, and thus we can always shrink $I^*_c$ to satisfy this assumption.

5. For every vertex $v$ nonadjacent to $U_1$, its interval is disjoint from $\{r^*_p(l'), 1p^*(r')\}$.

We give here two properties of $\mathcal{G}^*$.

**Claim 6.** The interval for every vertex in $N_G(s)$ fully contains $\{1p(c_2), r^*_p(c_1)\}$.

**Proof.** Suppose, for contradiction, there is a vertex $v \in N(s)$ such that $I^*_v$ does not fully contains $I^*_c \cap I^*_s$. Then by assumption (3), $I^*_v$ is disjoint from $I^*_c \cap I^*_s$. For any vertex $x \in U_1$, if $I^*_x$ intersects $I^*_v$, then by assumption (4), it is disjoint from $I^*_x \cap I^*_c$. In other words, $x$ cannot be adjacent to both $c_1$ and $c_2$ in $\mathcal{G}^*$. Since $x$ is adjacent to all of $c_1, c_2$, and $v$ in $G$, at least one of $xc_1, xc_2, and xv$ needs to be deleted. Therefore, $|E(G) \setminus E(\mathcal{G}^*)| \geq |U_1| > k$. This contradiction proves this claim.

As a result, $N_G(s) = N_{\mathcal{G}^*}(s)$, which we denote by $N(s)$.

**Claim 7.** There exist $p$ and $q$ with $1 \leq p \leq 3k + 3$ and $d - 3k - 3 \leq q < d$ such that $\mathcal{G}^*$ retains all edges incident to $N(b_p) \setminus N(s)$ and $N(b_q) \setminus N(s)$.

**Proof.** By symmetry, it suffices to verify the existence of $p$. By selection of $b$, for each $i$ with $1 \leq i \leq 3k + 3$, the sets $N(b_i) \setminus N(s)$ and $N(b_{i + 3}) \setminus N(s)$ are nonadjacent; otherwise, there is a shorter base than $b$, which is impossible. In particular, an edge cannot be incident to both $N(b_i) \setminus N(s)$ and $N(b_{i + 3}) \setminus N(s)$. Suppose that $E(G) \setminus E(\mathcal{G}^*)$ contains one edge incident to $N(b_i) \setminus N(s)$ for each $i$ with $1 \leq i \leq 3k + 3$, then its size must be at least $(3k + 3)/3 > k$, a contradiction.
A direct result of Claim 7 is that intervals for both \( b_p \) and \( b_q \) are fully contained in \( I_v^* \cap I_v^* \). By assumption (1), and noting that paths \( P_l \) and \( P_r \) remain in \( G^* \), the interval for \( b_q \) must be to the left of that for \( b_p \), i.e., \( r_p^*(b_p) < l_p^*(b_q) \). See Fig. 25. Let \( U_i^* \) be the set of vertices \( \bigcup_{1 \leq i \leq t} (l_i \cup r_i) \). 

**Claim 8.** Let \( X \) be the subset of vertices in \( V(G) \setminus (U_i^* \cup N(s)) \) whose intervals intersect \([r_p^*(b_p), l_p^*(b_q)]\). For each \( v \in X \), the interval \( I_v^* \) is fully contained in \([r_p^*(b_p), l_p^*(b_q)]\), and \( N_{G^*}(X) = N(s) \).

**Proof.** Since \( v \) is adjacent to neither \( b_p \) nor \( b_q \), the first assertion is straightforward. By Claim 7, \( N(s) \subseteq N_{G^*}(v) \) for every vertex \( v \in X \), and thus it suffices to show that \( N_{G^*}(v) \subseteq N(s) \). Suppose, for the contradiction of the second assertion, that there is a pair of vertices \( v \in X \) and \( u \in U_i^* \) such that \( v \sim u \) in \( G^* \). Such a vertex \( u \) must be adjacent to \( b_p \) or \( b_q \). However, \( u \) is then adjacent to neither \( b_{3k+6} \) nor \( b_{d-3k-6} \). This is impossible.

In particular, \( s \) belong to the set specified in Claim 8. Let \( c \) be the middle point between \( r_p^*(b_p) \) and \( l_p^*(b_q) \), i.e., \( c = (r_p^*(b_p) + l_p^*(b_q))/2 \). We present now the new interval model \( J = \{ I_v : v \in V(G) \} \). Consider first a vertex \( v \notin U_i^* \), which is either in \( N(s) \) or nonadjacent to \( b_p \) or \( b_q \). If \( I_v^* \) intersects but not fully contains \([r_p^*(b_p), l_p^*(b_q)]\), then \( I_v^* \) must be fully contained in \([r_p^*(b_p), l_p^*(b_q)]\) and we set \( I_v^* \) by projecting \( I_v^* \) from \([r_p^*(b_p), l_p^*(b_q)]\) from \([e - c, e + c]\). Otherwise, \( I_v^* \) either fully contains or is disjoint from \([r_p^*(b_p), l_p^*(b_q)]\), then we set \( I_v^* \) by the same rules. Note that \( s \) and \( N(s) \) are in the first and second categories respectively.

For a vertex \( v \in U_i^* \), the endpoints of \( I_v^* \) are given as follows. Note that since \( U_i^* \subseteq U_i \subseteq V(G) \setminus SP(G) \), the arc \( A_v \) is defined. If \( l_p^*(v) \) (resp., \( r_p^*(v) \)) is not in \([r_p^*(b_p), l_p^*(b_q)]\), then \( l_p(v) = r(p_v) \) (resp., \( r_p(v) = r^p(v) \)). Otherwise, \( l_p(v) \) (resp., \( r_p(v) \)) is obtained by projecting \( l_p^*(v) \) (resp., \( r_p^*(v) \)) from \([r_p^*(b_p), l_p^*(b_q)]\) to \([r_p^*(b_p), c - e]\) if \( r_p^*(v) < c \) or \( v \in X \), \([c + e, l_p^*(b_q)]\) if \( l_p(v) > e \) or \( v \in Y \).

