Unit Interval Editing is Fixed-Parameter Tractable*

YIXIN CAO†

Abstract

Given a graph \( G \) and integers \( k_1, k_2, \) and \( k_3, \) the unit interval editing problem asks whether \( G \) can be transformed into a unit interval graph by at most \( k_1 \) vertex deletions, \( k_2 \) edge deletions, and \( k_3 \) edge additions. We give an algorithm solving the problem in time \( 2^{O(k \log k)} \cdot (n + m) \), where \( k := k_1 + k_2 + k_3, \) and \( n, m \) denote respectively the numbers of vertices and edges of \( G. \) Therefore, it is fixed-parameter tractable parameterized by the total number of allowed operations.

This implies the fixed-parameter tractability of the unit interval edge deletion problem, for which we also present a more efficient algorithm running in time \( O(4^k \cdot (n + m)) \). Another result is an \( O(6^k \cdot (n + m)) \)-time algorithm for the unit interval vertex deletion problem, significantly improving the best-known algorithm running in time \( O(6^k \cdot n^k) \).

1 Introduction

A graph is a unit interval graph if its vertices can be assigned to unit-length intervals on the real line such that there is an edge between two vertices if and only if their corresponding intervals intersect. Most important applications of unit interval graphs were found in computational biology [2, 14, 15], where data are mainly obtained by unreliable experimental methods. Therefore, the graph representing the raw data is very unlikely to be a unit interval graph, and an important step before understanding the data is to find out and fix the hidden errors. For this purpose various graph modification problems have been formulated: Given a graph \( G \) on \( n \) vertices and \( m \) edges, is there a set of at most \( k \) modifications that make \( G \) a unit interval graph [15, 2, 14]. In particular, edge additions (completion) and edge deletions are used to fix false negatives and false positives respectively, while vertex deletions can be viewed as the elimination of outliers. We have thus three variants, which are all known to be NP-complete [28, 16, 14].

These modification problems to unit interval graphs have been well studied in the framework of parameterized computation, where the parameter is usually the number of modifications. Recall that a graph problem, with a nonnegative parameter \( k, \) is fixed-parameter tractable (FPT) if there is an algorithm solving it in time \( f(k) \cdot (n + m)^{O(1)} \), where \( f \) is a computable function depending only on \( k \) [10]. The problems unit interval completion and unit interval vertex deletion have been shown to be FPT by Kaplan et al. [15] and van Bevern et al. [24] respectively. In contrast, however, the parameterized complexity of the edge deletion version remained open to date, which we settle here. Indeed, we devise single-exponential linear-time parameterized algorithms for both deletion versions.

Theorem 1.1. The problems unit interval vertex deletion and unit interval edge deletion can be solved in time \( O(6^k \cdot (n + m)) \) and \( O(4^k \cdot (n + m)) \) respectively.

The algorithm for unit interval vertex deletion significantly improves the currently best-known parameterized algorithm for it, which takes \( O(6^k \cdot n^k) \) time [25]. Another algorithmic result of [25] is an \( O(n^7) \)-time approximation algorithm of ratio 6 for the problem, which we improve to the following.

Theorem 1.2. There is an \( O(nm) \)-time approximation algorithm of ratio 6 for the minimization version of the unit interval vertex deletion problem.

The structures and recognition of unit interval graphs have been well studied and well understood [9]. It is known that a graph is a unit interval graph if and only if it contains no claw, net, tent, (as depicted in Fig. 1,) or any hole (i.e., a cycle induced by at least four vertices) [21, 27]. Unit interval graphs are

*An extended abstract of this paper has appeared in the proceedings of ICALP 2015.

†Department of Computing, Hong Kong Polytechnic University, Hong Kong, China. Email: yixin.cao@polyu.edu.hk. This research was begun while the author was a postdoctoral research fellow at Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), supported by the European Research Council (ERC) under the grant 280152 and the Hungarian Scientific Research Fund (OTKA) under the grant NK105645.
We revisit the relation between unit interval graphs and some subclasses of proper circular-arc graphs, who also made explicit the use of bounded-search tree in disposing of finite forbidden induced subgraphs. Although this phase is conceptually intuitive, how to efficiently carry it out is rather nontrivial, e.g., the observation and the parameterized algorithm of Marx [19] for the chordal vertex deletion problem immediately imply the fixed-parameter tractability of the unit interval vertex deletion problem: one may break first all induced claws, nets, and tents, and then call Marx’s algorithm. Here we are using the hereditary property of unit interval graphs—recall that a graph class is hereditary if it is closed under taking induced subgraphs. However, neither approach can be adapted to the edge deletion version in a simple way. Compared to completion that needs to add $\Omega(\ell)$ edges to fill a $C_\ell$ (i.e., a hole of length $\ell$) in, an arbitrarily large hole can be fixed by a single edge deletion. On the other hand, the deletion of vertices leaves an induced subgraph, which allows us to focus on holes once all claws, nets, and tents have been eliminated; however, the deletion of edges to fix holes of a $\{\text{claw, net, tent}\}$-free graph may introduce new claw(s), net(s), and/or tent(s). Therefore, although a parameterized algorithm for the chordal edge deletion problem has also been presented by Marx [19], there is no obvious way to use it to solve the unit interval edge deletion problem.

Direct algorithms for unit interval vertex deletion were later discovered by van Bevern et al. [24] and van’t Hof and Villanger [25], both of which use a two-phase approach. The first phase of their algorithms breaks all forbidden induced subgraphs on at most six vertices. Note that this differentiates from the aforementioned simple approach in that it breaks not only claws, nets, and tents, but all $C_\ell$’s with $\ell \leq 6$. Although this phase is conceptually intuitive, how to efficiently carry it out is rather nontrivial, e.g., the simple brute-force way used by [24, 25] introduces an $n^6$ factor to the running time. Their approaches diverse completely in the second phase. Van Bevern et al. [24] used a complicated iterative compression procedure that has a very high time complexity, while van’t Hof and Villanger [25] showed that after the first phase, the graph is a proper circular-arc graph (its definition is postponed to Section 2), on which the problem is linear-time solvable. In the conference presentation where Villanger first announced the result, it was claimed that this settles the edge deletion version as well. However, the claimed result has not been materialized: it appears neither in the conference version [26] (which has a single author) nor in the significantly revised and extended journal version [25]. Unfortunately, this unsubstantiated claim did get circulated.

Although the algorithm of [25] is nice and simple, its self-contained proof is excruciatingly complex. We revisit the relation between unit interval graphs and some subclasses of proper circular-arc graphs, and study it in a structured way. In particular, we observe that unit interval graphs are precisely the intersection of chordal graphs and proper Helly circular-arc graphs. As a matter of fact, unit interval graphs can also be viewed as “proper Helly interval graphs,” thereby making a natural subclass of proper Helly circular-arc graphs. The full containment relations are summarized in Fig. 2. They inspire us to develop the parameterized algorithms stated in Theorem 1.1, though some nontrivial analysis is required to obtain the time bound for unit interval edge deletion.

Van Bevern et al. [24] showed that the unit interval vertex deletion problem remains NP-hard on $\{\text{claw, net, tent}\}$-free graphs, but it is not hard on $\{\text{claw, net}\}$-free graphs. In particular, we observe that unit interval graphs are precisely the intersection of chordal graphs and proper Helly circular-arc graphs. As a matter of fact, unit interval graphs can also be viewed as “proper Helly interval graphs,” thereby making a natural subclass of proper Helly circular-arc graphs. The full containment relations are summarized in Fig. 2. They inspire us to develop the parameterized algorithms stated in Theorem 1.1, though some nontrivial analysis is required to obtain the time bound for unit interval edge deletion.
Normal Helly cag Chordal graphs

Proper Helly cag Interval graphs (ig)

Unit Helly cag

Unit ig = Proper ig

Figure 2: Containment relation of related graph classes.
Normal Helly circular-arc \cap Chordal = Interval.
Proper Helly circular-arc \cap Chordal = Unit Helly circular-arc \cap Chordal = Unit interval.

net, tent -free graphs. After deriving a polynomial-time algorithm for the problem on \{claw, net, tent, C_4, C_5, C_6\}-free graphs, van’t Hof and Villanger [25] asked for its complexity on \{claw, net, tent, C_4\}-free graphs. Note that these graphs are not a subset of proper (Helly) circular-arc graphs, evidenced by the W_5 (Fig. 1d). We answer this question by characterizing connected \{claw, net, tent, C_4\}-free graphs that are not proper Helly circular-arc graphs: we show that such a graph must be like a W_5, on which the problem can also be solved in linear time.

Theorem 1.3. The problems unit interval vertex deletion and unit interval edge deletion can be solved in O(n + m) time on \{claw, net, tent, C_4\}-free graphs.

We remark that the techniques we developed in previous work [7] can also be used to derive Theorems 1.1 and 1.2. Those techniques, designed for interval graphs, are nevertheless far more complicated than necessary when applied to unit interval graphs. The approach we used in the current work, i.e., based on structural properties of proper Helly circular-arc graphs, is tailored for unit interval graphs, hence simpler and more natural. Another benefit of this approach is that it enables us to devise a parameterized algorithm for the general modification problem to unit interval graphs, which allows all three types of modification operations. This formulation generalizes all the three single-type modifications, and is also natural from the aspect of the aforementioned applications for de-noising data, where different types of errors are commonly found coexisting. Indeed, the assumption that the input data contains only a single type of errors is somewhat counterintuitive. Formally, given a graph G, the unit interval editing problem asks whether there are a set V_ of at most k_1 vertices, a set E_ of at most k_2 edges, and a set E_+ of at most k_3 non-edges, such that the deletion of V_ and E_ and the addition of E_+ make G a unit interval graph. We show that it is FPT, parameterized by the total number of allowed operations.