By construction, \( r_p(b_p) = r_p^*(b_p) \) and \( l_p(b_q) = l_p^*(b_q) \). Let \( G \) be the interval graph represented by the new interval model \( J \).

**Claim 9.** \( G \subseteq G^* \cap (E(G) \cap U_i^* \times U_i^*) \setminus E_{L,X,Y}, \) which is a subgraph of \( G \).

**Proof.** We consider first edges incident to \( V(G) \setminus U_i^* \). Let \( v \in V(G) \setminus U_i^* \). If \( I_v^* \) intersects \([r_p^*(b_p), l_p^*(b_q)]\), then by Claim 8 and the definition of projecting operation, \( v \) has the same neighbors in \( G^* \) and \( G \). Otherwise, \( I_v^* \) either fully contains or is disjoint from \([r_p^*(b_p), l_p^*(b_q)]\), and in both cases \( v \) has the same neighbors in \( G^* \) and \( G \).

On the adjacency of \( U_i^* \times U_i^* \), it is easy to verify that \( G \) is disjoint from \( E_{L,X,Y} \). For another pair of vertices, the definition of projecting operation, they are adjacent in \( G \) if and only if they are adjacent in \( G \).

It remains to show that \( G \) is maximum, i.e., the size of \( G \) is no less than \( |G^*| \). Recall that \( I_v^* \) must lie between \([r_p^*(b_p), l_p^*(b_q)]\). Therefore, \( b_{3k+3} \) and \( b_{d-3k-3} \) are disconnected in the subgraph \( G^* \setminus N(s) \). In other words, there must be some \( E_{L,X,Y} \) that are not in \( E(G) \), whose size is no less than \( E_{L,X,Y} \) by assumption. On the other hand, all different edges of \( G \) and \( G^* \) are incident to \( U_i^* \). In summary, we have \( |G| \geq |G^*| \), and this concludes this proof.

We are now ready to present the algorithm for the INTERVAL EDGE DELETION problem in Fig. 26. Recall that we are after a maximum spanning interval subgraph for a "YES" instance. The proof of Thm. 8.7 is similar as that for Thm. 8.4 and thus omitted.

**Theorem 8.7.** Algorithm interval-edge-deletion solves INTERVAL EDGE DELETION in \( k^{O(k)} \cdot |G| \) time.

### 8.4 The recognition of interval—ke

The frame of a \( \dag \) (resp., \( \ddag \)) has 6 vertices and 5 edges (resp., 7 vertices and 11 edges). Therefore, a frame \( F \) always has 10 missing edges. It is worth stressing that with our general notation, these 10 missing edges have exactly the same labels for \( \ddag \)s and \( \ddag \)s. Recall that \( E_+(F) = \{ lc, c1r, l', r, s', sl1, st1 \} \) for a frame \( \{ s : c1, c2 : l, l', r, r', r \} \).

**Proposition 8.8.** Let \( F \) be a frame \( \{ s : c1, c2 : l, l', r, r', r \} \) of a graph \( G \). If a minimum interval supergraph \( \hat{G} \) of \( G \) contains none of \( E_+(F) \), then \( F \) remains unchanged in \( \hat{G} \).
Algorithm interval-edge-deletion(G, k)
INPUT: a graph G and an integer k.
OUTPUT: a set E_− of at most k edges such that G − E_− is a maximum spanning interval subgraph of G; or "NO."
0 if k < 0 then return "NO"; if G is an interval graph then return ∅; E_− = ∅;
1 compute the quotient graph G_Q of G;
2 if G_Q is a clique then \( \text{Since } G \text{ is not an interval graph, at least one module is nontrivial.} \)
2.1 if at least two maximal strong modules do not induce cliques then
find a 4-hole and branch on deleting one edge of it;
2.2 else let M be the only non-clique module, return interval-edge-deletion(G[M], k);
3 if G_Q is an interval graph then \( \text{Note that a graph with no edge is an interval graph.} \)
3.1 if a non-simplicial vertex \( v \in G_Q \) corresponds to a non-clique module M then
find a 4-hole and branch on deleting one edge of it;
3.2 for each maximal strong module M that corresponds to a vertex in SI(G_Q) do
\( E_M = \text{interval-edge-deletion}(G[M], k); \)
if \( E_M = "NO" \) then return "NO";
\( E_− = E_− \cup E_M; \) \( k = k − |E_M|; \)
3.3 return E_−;
4 call decompose-LIG-prime(G_Q);
if a short hole or small CAW is found in G_Q then branch on deleting one edge of it;
\( \text{We have an olive-ring decomposition } K. \)
5 if a module M corresponding to \( v \in V(G_Q) \setminus SI(G_Q) \) does not induce a clique then
find a 4-hole and branch on deleting one edge of it;
6 if G_Q is not already a Helly circular-arc graph then
\( \text{Here vertices/edges are in } G_Q, \text{ thereby corresponding to modules/edges between modules.} \)
6.1 use Lem. 8.3 to find a small CAW or a minimal frame F;
6.2 if a small CAW is found then branch on deleting one edge of it;
6.3 find a shortest \( (U, v′) \)-path path B in \( U_1; \)
6.4 if \( |B| \leq k \) then branch using Lem. 8.5
6.5 find a minimum edge set E_−(F);
branch on deleting one edge in the frame or all edges in E_−(F);
7 call Lem. 6.7

Figure 26: Algorithm for INTERVAL EDGE DELETION

Proof. We need to show that \( \hat{G} \) cannot contain any of the other missed edges of the frame \( F \) not in E_+(F), i.e., \( \{s_l, sr, lr, lr′, l′r\} \). Since \( \hat{G} \) contains neither \( s_l′ \) nor \( l′c_2 \), it cannot contain edge \( sl \); otherwise \( \{sl′l′s\} \) is a 4-hole. A symmetric argument excludes \( sr \). Likewise, the nonexistence of \( l′r′ \) and \( lc_2 \) excludes the edge \( lr′; \) and a symmetric argument excludes \( l′r \). If \( lr \) is an edge of \( \hat{G} \), then there is a 5-hole \( \{cl′l′rr′\} \) or 4-hole \( \{lc_1c_2r\} \) depending on whether \( c_1 = c_2 \) or not. These contradictions conclude the proposition.