Theorem 1.4. The unit interval editing problem can be solved in time 2^{O(k \log k)} \cdot (n + m), where k := k_1 + k_2 + k_3.

By and large, our algorithm for unit interval editing again uses the two-phase approach. However, we are not able to show that it can be solved in polynomial time on proper Helly circular-arc graphs. Therefore, in the first phase, we do away with not only claws, nets, nets, and C_4’s, also all holes of length at most k_3 + 3, in all possible ways. The high exponential factor in the running time is due to purely this phase. After that, every hole has length at least k_3 + 4, and has to be fixed by vertex or edge deletions. We show that an inclusion-wise minimal solution of this reduced graph does not add edges, and the problem can then be solved in linear time.

The study of general modification problems was initiated by Cai [4], who observed that the problem is FPT if the objective graph class has a finite number of minimal forbidden induced subgraphs. More challenging is to devise parameterized algorithms for those graph classes whose minimal forbidden induced subgraphs are not finite. Prior to this paper, the only known nontrivial graph class on which the general modification problem is FPT is the chordal graphs [8]. Theorem 1.4 extends this territory by including another well-studied graph class. As a corollary, Theorem 1.4 implies the fixed-parameter tractability of the unit interval edge editing problem, which allows both edge operations but not vertex deletions [3]—we can try every combination of k_2 and k_3 as long as k_2 + k_3 does not exceed the given bound.
Organization. The rest of the paper is organized as follows. Section 2 presents combinatorial and algorithmic results on \{claw, net, tent, \(C_4\)\}-free graphs. Sections 3 and 4 present the algorithms for unit interval vertex deletion and unit interval edge deletion respectively (Theorems 1.1–1.3). Section 5 extends them to solve the general editing problem (Theorem 1.4). Section 6 closes this paper by discussing some possible improvement and new directions.

2 \{Claw, net, tent, \(C_4\)\}-free graphs

All graphs discussed in this paper are undirected and simple. A graph \(G\) is given by its vertex set \(V(G)\) and edge set \(E(G)\), whose cardinalities will be denoted by \(n\) and \(m\) respectively. All input graphs in this paper are assumed to be connected, and hence \(n = O(m)\). For \(\ell \geq 4\), we use \(C_\ell\) to denote an hole on \(\ell\) vertices; if we add a new vertex to a \(C_\ell\) and make it adjacent to no or all vertices in the hole, then we end with a \(C_\ell^*\) or \(W_\ell\), respectively. For a hole \(H\), we have \(|V(H)| = |E(H)|\), denoted by \(|H|\). The complement graph \(\overline{G}\) of a graph \(G\) is defined on the same vertex set \(V(G)\), where a pair of vertices \(u\) and \(v\) is adjacent if and only if \(uv \not\in E(G)\); e.g., \(W_5 = C_5^*\), and depicted in Fig. 1f is \(C_6\), the complement of \(C_6\).

An interval graph is the intersection graph of a set of intervals on the real line. A natural way to extend interval graphs is to use arcs and a circle in the place of intervals and the real line, and the intersection graph of arcs on a circle is a \textit{circular-arc graph}. The set of intervals or arcs is called an \textit{interval model} or \textit{arc model} respectively, and it can be specified by their 2n endpoints. In a \textit{unit interval} or a \textit{unit arc model}, every interval or arc has length 1. An interval or arc model is \textit{proper} if no interval or arc in it properly contains another interval or arc. A graph is a \textit{unit/proper interval/circular-arc graph} if has a unit/proper interval/arc model respectively. The forbidden induced subgraphs of these graph classes have been long known [27, 23].

Theorem 2.1 ([27]). A graph is a unit interval graph if and only if it contains no claw, net, tent, or any hole.

Clearly, any (unit/proper) interval model can be viewed as a (unit/proper) arc model with some point uncovered, and hence all (unit/proper) interval graphs are always (unit/proper) circular-arc graphs. A unit interval/arc model is necessarily proper, but the other way does not hold true in general. A well-known result states that a proper interval model can always be made unit, and thus these two graph classes coincide [21, 27]. This fact will be heavily used in the present paper; e.g., most of our proofs consist in modifying a proper arc model into a proper interval model, which represents the desired unit interval graph. On the other hand, it is easy to check that the tent is a proper circular-arc graph but not a unit circular-arc graph. Therefore, the class of unit circular-arc graphs is a proper subclass of proper circular-arc graphs. An arc model is \textit{Helly} if every set of pairwise intersecting arcs has a common intersection. A circular-arc graph is \textit{proper Helly} if it has an arc model that is both proper and Helly.\footnote{A word of caution is worth on the definition of proper Helly circular-arc graphs. One graph might admit two arc models, one being proper and the other Helly, but no arc model that is both proper and Helly; for example, the tent and the \(W_4\). Therefore, the class of proper Helly circular-arc graphs is not equivalent to the intersection of proper circular-arc graphs and Helly circular-arc graphs, but a proper subclass of it.}

The set of forbidden induced subgraphs of proper Helly circular-arc graphs were characterized by Lin et al. [17].

Theorem 2.2 ([17, 23]). A graph is a proper Helly circular-arc graph if and only if it contains no claw, net, tent, \(W_4, W_5, C_6\), or \(C_4^*\) for \(\ell \geq 4\).

A technical remark is worth here. What was characterized by Lin et al. [17, Corollary 5] is actually the proper circular-arc graphs that are not proper Helly circular-arc graphs: they must contain a \(W_4\) or tent. On the other hand, Tucker [23] had characterized the forbidden induced subgraphs of proper circular-arc graphs, which include, aside from those stated in Theorem 2.2 (claw, net, \(W_5, C_6^*\), and \(C_4^*\) for \(\ell \geq 4\), \(C_{2\ell}\), and \(C_{2\ell+1}^*\) for \(\ell \geq 4\). To see Theorem 2.2, note that for \(\ell \geq 4\), both \(C_{2\ell}\) and \(C_{2\ell+1}^*\) contain a \(W_4\), i.e., two non-incident edges and another independent vertex \(v\). Let \(h_1, h_2, \ldots, h_\ell\) denote the vertices in the hole \(C_\ell\). If \(\ell \geq 7\), then edges \(h_1h_2\) and \(h_4h_5\) are non-incident; therefore, in a \(C_7^*\), the vertex not in the hole can be the \(v\), while in an even longer hole, \(h_7\) can be the \(v\). Theorems 2.1 and 2.2 have the following consequence.

\footnote{The reason we choose “unit” over “proper” in the title of this paper is twofold. On the one hand, the applications we are interested in are more naturally represented by unit intervals. On the other hand, we want to avoid the use of “proper interval subgraphs,” which is ambiguous.}
Corollary 2.3. If a proper Helly circular-arc graph is chordal, then it is a unit interval graph.

The following can also be derived from Theorems 2.1 and 2.2. We omit its proof here, as it will be an easy consequence of Theorem 2.5.

Proposition 2.4. Every connected \{claw, net, tent, C_4, C_5\}-free graph is a proper Helly circular-arc graph.

Note that in Proposition 2.4, as well as most combinatorial statements to follow, we need the graph to be connected. Circular-arc graphs are not closed under taking disjoint unions. If a (proper Helly) circular-arc graph is not chordal, then it is necessarily connected. In other words, a disconnected (proper) circular-arc graph must be a (unit) interval graph.

We say that a recognition algorithm (for a graph class) is certifying if it provides a minimal forbidden induced subgraph of the input graph when it is determined to be not in this class. A linear-time certifying algorithm for recognizing proper Helly circular-arc graphs was given by Lin et al. [17], from which one can derive a linear-time algorithm for detecting an induced claw, net, tent, or \(C_5\) from a graph that is not a proper Helly circular-arc graph. This is sufficient for us to derive the fixed-parameter tractability of the problems under concern. Even so, we would take pain to prove slightly stronger results (than Proposition 2.4) on \(\mathcal{F}\)-free graphs, where \(\mathcal{F}\) denotes the set \{claw, net, tent, \(C_4\)\}. The purpose is threefold. First, it enables us to answer the question asked by van’t Hof and Villanger [25], i.e., the complexity of unit interval vertex deletion on \(\mathcal{F}\)-free graphs, thereby more accurately delimiting the complexity border of the problem. Second, as we will see, the disposal of \(C_5\)'s would otherwise dominate the branching step in our algorithm for unit interval edge deletion, so excluding them enables us to obtain better exponential dependence on \(k\) in the running time. Third, the combinatorial characterization might be of its own interest.

We use fat \(W_5\) to denote a graph obtained from a \(W_5\) by replacing each of its six vertices by a distinct clique, where the five cliques replacing the hole vertices make a fat hole, and the other clique is its hub. In principle, to verify whether a graph is a fat \(W_5\), one can calculate in linear time its modular decomposition tree [22], and then check its modules. A fat \(W_5\) has at most seven nontrivial strong modules, including the six cliques (some of them might be trivial), and the union of the five cliques in the fat hole. As shown in the following theorem, this can be done in a simpler way.

Theorem 2.5. Let \(G\) be a connected graph.

1. If \(G\) is \(\mathcal{F}\)-free, then it is either a fat \(W_5\) or a proper Helly circular-arc graph.
2. In \(O(m)\) time we can either detect an induced subgraph of \(G\) in \(\mathcal{F}\), partition \(V(G)\) into six cliques constituting a fat \(W_5\), or build a proper and Helly arc model for \(G\).

Proof. We prove only assertion (2) using the algorithm described in Fig. 3, and its correctness implies assertion (1). The algorithm starts by calling the certifying algorithm of Lin et al. [17] for recognizing proper Helly circular-arc graphs (step 0). It enters one of steps 1–4, or 6 based on the outcome of step 0. Here the subscripts of vertices in a hole \(C_5\) should be understood to be modulo \(\ell\).