If the graph is not chordal, we can find a hole and call Lem. 6.1 to fill it. Now that the graph is chordal we may assume that we have a caterpillar decomposition \( \mathcal{K} \) for \( G \). For any frame \( F \), vertices \( l′ \) and \( r′ \) must be in the main path of the caterpillar, which decide a left-right relation with respect to \( \mathcal{K} \), and more specifically, separate \( \mathcal{K} \) into three parts, the left, the middle, and the right. As long as a minimum interval supergraph \( \hat{G} \) of \( G \) does not contain any edge in \( E_+(F) \), by Prop. 8.8, \( F \) is unchanged. In an interval model \( J \) for \( G \), intervals for \( F \) are arranged as Fig. 4. One should note that in \( J \), however, intervals for the right-hand side vertices might intersect \( I_c \) or even lie to the left of it. Or informally, we might have to bend the right-hand of \( F \) to merge it to the left (see, e.g., Fig. 27), and a symmetric operation might be done to the left-hand side as well.

Figure 27: If modules corresponding to \( u, v \) are sufficiently large, we have to add edges to connect \( U \) to the "left" of \( r′ \). Recall that a vertex in \( G_Q \) corresponds to a module of \( G \).
**Lemma 8.9.** Let $F$ be a minimal frame $(s : c_1, c_2 : 1, v')$ of a prime chordal graph $G$, and let $X$ be a caterpillar decomposition for $G$. Let $S_i = K_i \cap K_{i+1} \setminus N(s)$ for $i < \ell_f(r')$, and let $l$ be an index such that $S_l$ attains the minimum value among all of them. If no minimum interval supergraph of $G$ contains an edge in $E_+(F)$, then there is a minimum interval supergraph of $\hat{G}$ that contains all edges between $s$ and $S_l$.

**Proof.** Let $\hat{G}^+$ be any minimum interval supergraph of $G$; by assumption, $E(\hat{G}^+)$ is disjoint from $E_+(F)$. We show that $\hat{G} = (V(G), E(\hat{G}^+) \setminus \{U_1 \times s \cup s \times S_l\})$ is the claimed minimum interval supergraph of $G$. It is clear that the supergraph defined by the new model contains strictly less edges than $\hat{G}$. Therefore, it suffices to show that $\hat{G}$ is an interval graph, for which we construct an interval model for $G$.

Let $U_1 = \{v : \text{right}(v) \leq \text{right}(l')\}$ and $U_R = \{v : \text{left}(v) \geq \text{left}(r')\}$. Observe that $\{|U_1, U_1, U_R, N(s)|\}$ partitions $V(G)$, and each of them induces a connected subgraph of $G$. Let $I^* = \{v : v \in V(G)\}$, where $I^*_r = [l^*(r), r^*(r)],$ be a normalized interval model for $G$. We will use the same values of $r^*(r')$ and $l^*(r')$ for $l^*(l')$ and $l^*(l)$. We start from characterizing the endpoints of intervals for $U_L$ and $U_R$ that lie between $[r^*(l'), l^*(r')]$. For this purpose, we make following assumptions:

1. The interval for $l'$ is to the left of that for $r'$ in $I^*$, i.e., $r^*(l') < l^*(r')$. Since $l'r'$ is not an edge of $\hat{G}^+$, this assumption is clear.

2. An endpoint $p$ of an interval $I_v$ is integral if and only if one of the following conditions is satisfied:
   - (i) $v \in U_L$; (ii) $v \in U_R$, and $p \leq r^*(l')$; and (iii) $v \in U_R$ and $p > r^*(l')$. These integral endpoints are assumed to be consecutive.

By definition, $l' \in U_L$ and $r' \in U_R$. Hence $l^*(r')$ and $r^*(l')$ are both integral, which separate the real line into three segments. The thickness of a nonintegral point $l$ is defined by

$$\theta_l = \begin{cases} 
\{v \in U_L : l \in I^*_v\} & \text{if } l < r^*(l'), \\
\{v \in U_L : l \in I^*_v\} & \text{if } l > r^*(l'), \\
\{v \in U_L : l \in I^*_v\} & \text{otherwise.}
\end{cases}$$

By definition, for every integer $i$, all points in $(i, i+1)$ have the same thickness. The following claim characterizes intervals for $U_L$ that go beyond the left of $1p(r')$ or even $r^*(l')$.

**Claim 10.** Let $p$ be the leftmost endpoint of all intervals for $U_R$. If $p < r^*(l')$, then $\theta_p < \theta_l$ holds for any nonintegral point $l$ with $|p| < l < r^*(l')$. If $r^*(l') < p < 1p(r')$, then $\theta_p < \theta_l + |\{v \in U_L : l \in I^*_v\}|$ holds for any nonintegral point $l$ with $\max(|p|, r^*(l')) < l < 1p(r')$.

**Proof.** By definition, $p$ is nonintegral. The right-hand terms of both inequalities are the number of intervals for $U_L \cup U_1$ that contain $l$. Here we prove the first assertion, and the second follows analogously. Suppose, for contradiction, that there exists such a point $l$ with $\theta_p \geq \theta_l$. Without loss of generality, we may assume that $\ell$ is the leftmost point of this property. We project the endpoints of intervals for $U_R$ from $[p, l]$ to $[l - \epsilon, l]$, and argue that it defines an interval supergraph of $G$ with strictly less edges than $\hat{G}^+$.