If the condition of any of steps 1–4 is satisfied, then either a proper and Helly arc model or a subgraph in \(\mathcal{F}\) is returned. The correctness of steps 1–3 is straightforward. Step 4.1 can find the path because \(G\) is connected; possibly \(v = x\), which is irrelevant in steps 4.2–4.4. Otherwise, the outcome of step 0 must be a \(W_5\); let it be \(H\) and \(v\). Steps 5–7 either detect an induced subgraph of \(G\) in \(\mathcal{F}\) or partition \(V(G)\) into six cliques constituting a fat \(W_5\). Step 6 scans vertices not in the \(W_5\) one by one, and proceeds based on the adjacency between \(x\) and \(H\). In step 6.1, \(H\) and \(x\) make a \(C_5\), which means that we can proceed exactly the same as step 4. Note that the situation of step 6.4 is satisfied if \(x\) is adjacent to four vertices of \(H\). If none of the steps 6.1 to 6.5 applies, then \(x\) has precisely three neighbors in \(H\) and they have to be consecutive. This is handled by step 6.6.

Steps 0 and 4 takes \(O(m)\) time. Step 6 scans the adjacency list of each vertex once, and hence takes \(O(m)\) time in total. Steps 1, 2, 3, 5, and 7 need only \(O(1)\) time. Therefore, the total running time of the algorithm is \(O(m)\). This concludes the proof.

Implied by Theorem 2.5, a connected \{claw, net, tent, \(C_4\), \(W_5\)\}-free graph is a proper Helly circular-arc graph, which in turns implies Proposition 2.4. At this point a natural question appealing to us is the relation between connected \{claw, net, tent, \(C_4\), \(C_5\)\}-free graphs and unit Helly circular-arc graphs. Recall that the class of unit interval graphs is a subclass of unit Helly circular-arc graphs, on which we have a similar statement as Corollary 2.3, i.e., a unit Helly circular-arc graph that is chordal is a unit interval
Lemma 2.7. a proper Helly circular-arc graph in linear time. This is another important step of our algorithm for the other words, the model must be normal as well). These observations enable us to find a shortest hole from 

\[ \text{[17, 6]} \] In a proper and Helly arc model for a non-chordal graph

Proposition 2.6. necessarily covers the circle, and it is minimal. Interestingly, the converse holds true as well.

G circular-arc graph, we need to exploit the arc models. A unit interval model is always a proper and Helly arc model, a subgraph in \( F \), or six cliques making a fat \( W_5 \).

Algorithm recognize-\( F \)-free(\( G \))

Input: a connected graph \( G \). Output: a proper and Helly arc model, a subgraph in \( F \), or six cliques making a fat \( W_5 \).

0. call the recognition algorithm for proper Helly circular-arc graphs [17];

1. if a proper and Helly arc model \( A \) is found then return it;

2. if a claw, net, or tent is found then return it;

3. if a \( W_4, C_4 \), or \( C_6 \) is found then return a \( C_4 \) contained in it;

4. if a \( C_4 \) with hole \( H \) and isolated vertex \( v \) is found then

4.1. use breadth-first search to find a shortest path \( v \cdots xyh_i \) from \( v \) to \( H \);

4.2. if \( y \) has a single neighbor \( h_i \) in \( H \) then return \( \{h_i, y, h_{i-1}, h_{i+1}\} \) a claw;

4.3. if \( y \) has only two neighbors on \( H \) that are consecutive, say \( \{h_i, h_{i+1}\} \) then return net \( \{x, y, h_{i-1}, h_i, h_{i+1}, h_{i+2}\} \);

4.4. return claw \( \{y, x, h_i, h_j\} \) where \( h_i \in N(y) \cap H \setminus \{h_{i-1}, h_i, h_{i+1}\} \);

\% The outcome of step 0 must be a \( W_5 \); let it be \( H \) and \( v \). All subscripts of \( h_i \) and \( K_i \) are modulo 5.

5. \( K_0 \leftarrow \{h_0\}; K_1 \leftarrow \{h_1\}; K_2 \leftarrow \{h_2\}; K_3 \leftarrow \{h_3\}; K_4 \leftarrow \{h_4\}; K_v \leftarrow \{v\};

6. for each vertex \( x \) not in the \( W_5 \) do

6.1. if \( x \) is not adjacent to \( H \) then similar as step 4 (\( H \) and \( x \) make a \( C_5 \));

6.2. if \( x \) has a single neighbor \( h_i \) in \( H \) then return claw \( \{h_i, x, h_{i-1}, h_{i+1}\} \);

6.3. if \( x \) is only adjacent to \( h_i, h_{i+1} \) in \( H \) then

if \( xv \in E(G) \) then return claw \( \{v, h_{i-1}, h_{i+1}, x\} \);

else return tent \( \{x, h_i, h_{i+1}, h_{i-1}, v, h_{i+2}\} \);

6.4. if \( x \) is adjacent to \( h_{i-1}, h_{i+1} \) but not \( h_i \) then return \( xh_{i-1}h_{i+1} \) as a \( C_4 \);

6.5. if \( x \) is adjacent to all vertices in \( H \) then

if \( xy \in E(G) \) for some \( y \in K_v \) or \( K_1 \) then return \( xh_{i-1}yh_{i+1} \) as a \( C_4 \);

else add \( x \) to \( K_v \); \% \( x \) is adjacent to all vertices in the six cliques.

6.6. else \% Hereafter \( \lvert N(x) \rvert \cap H = 3 \); let them be \( h_{i-1}, h_i, h_{i+1} \).

if \( xy \not\in E(G) \) for some \( y \in K_i \) or \( K_v \) then return \( xh_{i-1}yh_{i+1} \) as a \( C_4 \);

if \( xy \not\in E(G) \) for some \( y \in K_{i-1} \) then return claw \( \{v, h_{i-1}, h_{i+2}, h_{i+1}\} \);

if \( xy \not\in E(G) \) for some \( y \in K_{i+1} \) then return claw \( \{v, h_{i-2}, h_{i+1}, x\} \);

if \( xy \not\in E(G) \) for some \( y \in K_{i-2} \) then return \( xh_{i-1}h_{i+2}y \) as a \( C_4 \);

if \( xy \in E(G) \) for some \( y \in K_{i+2} \) then return \( xh_{i-1}h_{i+2}y \) as a \( C_4 \);

else add \( x \) to \( K_i \); \% \( K_v, K_{i-1}, K_i, K_{i+1} \subseteq N(x) \) and \( K_{i-2}, K_{i+2} \cap N(x) = \emptyset \).

7. return the six cliques.

Figure 3: Recognizing \( F \)-free graphs.

graph. However, a connected \{claw, net, tent, \( C_4, C_5 \}-free graph that is not a unit Helly circular-arc graph can be constructed as follows: starting from a \( C_4 \) with \( \ell \geq 6 \), for each edge \( h_ih_{i+1} \) in the hole add a new vertex \( v \) and two new edges \( v\mid h_i, v\mid h_{i+1} \). (This is actually the CI(\( \ell, 1 \)) graph defined by Tucker [23]; see also [17].) Therefore, Proposition 2.4 and Theorem 2.5 are the best we can expect in this sense.

Note that a \( C_4 \) is a proper Helly circular-arc graph. Thus, the algorithm of Theorem 2.5 is not yet a certifying algorithm for recognizing \( F \)-free graphs. To detect an induced \( C_4 \) from a proper Helly circular-arc graph, we need to exploit the arc models. A unit interval model is always a proper and Helly arc model, but a unit interval graph might have an arc model that is neither proper nor Helly. On the other hand, if a proper Helly circular-arc graph \( G \) is not chordal, then the set of arcs for vertices in a hole necessarily covers the circle, and it is minimal. Interestingly, the converse holds true as well.

Proposition 2.6. [17, 6] In a proper and Helly arc model for a non-chordal graph \( G \), a minimal set of arcs whose union covers the circle corresponds to a hole of \( G \).

Proposition 2.6 forbids a proper and Helly arc model to have two arcs that cover the entire circle (in other words, the model must be normal as well). These observations enable us to find a shortest hole from a proper Helly circular-arc graph in linear time. This is another important step of our algorithm for the unit interval editing problem—it is stronger than detecting \( C_4 \)'s, which, if existent, must be the shortest.

Lemma 2.7. There is an \( O(m) \)-time algorithm for finding a shortest hole of a proper Helly circular-arc graph.
Before proving Lemma 2.7, we need to introduce some notation. In this paper, all intervals and arcs are closed, and no distinct intervals or arcs will be allowed to share an endpoint in the same model; these restrictions do not sacrifice any generality. In an interval model, the interval $I_v$ for vertex $v$ is given by $[lp(v), rp(v)]$, where $0 \leq lp(v) < rp(v)$ are its left and right endpoints respectively. In an arc model, the arc $A_v$ for vertex $v$ is given by $[ccp(v), cp(v)]$, where $ccp(v)$ and $cp(v)$ are its counterclockwise and clockwise endpoints respectively. All points in an arc model are assumed to be nonnegative; in particular, they are between 0 (inclusive) and $\ell$ (exclusive), where $\ell$ is the perimeter of the circle. We point out that possibly $ccp(v) > cp(v)$; such an arc $A_v$ necessarily passes through the point 0. Note that rotating all arcs in the model does not change the intersections among them. Thus we can always assume that a particular arc contains or avoids the point 0. We say that an arc model is canonical if the perimeter of the circle is $2n$, and every endpoint is a different integer in $\{0, 1, \ldots, 2n - 1\}$.

Each point $x$ in an interval model $A$ or arc model $J$ defines a clique, denoted by $K_A(x)$ or $K_J(x)$ respectively, which is the set of vertices whose intervals or arcs contain $x$. There are at most $2n$ distinct cliques defined as such. If the model is Helly, then they include all maximal cliques of this graph $[13]$. For any point $\rho$ in an interval or arc model, we can find a small positive value $\epsilon$ such that the only possible endpoint in the interval $[\rho - \epsilon, \rho + \epsilon]$ is $\rho$. Here the value of $\epsilon$ should be understood as a function—depending on the interval/arc model as well as the point $\rho$—instead of a constant.

In a proper and Helly arc model $A$ for graph $G$, if $uv \in E(G)$, then exactly one of $ccp(v)$ and $cp(v)$ is contained in $A_u$. Thus, we can define a left-right relation for each pair of intersecting arcs, which can be understood from the viewpoint of an observer placed at the center of the model. We say that arc $A_v$ intersects arc $A_u$ from the left when $cp(v) \in A_u$, denoted by $v \rightarrow u$.

| Algorithm shortest-hole $(G, A)$ |
|----------------------------------|
| **Input:** a proper and Helly arc model $A$ for a non-chordal graph $G$. |
| **Output:** a shortest hole of $G$. |
| 0. make $A$ canonical where 0 is $ccp(v)$ for some $v$; |
| 1. for $i = 1, \ldots, cp(v) - 1$ do |
| 1.1. if $i$ is $ccp(x)$ then create a new array $\{x\}$; |
| \// these arrays are circularly linked so that the next of the last array is the first one. |
| 2. $w \leftarrow \bot$; $U \leftarrow$ the first array; |
| 3. for $i = cp(v) + 1, \ldots, 2n - 1$ do |
| 3.0. $z \leftarrow$ the last vertex of $U$; |
| 3.1. if $i$ is $ccp(x)$ then $w \leftarrow x$; continue$^\dagger$; |
| 3.2. if $i \neq cp(z)$ then continue; |
| 3.3. if $w = \bot$ then delete $U$; $U \leftarrow$ the next array of $U$; |
| 3.4. if $w \neq \bot$ then append $w$ to $U$; $w \leftarrow \bot$; $U \leftarrow$ the next array of $U$; |
| 4. for each $U$ till the last array do |
| 4.1. if the first and last vertices of $U$ are adjacent then return $U$; |
| 5. return $U \cup \{v\}$; |
| $\bot$ is the last array. |

$\dagger$: i.e., ignore the rest of this iteration of the for-loop and proceed to the next iteration.

Figure 4: Finding a shortest hole in a proper Helly circular-arc graph.

According to Lemma 2.6, every hole needs to visit some vertex in $K_A(\alpha)$ for any fixed point $\alpha$ in the model. Therefore, to find a shortest hole in $G$, it suffices to find for each $x \in K_A(\alpha)$ a shortest hole through $x$, and returns the shortest among them.

**Proof of Lemma 2.7.** The algorithm described in Fig. 4 finds for each $x \in K_A(cp(v) + 0.5)$ a shortest hole through $x$, and returns the shortest among them. Step 1 creates $|K_A(cp(v) + 0.5)|$ arrays, each starting with a distinct vertex in $K_A(cp(v) + 0.5)$, and they are ordered such that their (counter)clockwise endpoints are increasing. The main job of this algorithm is done in step 3, which works by scanning the endpoints from 0 to $2n - 1$. During this step, $w$ is the new vertex to be processed, and $U$ is the current array. Each new vertex is added to at most one array, while each array is either dropped or extended. We use $\bot$ as a dummy vertex, which means that no new vertex has been met after the last one has been put into the previous array. Step 3.1 records the rightmost arc that has been scanned. Once the clockwise endpoint of
the last vertex \( z \) of the current array \( U \) is met, \( w \) is appended to \( U \) (step 3.4). However, if \( w = \perp \), then we drop this array from further consideration (step 3.3). If after step 3, one of the arrays already induces a hole (i.e., the first and last vertices are adjacent), then it is returned in step 4.1. Otherwise, \( U \) does not induce a hole, and step 5 returns the hole induced by \( U \) and \( v \).

We now verify the correctness of the algorithm. It suffices to show that the length of the found hole is \( \min \{ h(x) : x \in K_A(\text{cp}(v) + 0.5) \} \), where \( h(x) \) denotes the length of the shortest holes through \( x \). We argue first that for each \( x \in K_A(\text{cp}(v) + 0.5) \), there is a hole that has length \( h(x) \) and visits all vertices in the array containing \( x \). Note that the vertex \( w \) added to an array (step 3.4) must have the rightmost arc that intersects \( A_z \) (step 3.1), where \( z \) is the last element in the array before this addition is made. Let \( h_0 h_1 h_2 \cdots \) be a shortest hole with \( x = h_0 \), and let \( A_{h_1} \) be the rightmost arc that intersects \( A_{h_2} \), then replacing \( h_1 \) by \( y \) gives a hole of the same length. Note that either \( y = h_1 \) or \( h_1 \rightarrow y \). The vertex \( y \) is adjacent to \( h_2 \) because the model is proper, and is not adjacent to other vertices (except \( h_0 \) and \( h_2 \)) in the hole because the hole is the shortest. This argument can be applied inductively. After step 3, the last vertex of every array is adjacent to \( v \); or more specifically, its arc contains 0. Now that \( v \) is adjacent to both the first and last vertices of each array, either \( U \) or \( U \cup \{ v \} \) induces a hole. This hole must have length \( h(x) \).

We argue then that the found hole is the shortest among all the holes decided by the remaining arrays. Note that step 1 creates \( \{ K_A(\text{cp}(v) + 0.5) \} \) arrays, and the number may decrease but never increase in step 3. Since step 3 processes arrays in a circular order starting from the first one, and each array is either deleted (step 3.3) or extended by adding one vertex (step 3.4), their sizes differ by at most one. In particular, at the end of step 3, if the current array \( U \) is not the first, then \( U \), as well as all the succeeding arrays, has one less element than its predecessor(s). If \( U \) or any array after \( U \) induces a hole, then its length is \( |U| \) and is the shortest (step 4.1). Otherwise, a hole needs at least \( |U| + 1 \) vertices (step 5).

It remains to argue that for any array \( U \) deleted in step 3.3, the found hole is no longer than \( h(x) \), where \( x \) is the first vertex of \( U \). All status in the following is referred at the moment before \( U \) is deleted (i.e., before step 3.3 that deletes \( U \)). Let \( z \) be the last vertex of \( U \). Let \( U' \) be the array that is immediately preceding \( U \), and let \( z' \) be its last vertex. Note that there is no arc with a counterclockwise endpoint between \( \text{cp}(z') \) and \( \text{cp}(z) \), as otherwise \( w \neq \perp \) and \( U \) would not be deleted. Therefore, any arc intersecting \( A_z \) starting from the right makes \( A_{z'} \) from the right. With the argument as above, it can be inferred that there is a hole \( H \) that has length \( h(x) \) and contains \( U \cup \{ z' \} \). We find a hole through \( U' \) of the same length as follows. If \( U' \) is not the first array, then \( |U'| = |U| + 1 \), and we replace \( U \cup \{ z' \} \) by \( U' \). Otherwise, \( |U'| = |U| \), and we replace \( U \cup \{ z' \} \) by \( \{ v \} \cup U' \). It is easy to verify that after this replacement, \( H \) remains a hole of the same length.

We now analyze its running time. Each of the 2\( n \) endpoints is scanned once, and each vertex belongs to at most one array. Using linked list to store an array, the addition of a new vertex can be implemented in constant time. Using a circularly linked list to organize the arrays, we can find the next array or delete the current one in constant time. It follows that the algorithm can be implemented in \( O(m) \) time.

## 3 Vertex deletion

We say that a set \( V_- \) of vertices is a hole cover of \( G \) if \( G - V_- \) is chordal. The hole covers of proper Helly circular-arc graphs are characterized by the following lemma. Noting that any local part of a proper and Helly arc model “behaves similarly” as an interval model, this is an easy extension of the clique separator property of interval graphs [12].

**Lemma 3.1.** Let \( A \) be a proper and Helly arc model for a non-chordal graph \( G \). A set \( V_- \subset V(G) \) is a hole cover of \( G \) if and only if it contains \( K_A(\alpha) \) for some point \( \alpha \in A \).

**Proof.** For any vertex set \( V_- \), the subgraph \( G - V_- \) is also a proper Helly circular-arc graph, and the set of arcs \( A' := \{ A_v : v \in V(G) \setminus V_- \} \) is a proper and Helly arc model for \( G - V_- \). For the “if” direction, we may rotate \( A \) to make \( \alpha = 0 \), and then setting \( I_v = A_v \) for each \( v \in V(G) \setminus V_- \) gives a proper interval model for \( G - V_- \). For the “only if” direction, note that if there is no \( \alpha \) with \( K_A(\alpha) \subseteq V_- \), then we can find a minimal set \( X \) of vertices from \( V(G) \setminus V_- \) such that \( \bigcup_{v \in X} A_v \) covers the whole circle in \( A \). According to Proposition 2.6, \( X \) induces a hole of \( G \), which remains in \( G - V_- \).

It is easy to verify that in a fat \( W_5 \), it suffices to delete a clique from the fat hole with the minimum size. Therefore, Theorem 2.5 and Lemma 3.1 imply the following linear-time algorithm.

**Corollary 3.2.** The unit interval vertex deletion problem can be solved in \( O(m) \) time on \( F \)-free graphs.

We are now ready to prove the main results of this section.
Theorem 3.3. There are an \(O(6^k \cdot m)\)-time parameterized algorithm for the unit interval vertex deletion problem and an \(O(nm)\)-time approximation algorithm of ratio 6 for its minimization version.