We verify first that the interval graph defined by the new model is a supergraph of $G$. Only intervals for $U_R$ have been modified, while adjacency between any pair of vertices in $U_R$ is retained. Thus, it suffices to consider a pair of vertices $v \in V(G) \setminus U_R$ and $u \in U_R$. Such a vertex $v$ is adjacent to $U_R$ only if $v \sim r'$ in $G$, and thus in the new model, the intervals for $u$ and $v$ remain intersecting. As $\theta_p \geq \theta_l$, the interval supergraph defined by the new model contains strictly less edges than $\hat{G}^+$. This contradiction justifies the claim.

Let $\theta$ be the minimum thickness in $1p(r'), r^*(l')].$ Let $\xi_1$ and $\xi_2$ be the smallest and largest integer such that $\theta_{\xi_1 + 0.5} = \theta = \theta_{\xi_2 + 0.5}$. It is possible that $\xi_1 = \xi_2$ in the original model. In this case, we add 1 to each endpoint of an interval that is $\geq \xi_1 + 1$, and then set $\hat{\xi}_2 = \xi_1 + 1$. After this change we can assume $\hat{\xi}_1 = \xi_2$. Note that then the integral point $\hat{\xi}_2$ is not an endpoint of any interval, and we remark that this fact is immaterial in the argument to follow.

**Claim 11.** Let $p = 1p^*(U_R)$ and $p = r^*(U_L)$. No interval for $v \in U_R$ has an endpoint in $[|p|, \xi_2]$, and no interval for $v \in U_L$ has an endpoint in $[\xi_1, |q|]$.

**Proof.** We show by contradiction, that is, we construct a strictly smaller supergraph of $G$ than $\hat{G}^+$ supposing such a vertex exists. We give here the details on modifying intervals for $U_R$ and $s$, while the modifications
for $U_1$ are symmetrical and hence omitted. Meanwhile, we keep intervals for $N(s)$ and $U_1$ unchanged. Let

$$U'_1 = \{ v \in U_1 : rp^*(v) > rp(1') \}$$

and $U'_2 = \{ v \in U_2 : lp^*(v) < lp(1') \}$.

If $p \geq \xi_2$, then $I_c = I_c^*$ for every $v \in U_2$. In the remaining case, $p$ is non-integral. If $rp(1') < p < \xi_2$ then we project all endpoints for $U_1$ in $[p, \xi_2 + 1]$ to $[\xi_2 + \epsilon, \xi_2 + 1]$. Otherwise, $p < rp(1')$ (noting that $p \neq rp(1')$), we take a point $t$ in $[p, \xi_2]$ such that the number of intervals for $U_2$ that contains $t$ is minimum. We project all endpoints for $U_1$ in $[p, \xi_2]$ to $[p, p]$ and all endpoints for $U_2$ in $[\xi_2, \xi_2 + 1]$ to $[\xi_2, \xi_2 + 1]$. Note that the operation is based on the thickness, which is irrelevant to $U_1^*$ and $U_2^*$. Therefore, the operations are well-defined.

It is easy to verify that this new model defines an interval supergraph of $G$. We now verify its size is no more than $\| \hat{G}^* \|$. Clearly, the adjacency between any pair of vertices in $U_1$ is not changed. We calculate the three parts, namely, between $U_1^*$ and $V(G) \setminus (U_1^* \cup U_2^*)$, between $U_2^*$ and $V(G) \setminus (U_1^* \cup U_2^*)$, and between $U_1^*$ and $U_3^*$.

Consider first edges of $\hat{G}^*$ between $U_1^*$ and $V(G) \setminus (U_1^* \cup U_2^*)$. It suffices to consider edges between $U_1^*$ and $U_2^*$ where $U_0 = \{ v : I_v^* \cap ([p], \xi_2 + 1) \neq \emptyset \}$. Let $I$ be the set of vertices with intervals in $[p, \xi_2]$ (i.e., its new interval is in $[p, [p]]$; $ii$) $U_2$ be the set of vertices with intervals containing $\xi_2$ (i.e., its new interval contains in $[p, [p]]$, $\xi_2$; $iii$) $U_3$ be the set of vertices with intervals in $[\xi_2, \xi_2 + 1]$ (i.e., its new interval is in $[\xi_2, \xi_2 + 1]$).

In the new graph, the number of edges between $U_1^*$ and $V(G) \setminus (U_1^* \cup U_2^*)$ is:

$$|U_1| \cdot \theta_p + |U_2| \cdot |U_0| + |U_3| \cdot \emptyset$$

We now show that $\hat{G}^*$ contains more than this number of edges between $U_1^*$ and $V(G) \setminus (U_1^* \cup U_2^*)$. We consider every $[i, i + 1]$ with $p < i < \xi_2$. If $i$ is the left endpoint of some $v \in U_2$, then we assign it to all edges between $v$ and $\{ u \in U_1^* : i \in I_v^* \}$. If $i + 1$ is the right endpoint of any vertex $v \in U_1$, then we assign to it all edges between $v$ and $\{ u \in U_1^* : i + 1 \in I_v^* \}$. For every vertex $v \in U_1^*$ with $rp^*(v) \in (i, i + 1)$ and $\ell \in \emptyset$, we assign to it all edges between $v$ and $\{ u \in U_0 : [i, i + 1] \in I_v^* \}$. For every vertex $v \in U_1^*$ with $lp^*(v) \in (i, i + 1)$ and $\ell \in \emptyset$, we assign to it all edges between $v$ and $\{ u \in U_0 : [i, i + 1] \in I_v^* \}$. As such we assigned 4 number of edges. As edges assigned to different vertex are disjoint, the total number of assigned edges is a lower bound of edges of $\hat{G}^*$ between $U_1^*$ and $V(G) \setminus (U_1^* \cup U_2^*)$. We point out that the equality holds only when there is no interval for $v \in U_1$ with an endpoint in $[p, \xi_2]$.

A symmetric argument applies to edges between $U_1^*$ and $V(G) \setminus (U_1^* \cup U_2^*)$, and it remains to count edges between $U_1^*$ and $U_2^*$. Note that $\xi_1 < \xi_1 + 1 \leq \xi_2$. By the selection of $\xi_2$, every vertex $v \in U_1^*$ is adjacent to at least $\{ u \in U_1 : t \in I_u^* \}$, and thus in $\hat{G}^*$, it has no more neighbors in $U_1^*$ in $\hat{G}^*$ than $\hat{G}^*$. By a symmetric argument, we can show that every vertex $v \in U_1^*$ has no more neighbors in $U_2^*$ in $\hat{G}^*$ than $\hat{G}^*$.

Summing the three parts up, we conclude that $\| \hat{G}^* \| < \| \hat{G}^* \|$ when either of the claimed conditions is not satisfied. This contradiction concludes the claim.

Adding edges among $U_1$ will not decrease the thickness of any point. Therefore, the fact that $\hat{G}$ is the minimum implies $\hat{G}(U_1) = G(U_1)$. As a result, and since $S_t$ has the minimum size, all nonintegral points that contained by all intervals for $S_t$ have the minimum thickness. In other words, they are either in one of $[\xi_1, \xi_1 + 1]$ and $[\xi_2, \xi_2 + 1]$, or in between. We can thus accordingly set $I_c$ to $[\xi_1 + 1 - \epsilon/2, \xi_2 + 1 - \epsilon]$, $[\xi_2, \xi_2 + 1]$, $[p - \epsilon, p + \epsilon]$ with $p \in \bigcap_{v \in S} I_v^*$. Clearly, this gives an interval supergraph of $G$. By Claim 11 the number of edges added incident to $s$ is minimum. Therefore, there is a minimum interval supergraph satisfying the claimed condition.

We are now ready to present the parameterized algorithm for the INTERVAL COMPLETION problem, which is given in Fig. 29. The proof is similar as that for Thm. 8.4 and thus omitted.

**Theorem 8.10.** Algorithm interval-completion solves INTERVAL COMPLETION in $O(6^k \cdot \|G\|)$ time.
Algorithm interval-completion($G, k$)
  INPUT: a graph $G$ and an integer $k$.
  OUTPUT: a set $E_+$ of at most $k$ edges such that $G + E_+$ is a minimum interval supergraph of $G$; or "NO."

  0   if $k < 0$ then return "NO"; if $G$ is an interval graph then return $\emptyset$; $E_+ = \emptyset$;
  1   if $G$ contains a hole $H$ then call Lem. 6.1 to fill it;
  2   if $G_Q$ is an interval graph then
    2.1  for each maximal module $M$ that does not induce an interval graph do
          $E_M = \text{interval-completion}(G[M], k);$ if $E_M = \text{"NO"}$ then return "$\text{NO}"$;
          $E_+ = E_+ \cup E_M; k = k - |E_+|;$
    2.2  return $E_+$;
  3   call decompose-LIG-prime($G_Q$);
  4   if a small CAW is found in $G_Q$ then branch using Lem. 6.2 to fill it; \ \
    \ We have a caterpillar decomposition $K$ for $G_Q$.
  5   if $G_Q$ is not already a Helly circular-arc graph then
    5.1  use Lem. 8.2 to find a small CAW or a minimal frame $F$;
    5.2  if a small CAW is found in $G_Q$ then branch using Lem. 6.2 to fill it;
    5.3  branch on adding one edge in $E_+$ (F) or using Lem. 8.9.

Figure 29: Algorithm for INTERVAL COMPLETION

9 Concluding remarks

9.1 Further characterizations and recognition of locally interval graphs

From Prop. 1.3 and Thm. 1.4 we can derive the following property of locally interval graph. Here the locally interval graphs not required to be prime.

Theorem 9.1. Let $G$ be a connected locally interval graph.

- If $G$ is not chordal, then there is shortest hole $H$ such that $N[H] = V(G)$;
- otherwise, there is a pair of vertices $u, v$ and a shortest $(u, v)$-path $P$ such that $N[P] = V(G)$.

Moreover, we can in $O(||G||)$ time find this hole $H$ or path $P$.

Similar as Lem. 5.1, such a hole can be locally modified from any shortest hole (Lem. 5.13). For the chordal case, the pair of vertices $u, v$ can be chosen from the end bags of the caterpillar decomposition of $G_Q$, and the path can be obtained from any shortest $(u, v)$-path in a similar way. Thm. 9.1 should be compared with a trivial fact for non-chordal circular-arc graphs (by definition) and dominating pairs in AT-free graphs ([21]), respectively.

We recall again that the existence of an olive-ring decomposition is not a sufficient condition for a prime graph to be a locally interval graph, and thus we do not have a (certifying) recognition algorithm for locally interval graphs.

Conjecture 1. There is a linear-time recognition algorithm that either asserts the graph is a locally interval graph, or detects a 4- or 5-hole, net, sun, rising sun, long claw, or whipping top.

Conjecture 2. There is a linear-time algorithm that, given a locally interval graph, either verifies it is an interval graph, or returns a non-interval subgraph of the minimum number of vertices.