Proof. Let \((G, k)\) be an instance of unit interval vertex deletion; we may assume that G is not a unit interval graph and \(k > 0\). The parameterized algorithm calls first Theorem 2.5(2) to decide whether it has an induced subgraph in \(\mathcal{F}\), and then based on the outcome, it solves the problem by making recursive calls to itself, or calling the algorithm of Corollary 3.2. If an induced subgraph \(F\) in \(\mathcal{F}\) is found, it calls itself \(|V(F)|\) times, each with a new instance \((G - v, k - 1)\) for some \(v \in V(F)\); since we need to delete at least one vertex from \(V(F)\), the original instance \((G, k)\) is a yes-instance if and only if at least one of the instances \((G - v, k - 1)\) is a yes-instance. Otherwise, \(G\) is \(\mathcal{F}\)-free and the algorithm calls Corollary 3.2 to solve it. The correctness of the algorithm follows from discussion above and Corollary 3.2. On each subgraph in \(\mathcal{F}\), which has at most 6 vertices, at most 6 recursive calls are made, all with parameter value \(k - 1\). By Theorem 2.5, each recursive call is made in \(O(m)\) time; each call of Corollary 3.2 takes \(O(m)\) time. Therefore, the total running time is \(O(6^k \cdot m)\).

The approximation algorithm is adapted from the parameterized algorithm as follows. For the subgraph \(F\) found by Theorem 2.5, we delete all its vertices. We continue the process until the remaining graph is \(\mathcal{F}\)-free, and then we call Corollary 3.2 to solve it optimally. Each subgraph in \(\mathcal{F}\) has 4 or 6 vertices, and thus at most \(n/4\) such subgraphs can be detected and deleted, each taking \(O(m)\) time, hence \(O(nm)\) in total. Corollary 3.2 takes another \(O(m)\) time. The ratio is clearly 6, and the total running time is \(O(nm) + O(m) = O(nm)\).

4 Edge deletion

Inspired by Lemma 3.1, one may expect a similarly nice characterization for a minimal set of edges whose deletion from a proper Helly circular-arc graph \(G\) makes it chordal, e.g., it is “local” to some point in an arc model for \(G\). This is nevertheless not the case; as shown in Fig. 5, they may behave in a very strange or pathological way.

Recall that \(v \rightarrow u\) means that arc \(A_v\) intersects arc \(A_u\) from the left when \(cp(v) \in A_u\). For each point \(\alpha\) in an arc model \(A\), we can define the following set of edges:

\[
\overrightarrow{E}_A(\alpha) = \{vu : v \in K_A(\alpha), u \notin K_A(\alpha), v \rightarrow u\}. \tag{1}
\]

It can be symmetrically defined as \([vu : v \notin K_A(\beta), u \in K_A(\beta), v \rightarrow u]\), where \(\beta := \max\{cp(x) : x \in K_A(\alpha)\} + \epsilon\). It is easy to verify that the following gives a proper interval model for \(G - \overrightarrow{E}_A(0)\):

\[
I_v := \begin{cases} 
[ccp(v), cpc(v) + \ell] & \text{if } v \in K_A(0), \\
[ccp(v), cpc(v)] & \text{otherwise},
\end{cases}
\]

where \(\ell\) is the perimeter of the circle in \(A\); see Fig. 6. For an arbitrary point \(\alpha\), the model \(G - \overrightarrow{E}_A(\alpha)\) can be given analogously, e.g., we may rotate the model first to make \(\alpha = 0\).

Proposition 4.1. Let \(A\) be a proper and Helly arc model for a non-chordal graph \(G\). For any point \(\alpha\) in \(A\), the subgraph \(G - \overrightarrow{E}_A(\alpha)\) is a unit interval graph.

The other direction is more involved and is more challenging. For two disjoint sets \(X, Y\) of vertices, let \(E_G(X, Y)\) denote the set of edges of \(G\) that has one end in \(X\) and the other end in \(Y\), i.e., \((X \times Y) \cap E(G)\).
A unit interval graph $G$ is called a **spanning unit interval subgraph** of $G$ if $V(G) = V(G)$ and $E(G) \subseteq E(G)$; it is called **maximum** if it has the largest number of edges among all spanning unit interval subgraphs of $G$. To prove all maximum spanning unit interval subgraphs have a certain property, we use the following argument by contradiction. Given a spanning unit interval subgraph $G$ not having the property, we locally modify a unit interval model $I'$ for $G$ to a **proper** interval model $I'$ such that the represented graph $G'$ satisfies $E(G') \subseteq E(G)$ and $|E(G')| > |E(G)|$. Recall that by the selection of $\epsilon$, an arc covering $\rho \pm \epsilon$ must contain $\rho$.

**Lemma 4.2.** Let $A$ be a proper and Helly arc model for a non-chordal graph $G$. For any maximum spanning unit interval subgraph $G'$ of $G$, the deleted edges $E(G) \setminus E(G')$ are $\mathcal{P}_{A}(\rho)$ for some point $\rho$ in $A$.

**Proof.** We fix a unit interval model $I$ for $G$. Let $v$ be the vertex with the leftmost interval $[0,1]$ in $I$. All arcs in the proof are referred to the model $A$ for $G$. Denote by $u$ and $w$ the vertices of $G[v]$ that have the leftmost and the rightmost arcs respectively; possibly $u = v$ and/or $w = v$. Note that $N_{G}[v] = K_{2}(1)$, which is a clique; hence, if $u \neq w$, then $uv \in E(G)$ and, in particular, $u \rightarrow w$. Let $\alpha := \text{ccp}(u) - \epsilon$ and $\beta := \text{ccp}(w) + \epsilon$; for notational convenience, we may assume that $[\alpha, \beta]$ does not cover the point 0 (the union of $A_{u}, A_{v}$, and $A_{w}$ does not cover the circle). Note that an arc covering $\alpha$ or $\beta$ has to intersect $A_{u}$ or $A_{w}$ respectively. Thus, since the model is proper and by Proposition 2.6, no arc contains both $\alpha$ and $\beta$. On the other hand, the arc $A_{x}$ for any $x \in N_{G}[v]$ is in $(\alpha, \beta)$. Therefore, $K_{A}(\alpha), K_{A}(\beta)$, and $N_{G}[v]$ are pairwise disjoint.

We argue first that $K_{A}(\alpha)$ and $K_{A}(\beta)$ cannot be both adjacent to $N_{G}[v]$ in $G$. This holds vacuously if $N_{G}[v]$ is a single component of $G$. Hence we assume otherwise: let $I_{x}$ be the first interval with $1p(x) > rp(v)$, then $1p(x) < 2$ and $N_{G}(x)$ intersects $N_{G}[v]$. Note that $x \not\in N_{G}[v]$, as otherwise setting $I_{x}$ to $[1p(x) + \epsilon - 1, 1p(x) + \epsilon]$ gives a spanning unit interval subgraph of $G$ with edges $E(G) \cup \{vx\}$, a contradiction. Hence, $x$ is in either $K_{A}(\alpha)$ or $K_{A}(\beta)$. Assume first that $x \in K_{A}(\alpha) \setminus N_{G}[v]$ and we show that $K_{A}(\beta)$ is not adjacent to $N_{G}[v]$ in $G$. Suppose for contradiction, that there is some vertex $y \in K_{A}(\beta)$ such that $I_{y}$ intersects $I_{z}$ for $z \in N_{G}[v]$. Then $1p(y) < rp(z) < 2 < 1p(x)$, which means that $I_{y}$ intersects $I_{x}$ as well (noting $1p(x) < 1p(y)$). As a result, $xy \in E(G) \subseteq E(G)$, and $A_{y}$ intersects $A_{z}$; by the selection of $\alpha$ and $\beta$, we must have $y \rightarrow x$, but then $A_{x}, A_{y},$ and $A_{z}$ do not satisfy the Helly property, a contradiction. A symmetric argument implies that $K_{A}(\alpha)$ is nonadjacent to $N_{G}[v]$ in $G$ when $x \in K_{A}(\beta) \setminus N_{G}[v]$.

Assume without loss of generality that $K_{A}(\alpha)$ is not adjacent to $N_{G}[v]$ in $G$. Let $\mu := \text{ccp}(u)$; note that $\mu \in A_{v}$ and $K_{A}(\mu) \subseteq N_{G}[v]$. Since the model $A$ is proper and Helly, no arc in $A$ can contain both $\alpha$ and $\mu$. Therefore, $K_{A}(\alpha)$ and $K_{A}(\mu)$ are disjoint, and from the definition of $\alpha$, we can conclude that

$$\mathcal{P}_{A}(\alpha) = E(G)(K_{A}(\alpha), K_{A}(\mu))$$

Let $E_{\pm} := E(G) \setminus E(G')$; by Proposition 4.1, $|E_{\pm}| \leq |E_{A}(\alpha)|$. We argue that they have to be equal. Suppose for contradiction, $E_{\pm} \neq E_{A}(\alpha)$, then $E_{A}(\alpha) \not\subseteq E_{\pm}$. There must be some vertices in $K_{A}(\mu)$ that are adjacent to $K_{A}(\alpha)$ in $G$; by assumption (that $K_{A}(\alpha)$ is not adjacent to $N_{G}[v]$ in $G$), these vertices are not in $N_{G}[v]$. Let $X := K_{A}(\mu) \setminus N_{G}[v]$. We take a vertex $x \in X$ such that $N_{G}(x) \cap K_{A}(\alpha)$ has the largest cardinality, which is positive. Recall that $u \in N_{G}[v]$; hence $x \neq u$ and $u \rightarrow x$. We may assume that $1p(x)$ is contained in some interval for a vertex in $K_{A}(\alpha)$, and the other case follows by symmetry. Note that $N_{G}(x) \setminus N_{G}[u]$ is disjoint from $K_{A}(\alpha)$, and by the Helly property, it cannot be adjacent to $N_{G}(x) \cap K_{A}(\alpha)$. Therefore, for any $y \in N_{G}(x) \setminus N_{G}[u] \subseteq N_{G}(x) \setminus N_{G}[u]$, the interval $I_{y}$ has to contain $rp(x)$; in other
words, \( lp(y) \in I_x \). Let

\[
\gamma := \begin{cases} 
    rp(x) & \text{if } N_G(x) \cap N_G(u) = \emptyset, \\
    \min_{y \in N_G(x) \setminus N_G(u)} lp(y) & \text{otherwise.}
\end{cases}
\]