Both conjectures seem very promising, and they, if verified, would imply the following result, which can be viewed as a linear-time approximation algorithm (with ratio $7/4$) for finding a minimum non-interval subgraph. Recall that unless a triangle can be detected in linear time, we cannot detect a minimum non-interval subgraph in linear time. Therefore, this is the best we can expect:

Conjecture 1+2. There is a linear-time algorithm that, given a non-interval graph, returns either a 4- or 5-hole, net, sun, rising sun, long claw, whipping top, or a non-interval subgraph of the minimum number of vertices.
9.2 Improvement and generalizations

The algorithm for INTERVAL EDGE DELETION runs in $k^{O(k)} \cdot \|G\|$ time, which is decided by the disposal of large CAWs. We would like to ask whether it can be improved to $O(c^k \cdot \|G\|)$ to match the runtime of other algorithms. We point out that similar as Prop. 5.2 the 12-direction branching for small CAWs can also be improved as follows.

**Proposition 9.2.** For each small CAW in a graph $G$, there are at most 7 groups of edges such that in any interval subgraph of $G$, at least one of them is totally absent.

![Figure 30: At least one dashed edge or a group of dotted edges is deleted.](image)

We would also like to ask for the existence of subexponential algorithms for the completion problems, especially for the UNIT INTERVAL COMPLETION problem. On related work, one is referred to the subexponential algorithm for chordal completion [32], and recent update for several other graph classes [30]. On the other hand, we surmise that the algorithms for both vertex deletion problems are already asymptotically optimum. The other related question is on the complexity of INTERVAL VERTEX/EDGE DELETION restricted to chordal graphs and other graph classes.

The present paper has been focused on single-operation modification problems. This constraint, however, is not inherent, e.g., the algorithm of Marx [68] for chordal graphs allows both vertex and edge deletions, and our techniques can be revised to furnish such an FPT algorithm for interval graphs. One could even allow simultaneously all three operations. In fact, Cai [14] has formulated the following generic modification problem on a hereditary graph class $\mathcal{G}$:

\[(k_1, k_2, k_3)\text{-MODIFICATION TO } G\]

**Input:** A graph $G$ and three nonnegative integers $k_1$, $k_2$, and $k_3$.

**Task:** Either find i) a subset $V_+ \subseteq V(G)$ of at most $k_1$ vertices, ii) a subset $E_- \subseteq E(G)$ of at most $k_2$ edges, and iii) a subset $E_+ \subseteq E(G)$ of at most $k_3$ missing edges, such that $(V(G) \setminus V_+, E(G) \setminus E(E_- \cup E_+))$ is a graph in $\mathcal{G}$, or report that no such sets exist.

This formulation generalizes all modification problems studied in this paper. It is worth noting that it does not make sense to impose a combined quota on the total number of modifications, as it is then trivially degenerated to the vertex deletion problem. Another natural extension to the the edge modification problems is the interval sandwich problem, which, given two graphs $G_1$ and $G_2$ such that $G_1 \subseteq G_2$, asks for an interval graph $G$ such $G_1 \subseteq G \subseteq G_2$ [41, 49]. The candidates for parameters include $\|G\| - \|G_1\|$ and $\|G_2\| - \|G\|$. On chordal graphs, both the general modification problem and the sandwich problem (parameterized by $\|G\| - \|G_1\|$) are known to be FPT [16, 32]. It is natural to investigate the fixed-parameter tractability of these problems on (unit) interval graphs. We remark that it is not clear whether there always exists a module-preserving solution, and how to dispose of large CAWs in a local way.

It is known that finding minimal non-interval subgraph is closely related to finding Tucker sub-matrices, i.e., minimal sub-matrices that invalidate the consecutive-ones property (CIP). Again, a Tucker sub-matrix can be found in linear time [67, 28, Section 3.3]. But as finding Tucker sub-matrices is trivially harder than finding non-interval subgraphs, there is little hope to find a minimum one in the same time. Can our technique be used to solve problems related to CIP ([28, 71])?

---

7Some authors considered combined edge modifications and they use a single budget for both additions and deletions, e.g., Burzyn et al. [59] proved that it is NP-hard to obtain an interval graph by minimum number of edge modifications. See also Natanzon, Shamir, and Sharan [72] and the thesis of the third author [78] and the references therein. We point out that this formulation is quite different from ours, e.g., the NP-hardness of its addition and/or deletion versions does not carry to it in an easy way.
Acknowledgment. I am grateful to Paul Seymour and Michel Habib for helpful discussion on the complexity of finding shortest holds. I am indebted to Sylvain Guillemot and Yang Liu for careful reading of an early version of this manuscript and their penetrating criticism. I am deeply indebted to Concert YY - Showcase Of Wyman Wong, which accompanies the author during the writing of the whole paper.

References

[1] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic and Discrete Methods, 8(2):277–284, 1987.

[2] Richard Arratia, Eric S. Lander, Simon Tavaré, and Michael S. Waterman. Genomic mapping by anchoring random clones: A mathematical analysis. Genomics, 11(4):806–827, 1991.

[3] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. Journal of the ACM, 48(5):1069–1090, 2001.

[4] Seymour Benzer. On the topology of the genetic fine structure. Proceedings of the National Academy of Sciences, 45(11):1607–1620, 1959.

[5] Jean R. S. Blair and Barry W. Peyton. An introduction to chordal graphs and clique trees. In J. A. George, J. R. Gilbert, and J. W.-H. Liu, editors, Graph Theory and Sparse Matrix Computation, volume 56 of IMA, pages 1–29. Springer-Verlag, 1993.

[6] Guillaume Blin, Romeo Rizzi, and Stéphane Vialette. A faster algorithm for finding minimum Tucker submatrices. Theory of Computing Systems, 51(3):270–281, 2012.

[7] Ivan Bliznets, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Largest chordal and interval subgraphs faster than 2^n. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, ESA, volume 8125 of LNCS, pages 193–204. Springer, 2013.

[8] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, Michael T. Hallett, and Harold T. Wareham. Parameterized complexity analysis in computational biology. Computer applications in the biosciences, 11(1):49–57, 1995.

[9] Hans L. Bodlaender, Ton Kloks, and Dieter Kratsch. Treewidth and pathwidth of permutation graphs. SIAM Journal on Discrete Mathematics, 8(4):606–616, 1995. A preliminary version appeared in ICALP 1993.

[10] Kellogg S. Booth and George 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. A preliminary version appeared in STOC 1975.

[11] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. Graph classes: a survey. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[12] Peter Buneman. A characterization of rigid circuit graphs. Discrete Mathematics, 9(4):205–212, 1974.

[13] Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. NP-completeness results for edge modification problems. Discrete Applied Mathematics, 154(13):1824–1844, 2006.

[14] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. Information Processing Letters, 58(14):171–176, 1996.

[15] Leizhen Cai. Parameterized complexity of vertex colouring. Discrete Applied Mathematics, 127(3):415–429, 2003.

[16] Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. In Ernst W. Mayr and Natacha Portier, editors, Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science, STACS 2014, volume 25 of LIPIcs, pages 214–225. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

[17] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. In Chandra Chekuri, editor, Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 122–141. SIAM, 2014.

[18] Hsien-Chih Chang and Hsueh-I Lu. A faster algorithm to recognize even-hole-free graphs. arXiv:1311.0358, A preliminary version (with a slightly weaker result) appeared in SODA 2012.

[19] Maria Chudnovsky, Ken-ichi Kawarabayashi, and Paul D. Seymour. Detecting even holes. Journal of Graph Theory, 48(2):85–111, 2005.

[20] Derek G. Corneil, Hiryong Kim, Sridhar Natarajan, Stephan Olariu, and Alan P. Sprague. Simple linear time recognition of unit interval graphs. Information Processing Letters, 55(2):99–104, 1995.

[21] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. Asteroidal Triple-free graphs. SIAM Journal on Discrete Mathematics, 10(3):399–430, 1997. A preliminary version appeared in WG 1993.

[22] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. The LBFS structure and recognition of interval graphs. SIAM Journal on Discrete Mathematics, 23(4):1905–1953, 2009.
[23] Christophe Crespelle and Ioan Todinca. An $O(n^2)$-time algorithm for the minimal interval completion problem. *Theoretical Computer Science*, 494:75–85, 2013.

[24] Fabien de Montgolfier. *Décomposition modulaire des graphes. Théorie, extensions et algorithmes*. PhD thesis, Université Montpellier II, Montpellier, Hérault, France, 2003.

[25] Xiaotie Deng, Pavol Hell, and Jing Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing*, 25(2):390–403, 1996.

[26] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, 4 edition, 2010.

[27] Gabriel A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25(1):71–76, 1961.

[28] Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property*. PhD thesis, Friedrich-Schiller-Universität Jena, 2008.

[29] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[30] Pál Grønås Drange, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. Exploring subexponential parameterized complexity of completion problems. To appear in STACS 2014. Full version available online at arXiv:1309.4022., 2013.

[31] Paul Erdős, András Gyárfás, Edward T. Ordman, and Yechezkel Zalcstein. The size of chordal, interval and threshold subgraphs. *Combinatorica*, 9(3):245–253, 1989.

[32] Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM Journal on Computing*, 42(6):2197–2216, 2013. A preliminary version appeared in SODA 2012.

[33] Delbert R. Fulkerson and Oliver A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

[34] Tibor Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungaricae*, 18:25–66, 1967. Translated by Frédéric Maffray and Myriam Preissmann as in *Perfect Graphs*, Jorge L. Ramírez-Alfonsín and Bruce A. Reed, Eds., Wiley, 2001, pp.25-66.

[35] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[36] Fănică Gavril. Algorithms on circular-arc graphs. *Networks*, 4:357–369, 1974.

[37] Alan George and Joseph W. H. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall series in computational mathematics. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[38] Paul W. Goldberg, Martin C. Golumbic, Haim Kaplan, and Ron Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2(1):139–152, 1995.

[39] Martin C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. North-Holland Publishing Co., Amsterdam, The Netherlands, 2004.

[40] Martin C. Golumbic, Haim Kaplan, and Ron Shamir. Graph sandwich problems. *Journal of Algorithms*, 19(3):449–473, 1995.

[41] Martin C. Golumbic and Ron Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM*, 40(5):1108–1133, 1993.

[42] Michel Habib, Ross M. McConnell, Christophe Paul, and Laurent 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.

[43] Michel Habib and Rolf H. Möhring. Treewidth of cocomparability graphs and a new order-theoretic parameter. *Order*, 11(1):47–60, 1994.

[44] Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[45] György Hajós. (problem 65) Über eine art von graphen. *Internationale Mathematische Nachrichten*, 11, 1957.

[46] Pinar Heggernes, Karol Suchan, Ioan Todinca, and Yngve Villanger. Characterizing minimal interval completions. In Wolfgang Thomas and Pascal Weil, editors, *STACS*, volume 4393 of *LNCS*, pages 236–247. Springer, 2007.

[47] Pavol Hell and Jing Huang. Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM Journal on Discrete Mathematics*, 18(3):554–570, 2004.

[48] Wen-Lian Hsu. $O(m \cdot n)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM Journal on Computing*, 24(3):411–439, 1995.

[49] Wen-Lian Hsu and Tze-Heng Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28(3):1004–1020, 1999.
Haim Kaplan, Ron Shamir, and Robert E. Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing*, 28(5):1906–1922, 1999. A preliminary version appeared in FOCS 1994.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

Richard M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 278–285. ACM, 1993.

Toshinobu Kashiwabara and Toshio Fujisawa. An NP-complete problem on interval graphs. In *1979 International Symposium on Circuits and Systems, Proceedings*, pages 82–83, 1979.

Toshinobu Kashiwabara and Toshio Fujisawa. NP-completeness of the problem of finding a minimum-clique-number interval graph containing a given graph as a subgraph. In *1979 International Symposium on Circuits and Systems, Proceedings*, pages 657–660, 1979.