Setting \( I'_u = [\min_{y \in I_x} lp(y) - \epsilon, \gamma - \epsilon] \) gives also a proper interval model \( J' \). To see that \( J' \) represents a subgraph of \( G \), note that \( N_G(x) \cap K_A(\alpha) \subseteq N_G(u) \). Since \( G \) is a maximum spanning unit interval subgraph of \( G \), it follows that \( |N_G(x) \cap K_A(\alpha)| \leq |N_G(u)| \). Likewise, since \( N_G(u) \subseteq N_G[v] \), it follows that \( N_G[v] = N_G[u] \) (otherwise we can set \( I'_u = [lp(u) - \epsilon, rp(u) - \epsilon] \) to get a larger spanning unit interval subgraph of \( G \)). Therefore, for every \( x' \in X \), it holds that

\[
|N_G(x') \cap K_A(\alpha)| \leq |N_G(x) \cap K_A(\alpha)| \leq |N_G(u)| < |N_G[u]| = |N_G[v]|
\]

The first inequality is ensured by the selection of \( x \). However, noting that \( N_G[v] \subseteq N_G(x') \) for every \( x' \in X \), it can be inferred

\[
|E_{-}\| = |E_G(K_A(\alpha), N_G[v])| + |E_G(X, N_G[v])| + |E_G(K_A(\alpha), X)| - |E_G(K_A(\alpha), X)| = |E_G(K_A(\alpha), N_G[v])| + |E_G(K_A(\alpha), X)| + \sum_{x' \in \mathcal{E}} (|N_G[v]| - |N_G(x') \cap K_A(\alpha)|)
\]

which contradicts Proposition 4.1. Thus, \( E_- = \overrightarrow{E}(\alpha) \), and this concludes the proof.

It is worth stressing that a thinnest place in an arc model with respect to edges is not necessarily a thinnest place with respect to vertices; see, e.g., Fig. 7. There is a linear number of different places to check, and thus the edge deletion problem can also be solved in linear time on proper Helly circular-arc graphs. The problem is also simple on fat \( W_5 \)'s.

![Figure 7: The thinnest points for vertex and edge are \( \alpha \) and \( \beta \) respectively: \( K_A(\alpha) = 2 < K_A(\beta) = 3 \); while \( \overrightarrow{E}(\alpha) = 8 > \overrightarrow{E}(\beta) = 6 \).](image)

**Theorem 4.3.** The unit interval edge deletion problem can be solved in \( O(m) \) time (1) on proper Helly circular-arc graphs and (2) on \( F \)-free graphs.

**Proof.** We may assume that the input graph \( G \) is connected, as otherwise we work on its components one by one.

Consider first (1). We build a proper and Helly arc model \( A \) for \( G \); without loss of generality, assume that it is canonical. According to Lemma 4.2, the problem reduces to finding a point \( \alpha \) in \( A \) such that \( \overrightarrow{E}_A(\alpha) \) is minimized. It suffices to consider the 2n points \( i + 0.5 \) for \( i \in \{0, \ldots, 2n - 1\} \). We calculate first \( \overrightarrow{E}_A(0.5) \), and then for \( i = 1, \ldots, 2n - 1 \), we deduce \( \overrightarrow{E}_A(i + 0.5) \) from \( \overrightarrow{E}_A(i - 0.5) \) as follows. If \( i \) is a clockwise endpoint of some arc, then \( \overrightarrow{E}_A(i + 0.5) = \overrightarrow{E}_A(i - 0.5) \). Otherwise, \( i = ccp(v) \) for some vertex \( v \), then the difference between \( \overrightarrow{E}_A(i + 0.5) \) and \( \overrightarrow{E}_A(i - 0.5) \) is the set of edges incident to \( v \). In particular, \( \{uv : u \to v\} = \overrightarrow{E}_A(i - 0.5) \setminus \overrightarrow{E}_A(i + 0.5), \) while \( \{uv : v \to u\} = \overrightarrow{E}_A(i + 0.5) \setminus \overrightarrow{E}_A(i - 0.5) \). Note that the initial value \( \overrightarrow{E}_A(0.5) \) can be calculated in \( O(m) \) time, and then each vertex and its adjacency list is scanned exactly once. It follows that the total running time is \( O(m) \). 

11
We remark that it is necessary to impose the quotas for different modifications in the stated, though by definition, \(N_G[u] = N_G[v]\). We argue that \(N_G[u] = N_G[v]\) for any maximum spanning unit interval subgraph \(G\) of \(G\). Suppose the contrary, then setting \(l_u\) to \(l_v\) or \(l_{\text{cr}}\) to \(l_u\) will end with a spanning unit interval subgraph of \(G\) with strictly more edges than \(G\). Therefore, we need to delete \(E_G(K_i, K_i+1)\) as well as \(E_G(K_5, K_5)\) or \(E_G(K_5, K_5+1)\) for some \(i \in \{0, \ldots, 4\}\) (all subscripts are modulo 5). Once the sizes of all six cliques have been calculated, which can be done in \(O(m)\) time, the minimum set of edges can be decided in constant time. Therefore, the total running time is \(O(m)\). The proof is now complete.

Theorems 4.3 and 2.5 already imply a bound-search tree algorithm for the unit interval edge deletion problem running in time \(O(9^k \cdot m)\). Here the constant 9 is decided by the tent, which has 9 edges. However, a closer look at it tells us that deleting one edge from a tent introduces a claw or \(C_4\), which forces us to delete some other edge(s). The disposal of a net is similar. This observation and a refined analysis will yield the running time claimed in Theorem 1.1. The labels for a net and a tent are as given in Fig. 1.

**Theorem 4.4.** There is an \(O(4^k \cdot m)\)-time parameterized algorithm for the unit interval edge deletion problem.

**Proof.** The algorithm goes similarly as the parameterized algorithm for unit interval vertex deletion used in the proof of Theorem 3.3: it disposes of subgraphs in \(T\) by calling itself, and then calls the algorithm of Theorem 4.3 to solve the remaining instance on an \(T\)-free graph. When a claw or \(C_4\) is found, the algorithm makes 3 or 4 calls to itself, each with a new instance parameter value \(k - 1\) (deleting one edge from the claw or \(C_4\)). For a net, the algorithm makes 6 calls to itself, of which 3 with parameter value \(k - 1\) (deleting edge \(v_i, t_i\) for \(i = 1, 2, 3\)), and another 3 with parameter value \(k - 2\) (deleting two edges from the triangle \(v_1v_2v_3\)). For a tent, the algorithm makes 9 calls to itself, all with parameter value \(k - 2\); in particular, it deletes two edges from a triangle involving a vertex \(t_i\) with \(i = 1, 2, 3\).

We verify first the correctness of the algorithm, for which we show that for any spanning unit interval subgraph \(G\) of \(G\), there is at least one recursive call whose graph \(G'\) satisfies \(E(G) \subseteq E(G') \subseteq E(G)\). This is obvious when the recursive calls are made on a claws or \(C_4\). For a net, if none of the three edges \(v_1t_1, v_2t_2, \) and \(v_3t_3\) is deleted, then at least two edges from the triangle \(v_1v_2v_3\) have to be deleted (deleting only one leaves a claw). For a tent, we consider the intervals for \(v_1, v_2, \) and \(v_3\) in a unit interval model \(T\) for \(G\). Assume first that \(v_1v_2v_3\) remains a triangle of \(G\). Then the length of \([lp(v_1), rp(v_3)]\) is shorter than 2, and for at least one \(i \in \{1, 2, 3\}\) the interval for \(t_i\) is disjoint from it. In other words, we need to delete both edges incident to \(t_i\). Assume then \(v_1v_2v_3\) is not a triangle of \(G\), and without loss of generality \(lp(v_1) < lp(v_2) < lp(v_3)\). Then \(v_1v_3 \notin E(G)\) and \(t_2\) cannot be adjacent to both \(v_1\) and \(v_3\). Therefore, we need to delete two edges from the triangle \(v_2v_3\). Every of the possible cases is considered in some recursive call(s). This ensures the correctness of the bounded search tree procedure, and the algorithm.

With standard technique, it is easy to verify that \(O(4^k)\) recursive calls are made, each in \(O(m)\) time. Moreover, the algorithm for Theorem 4.3 is called \(O(4^k)\) times. It follows that the total running time of the algorithm is \(O(4^k \cdot m)\).

What dominates the branching step is the disposal of \(C_4\)'s. With the technique the author developed in [18], we may (slightly) improve the runtime to \(O(c^k \cdot m)\) for some constant \(c < 4\). To avoid blurring the focus, we omit the details.

### 5 General editing

Let \(V_- \subseteq V(G)\), and let \(E_-\) and \(E_+\) be a set of edges and a set of non-edges of \(G - V_-\) respectively. We say that \((V_-, E_-, E_+)\) is an editing set of \(G\) if the deletion of \(E_-\) from and the addition of \(E_+\) to \(G - V_-\) create a unit interval graph. Its size is defined to be the 3-tuple \((|V_-|, |E_-|, |E_+|)\), and we say that it is smaller than \((k_1, k_2, k_3)\) if all of \(|V_-| \leq k_1\) and \(|E_-| \leq k_2\) and \(|E_+| \leq k_3\) hold true and at least one inequality is strict. The unit interval editing problem is formally defined as follows.

**Input:** A graph \(G\) and three nonnegative integers \(k_1, k_2,\) and \(k_3\).

**Task:** Either construct an editing set \((V_-, E_-, E_+)\) of \(G\) that has size at most \((k_1, k_2, k_3)\), or report that no such set exists.

We remark that it is necessary to impose the quotas for different modifications in the stated, though
cumbersome, way. Since vertex deletions are clearly preferable to both edge operations, the problem would be computationally equivalent to unit interval vertex deletion if we have a single budget on the total number of operations.

By and large, our algorithm for the unit interval editing also uses the same two-phase approach as the previous algorithms. The main discrepancy lies in the first phase, when we are not satisfied with an $F$-free graph; in particular, we also want to do away with all holes of length at most $k_3 + 3$, which are precisely those holes fixable by merely adding edges (recall that at least $\ell - 3$ edges are needed to fill a $C_4$ in). In the very special cases where $k_3 = 0$ or 1, a fat $W_5$ satisfies these conditions. It is not hard to solve these cases, but to make the rest more focused and also simplify the presentation, we also exclude these cases by disposing of all $C_5$’s in the first phase.

A graph is called reduced if it contains no claw, net, tent, $C_4$, $C_5$, or $C_7$ with $\ell \leq k_3 + 3$. A reduced graph $G$ is a proper Helly circular-arc graph, and hence if it happens to be chordal, then it must be a unit interval graph, and we terminate the algorithm. Otherwise, our algorithm enters the second phase. Now that $G$ is reduced, every minimal forbidden induced subgraph is a hole $C_\ell$ with $\ell > k_3 + 3$, which can only be fixed by deleting vertices and/or edges. Here we again exploit a proper and Helly arc model $A$ for $G$. According to Lemma 3.1, if there exists some point $p$ in the model such that $|K_A(p)| \leq k_1$, then it suffices to delete all vertices in $K_A(p)$, which results in a subgraph that is a unit interval graph. Therefore, we may assume hereafter that no such point exists, then $G$ remains reduced and non-chordal after at most $k_1$ vertex deletions. As a result, we have to delete edges as well.

Consider an (inclusion-wise minimal) editing set $(V_-, E_-, E_\ell)$ to a reduced graph $G$. It is easy to verify that $(\emptyset, E_-, E_\ell)$ is an (inclusion-wise minimal) editing set of the reduced graph $G - V_-$. In particular, $E_-$ intersects all holes of $G - V_-$. We use $A - V_-$ as a shorthand for $\{A_v : v \notin V_\ell\}$. One may want to use Lemma 4.2 to find a minimum set $E_-$ of edges (i.e., $E_{A-V_-}(\alpha)$ for some point $\alpha$) to finish the task. However, Lemma 4.2 has not ruled out the possibility that we delete less edges to break all long holes, and subsequently add edges to fix the incurred subgraphs in $\{\text{claw, net, tent, } C_4, C_5, C_7\}$ with $\ell \leq k_3 + 3$. So we need the following lemma.

**Lemma 5.1.** Let $(V_-, E_-, E_\ell)$ be an inclusion-wise minimal editing set of a reduced graph $G$. If $|E_\ell| \leq k_3$, then $E_- = \emptyset$.

**Proof.** We may assume without loss of generality $V_- = \emptyset$, as otherwise it suffices to consider the inclusion-wise minimal editing set $(\emptyset, E_-, E_\ell)$ to the still reduced graph $G - V_-$. Let $A$ be a proper and Helly arc model for $G$. Let $E_-'$ be an inclusion-wise minimal subset of $E_-$ such that for every hole in $G - E_-'$, the union of arcs for its vertices does not cover the circle of $A$. Note that $E_-'$ exists because $E_-$ itself satisfies this condition: suppose that there exists in $G - E_-$ a hole whose arcs cover the circle, then it has at least $k_3 + 4$ vertices (Proposition 2.6) and cannot be fixed by the addition of $E_\ell$. We argue that $G := G - E_-'$ is already a unit interval graph. It follows that $E_- = E_-'$ and $E_\ell = \emptyset$.

Suppose for contradiction, there is $X \subseteq V(G)$ inducing a claw, net, tent, or a hole in $G[X]$. We find three vertices $u, v, w \in X$ such that $uw \in E_-'$ and $uv, wv \in E(G)$ as follows. Note that $\bigcup_{v \in X} A_v$ cannot cover the whole circle: by assumption, this is true when $G$ is a hole; on the other hand, by Lemma 2.6, and noting that $G$ is $\{C_4, C_5\}$-free, at least 6 arcs are needed to cover the circle (Proposition 2.6), but a claw, net, or tent has at most 6 vertices, and cannot be a subgraph of a $C_6$. Thus, $G[X]$ is a unit interval graph. So we can find two vertices $x, z$ from $X$ having $xz \in E_-'$. We find a shortest $x-z$ path in $G[X]$. If the path has more than one inner vertex, then it makes a hole together with $xz$, which means that there exists an inner vertex $y$ of this path such that $xy \in E_-'$ or $yz \in E_-'$. We consider then the new pair $x, y$ or $y, z$ accordingly. Note that their distance in $G[X]$ is smaller than $xz$, and hence repeating this argument (at most $|X| - 3$ times) will end with two vertices with distance precisely 2 in $G[X]$. They are the desired $u$ and $w$, while any common neighbor of them in $G[X]$ can be $v$. By the minimality of $E_-'$, in $G + uw$ there exists a hole $H$ such that $H$ cannot penetrate the circle in $A$. This hole $H$ necessarily passes $uw$, and we denote it by $x_1 x_2 \cdots x_{\ell-1} x_{\ell}$, where $x_1 = u$ and $x_{\ell} = w$. Note that $Au$ intersects $Aw$, and since $A$ is proper and Helly, $A_u, A_v, A_w$ cannot cover the circle; moreover, $A_u$ cannot intersect the arc for all $x_i$ with $1 < i < \ell$. From $x_1 x_2 \cdots x_{\ell-1} x_{\ell}$ we can find $p$ and $q$ such that $1 \leq p < q \leq \ell$ and $v x_p, v x_q \in E(G)$ but $v x_i \notin E(G)$ for every $p < i < q$. Here possibly $p = 1$ and/or $q = \ell$. Then $v x_p \cdots x_q$ makes a hole of $G$, and the union of its arcs covers the circle, contradicting the definition of $E_-'. This concludes the proof.

Therefore, a yes-instance on a reduced graph always has a solution adding no edges. By Lemma 4.2, for any editing set $(V_-, E_-, \emptyset)$, we can always find some point $\alpha$ in the model and use $E_{A-V_-}(\alpha)$ to replace $E_-$. After that, we can use the vertices “close” to this point to replace $V_-$. Therefore, the problem
We now consider the original model \( A \) we notice that (1) the problem is also easy on fat and tents) and edge deletions (on graphs and an \( O(k) \) \( \alpha \) of at most \( m \) vertices, remains reduced and non-chordal. Hence, \( q > 0 \). For each point \( \rho \) in \( A \), we can define an editing set \( (V\_\alpha, E\_\alpha) \) by taking the \( p \) vertices in \( K\_\alpha \) with the rightmost arcs as \( V\_\alpha \) and \( E\_\alpha \) as \( E\_\alpha \). We argue first that the minimum cardinality of this edge set, taken among all points in \( A \) is the desired number \( q \).

Let \( (V\_\alpha, E\_\alpha, 0) \) be an editing set of \( G \) with size \((p, q, 0)\). According to Lemma 4.2, there is a point \( \alpha \) such that the deletion of \( E\_\alpha := E\_\alpha \setminus V\_\alpha \) from \( G \) makes it a unit interval graph and \( |E\_\alpha| \leq |E\_\alpha| \).

We now consider the original model \( A \). Note that a vertex in \( V\_\alpha \) is in either \( K\_\alpha \) or \( \{v \notin K\_\alpha : u \to v, u \in K\_\alpha\} \); otherwise replacing this vertex by any end of an edge in \( E\_\alpha \), and removing this edge from \( E\_\alpha \) gives an editing set of size \((p, q - 1, 0)\). Let \( V\_\alpha \) comprise the \( |V\_\alpha \cap K\_\alpha| \) vertices of \( K\_\alpha \) whose arcs are the rightmost in \( A \), as well as the first \( |V\_\alpha \setminus K\_\alpha| \) vertices whose arcs are to the right of \( \alpha \).

And let \( E\_\alpha := E\_\alpha \setminus V\_\alpha \). It is easy to verify that \( |E\_\alpha| \leq |E\_\alpha| = q \) and \( (V\_\alpha, E\_\alpha, 0) \) is also an editing set of \( G \) (Lemma 4.1). Note that arcs for \( V\_\alpha \) are consecutive in \( A \). Let \( v \) be the vertex in \( V\_\alpha \) with the rightmost arc, and then \( ccp(v) - \epsilon \) is the desired point \( \rho \).

We give now the \( O(m) \)-time algorithm for finding the desired point, for which we assume that \( A \) is canonical. It suffices to consider the \( 2n \) points \( i + 0.5 \) for \( i \in \{0, \ldots, 2n - 1\} \). We calculate first the \( V\_\alpha \) and \( E\_\alpha \) for \( 0.5 \), and maintain a queue of \( p \) elements, which are the vertices corresponding to the \( p \) rightmost arcs containing \( 0.5 \). For \( i = 1, \ldots, 2n - 1 \), we deduce the new sets for \( i + 0.5 \) from the previous point as follows. If \( i \) is a clockwise endpoint of some arc, both of them do not change. Otherwise, \( i = ccp(v) \) for some vertex \( v \), then we enqueue \( v \), and dequeue \( u \), and the difference between \( E\_\alpha \) and \( E\_\alpha \setminus V\_\alpha \) is the number of edges incident to \( u \). In particular, \( \{x : x \to u\} = E\_\alpha \setminus E\_\alpha \) is the initial value \( E\_\alpha \) can be found in \( O(m) \) time, and then each vertex and its adjacency is scanned exactly once. The total running time is \( O(m) \). This concludes the proof.