David George Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific Journal of Mathematics*, 28:565–570, 1969.

W. Sean Kennedy and Andrew D. King. Finding a smallest odd hole in a claw-free graph using global structure. *Discrete Applied Mathematics*, 161(16-17):2492–2498, 2013.

Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3-4):115–121, 2000. A preliminary version appeared in WG 1995.

Ton Kloks, Dieter Kratsch, and Jeremy P. Spinrad. On treewidth and minimum fill-in of Asteroidal Triple-free graphs. *Theor. Comput. Sci.*, 175(2):309–335, 1997.

Ekkehard Köhler. Recognizing graphs without asteroidal triples. *Journal of Discrete Algorithms*, 2(4):439–452, 2004. A preliminary version appeared in WG 2000.

Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989.

Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006. A preliminary version appeared in SODA 2003.

Dieter Kratsch and Jeremy P. Spinrad. Between $O(nm)$ and $O^n$. *SIAM Journal on Computing*, 36(2):310–325, 2006. A preliminary version appeared in SODA 2003.

Eric S. Lander and Michael S. Waterman. Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics*, 2(3):231–239, 1988.

C. G. Lekkerkerker and J. Ch. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51:45–64, 1962.

John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

Min Chih Lin, Francisco J. Soulignac, and Jayme Luiz Szwarcfiter. Normal Helly circular-arc graphs and its subclasses. *Discrete Applied Mathematics*, 161(7-8):1037–1059, 2013.

Nathan Lindzey and Ross M. McConnell. On finding Tucker submatrices and Lekkerkerker-Boland subgraphs. In Andreas Brandstädt, Klaus Jansen, and Rüdiger Reischuk, editors, *Revised Papers of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2013*, volume 8165 of LNCS, pages 345–357, 2013.

Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithica*, 57(4):747–768, 2010.

Rolf H. Möhring. Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and boolean functions. *Annals of Operations Research*, 4(1):195–225, 1985.

Rolf H. Möhring. Triangulating graphs without asteroidal triples. *Discrete Applied Mathematics*, 64(3):281–287, 1996.

N. S. Narayanaswamy and R. Subashini. FPT algorithms for consecutive ones submatrix problems. In Gregory Gutin and Stefan Szeider, editors, *Proceedings of the 8th International Symposium on Parameterized and Exact Computation, IPEC 2013*, volume 8246 of LNCS, pages 295–307, 2013.

Assaf Natanzon, Ron Shamir, and Roded Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001. A preliminary version appeared in LNCS vol. 1665, WG 1999.

Tatsuo Ohtsuki, Hajimu Mori, Toshinobu Kashiwabara, and Toshio Fujisawa. On minimal augmentation of a graph to obtain an interval graph. *Journal of Computer and System Sciences*, 22(1):60–97, 1981.
[74] Dömötör Pálvölgyi. Lower bound on the size of maximum interval induced subgraphs of an n-vertex graph G. Theoretical Computer Science Stack Exchange, 2013.

[75] Christos H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. Computing, 16(3):263–270, 1976.

[76] Sheng-Lung Peng and Chi-Kang Chen. On the interval completion of chordal graphs. Discrete Applied Mathematics, 154(6):1003–1010, 2006.

[77] Donald J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In Ronald C. Reed, editor, Graph Theory and Computing, pages 183–217. Academic Press, New York, 1973.

[78] Roded Sharan. Graph Modification Problems and their Applications to Genomic Research. PhD thesis, Tel-Aviv University, Tel Aviv, Israel, 2002.

[79] Jeremy P. Spinrad. Efficient Graph Representations, volume 19 of Fields Institute Monographs. American Mathematical Society, Providence, RI, 2003.

[80] Robert E. Tarjan. Graph theory and gaussian elimination. In James R. Bunch and Donald J. Rose, editors, Sparse Matrix Computations, pages 3–22. Academic Press, New York, 1976. Also available as technical report CS-TR-75-526, Computer Science Department, Stanford University.

[81] Robert E. Tarjan. Data Structures and Network Algorithms. Society for Industrial and Applied Mathematics, 1983.

[82] Robert E. Tarjan and Mihalis Yannakakis. Addendum: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM Journal on Computing, 14(1):254–255, 1985.

[83] Alan C. Tucker. A structure theorem for the consecutive 1’s property. Journal of Combinatorial Theory, Series B, 12(2):153–162, 1972.

[84] Alan C. Tucker. An efficient test for circular-arc graphs. SIAM Journal on Computing, 9(1):1–24, 1980.

[85] Pim van ’t Hof and Yngve Villanger. Proper interval vertex deletion. Algorithmica, 65(4):845–867, 2013.

[86] Yngve Villanger, Pinar Heggernes, Christophe Paul, and Jan Arne Telle. Interval completion is fixed parameter tractable. SIAM Journal on Computing, 38(5):2007–2020, 2009. A preliminary version appeared in STOC 2007.

[87] Michael S. Waterman and Jerrold R. Griggs. Interval graphs and maps of DNA. Bulletin of Mathematical Biology, 48(2):189–195, 1986.

[88] Gerd Wegner. Eigenschaften der Nerven homologisch-einfacher Familien im R^n. PhD thesis, Universität Göttingen, 1967.

[89] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In Howard J. Karloff and Toniann Pitassi, editors, STOC, pages 887–898. ACM, 2012.

[90] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, pages 645–654. IEEE Computer Society, 2010. Full version available at http://www.stanford.edu/~rrwill/tria-mmult.pdf.

[91] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. SIAM Journal on Algebraic and Discrete Methods, 2(1):77–79, 1981.

[92] Peisen Zhang, Eric A. Schon, Stuart G. Fischer, Eftihia Cayanis, Janie Weiss, Susan Kistler, and Philip E. Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. Computer applications in the biosciences, 10(3):309–317, 1994.