The combinatorial characterization on mixed hole covers consisting of both vertices and edges, thereby extending Lemmas 3.1 and 4.2. The algorithm is similar as the algorithm used for Theorem 4.3. Lem 5.1 and 5.2 have the following consequence: it suffices to call the algorithm with \( p = k_1 \), and returns the found editing set if \( q \leq k_2 \), or “NO” otherwise.

**Corollary 5.3.** The unit interval editing problem can be solved in \( O(m) \) time on reduced graphs.

Putting together these steps, the fixed-parameter tractability of unit interval editing follows. Note that to fill a hole, we need to add an edge whose ends have distance 2.

**Proof of Theorem 1.4.** We start by calling Theorem 2.5. If a subgraph in \( F \) or \( W_k \) is detected, then we branch on all possible ways of destroying it or the contained \( C_k \). Otherwise, we have in our disposal a proper and Helly arc model for \( G \), and we call Lemma 2.7 to find a shortest hole \( C_k \). If \( \ell \leq k_3 + 3 \), then we either delete one of its \( \ell \) vertices and \( \ell \) edges, or add one of \( \ell \) edges \( h_1, h_{t+2} \) (the subscripts are modulo \( \ell \)). We repeat these two steps until the graph is reduced, and then call the algorithm of Corollary 5.3 to solve it. The correctness of this algorithm follows from Lemmas 2.7 and Corollary 5.3. In the disposal of a subgraph of \( F \), at most 24 recursive calls are made, while 3\( \ell \) for \( C_k \), each having a parameter \( k \). Therefore, the total number of instances (with reduced graphs) made in the algorithm is \( O(k_3 + 1)^k \). It follows that the total running time of the algorithm is \( O(k_3 + 1)^k \cdot m \).

It is worth mentioning that Lemma 5.2 actually implies a linear-time algorithm for the unit interval deletion problem (which allows \( k_1 \) vertex deletions and \( k_2 \) edge deletions) on the proper Helly circular-arc graphs and an \( O(10^{k_1 + k_2}) \)-time algorithm for it on general graphs. The constant 10 can be even smaller if we notice that (1) the problem is also easy on fat \( W_k \)'s, and (2) the worst cases for vertex deletions (nets and tents) and edge deletions (\( C_k \)'s) are different.
6 Concluding remarks

All aforementioned algorithms exploit the characterization of unit interval graphs by forbidden induced subgraphs [27]. Very recently, Bliznets et al. [1] used a different approach to produce a subexponential-time parameterized algorithm for unit interval completion (whose polynomial factor is however very high). Using the parameter-preserved reduction from vertex cover [16], one can show that the vertex deletion version cannot be solved in $2^o(k) \cdot n^{O(1)}$ time, unless the Exponent Time Hypothesis fails [5]. Now that the edge deletion version is FPT as well, we would naturally ask to which side it belongs. The evidence we now have is in favor of the hard side: in all related graph classes, the edge deletion versions seem to be harder than their vertex deletion counterpart. As said, it is not hard to slightly improve the constant $c$ in the running time $O(c^k \cdot m)$, but a significant improvement would need some new observation. More interesting is to fathom their limits. In particular, can the deletion problems be solved in time $O(2^k \cdot m)$? We point out that although we start by breaking small forbidden induced subgraphs, our major proof technique is instead manipulating (proper/unit) interval models. The technique of combining (constructive) interval models and (destructive) forbidden induced subgraphs is worth further study on related problems.

We leave it as an open problem the existence of polynomial kernels of the unit interval edge deletion problem. Recall that polynomial kernels were known for unit interval completion [15, 20] and unit interval vertex deletion [11]. In contrast to the $O(k^2)$-vertex kernel for unit interval completion, the kernel size $O(k^{53})$ for unit interval vertex deletion is way too large, so further efforts are needed to make it reasonably small.

The algorithm for unit interval editing is the second nontrivial FPT algorithm for the general editing problem. The main ingredient of our algorithm is the characterization of the mixed deletion of vertices and edges to break holes. A similar study has been conducted in the algorithm for the chordal editing problem [8]. In contrast to that, Lemmas 5.1 and 5.2 are nicer. For example, we have shown that once small forbidden subgraphs have been all fixed, no edge additions are further needed. We conjecture this is also true for the chordal editing problem, but we failed to find a proof. Both algorithms for unit interval editing and chordal editing suffer from high running time, and the main culprits are exactly the mixed deletion of vertices and edges, on which very little study had been done. We hope that our work will trigger more studies on this direction, which will further deepen our understanding of various graph classes.

References

[1] Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michal Pilipczuk. A subexponential parameterized algorithm for proper interval completion. In Andreas S. Schulz and Dorothea Wagner, editors, Algorithms - ESA 2014, volume 8737 of LNCS, pages 173–184. Springer, 2014. Full version available at arXiv:1402.3472. doi:10.1007/978-3-662-44777-2_15.

[2] Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. On intervalizing k-colored graphs for DNA physical mapping. Discrete Applied Mathematics, 71(1-3):55–77, 1996. A preliminary version appeared in ICALP 1995. doi:10.1016/S0166-218X(96)00057-1.

[3] Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. NP-completeness results for edge modification problems. Discrete Applied Mathematics, 154(13):1824–1844, 2006. doi:10.1016/j.dam.2006.03.031.

[4] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. Information Processing Letters, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.

[5] Liming Cai and David W. Juedes. On the existence of subexponential parameterized algorithms. Journal of Computer and System Sciences, 67(4):789–807, 2003. doi:10.1016/S0022-0000(03)00074-6.

[6] Yixin Cao. Direct and certifying recognition of normal Helly circular-arc graphs in linear time. In Jianer Chen, John Hopcroft, and Jianxin Wang, editors, Frontiers in Algorithmics - FAW 2014, volume 8497 of LNCS, pages 13–24. Springer, 2014. Full version available at arXiv:1405.0329. doi:10.1007/978-3-319-08016-1_2.

[7] Yixin Cao. Linear recognition of almost interval graphs. arXiv:1403.1515, 2014.

[8] 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), volume 25 of LIPIcs, pages 214–225. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. Full version available at arXiv:1405.7859. doi:10.4230/LIPIcs.STACS.2014.214.

[9] 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. doi:10.1137/S0097539792269095.
[10] Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Undergraduate texts in computer science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

[11] Fedor V. Fomin, Saket Saurabh, and Yngve Villanger. A polynomial kernel for proper interval vertex deletion. SIAM Journal on Discrete Mathematics, 27(4):1964–1976, 2013. A preliminary version appeared in ESA 2012. doi:10.1137/12089051X.

[12] Delbert R. Fulkerson and Oliver A. Gross. Incidence matrices and interval graphs. Pacific Journal of Mathematics, 15(3):835–855, 1965. doi:10.2140/pjm.1965.15.835.

[13] F˘anic˘a Gavril. Algorithms on circular-arc graphs. Networks, 4:357–369, 1974. doi:10.1002/net.3230040407.

[14] 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. doi:10.1089/cmb.1995.2.139.

[15] Haim Kaplan, Ron Shamir, and Robert Endre 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. doi:10.1137/S0097539796303044.

[16] 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. Preliminary versions independently presented in STOC 1978. doi:10.1016/0022-0000(80)90060-4.

[17] Min Chih Lin, Francisco J. Soulignac, and Jayme L. Szwarcfiter. Normal Helly circular-arc graphs and its subclasses. Discrete Applied Mathematics, 161(7-8):1037–1059, 2013. doi:10.1016/j.dam.2012.11.005.

[18] Yunlong Liu, Jianxin Wang, Jie You, Jianer Chen, and Yixin Cao. Edge deletion problems: Branching facilitated by modular decomposition. Theoretical Computer Science, 573:63–70, 2015. doi:10.1016/j.tcs.2015.01.049.

[19] D´aniel Marx. Chordal deletion is fixed-parameter tractable. Algorithmica, 57(4):747–768, 2010. doi:10.1007/s00453-008-9233-8.

[20] Assaf Natanzon, Ron Shamir, and Roded Sharan. A polynomial approximation algorithm for the minimum fill-in problem. SIAM Journal on Computing, 30(4):1067–1079, 2000. A preliminary version appeared in STOC 1998. doi:10.1137/S0097539798336073.

[21] Fred S. Roberts. Indifference graphs. In Frank Harary, editor, Proof Techniques in Graph Theory (Proc. Second Ann Arbor Graph Theory Conf., 1968), pages 139–146. Academic Press, New York, 1969.

[22] Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In Automata, Languages and Programming (ICALP), volume 5125 of LNCS, pages 634–645, Berlin Heidelberg, 2008. Springer-Verlag. doi:10.1007/978-3-540-70575-8_52.

[23] Alan C. Tucker. Structure theorems for some circular-arc graphs. Discrete Mathematics, 7(1-2):167–195, 1974. doi:10.1016/S0012-365X(74)80027-0.

[24] Ren´e van Bevern, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Measuring indifference: Unit interval vertex deletion. In Dimitrios M. Thilikos, editor, Graph Theoretic Concepts in Computer Science - WG 2010, volume 6410 of LNCS, pages 232–243, 2010. doi:10.1007/978-3-642-16926-7_22.

[25] Pim van ‘t Hof and Yngve Villanger. Proper interval vertex deletion. Algorithmica, 65(4):845–867, 2013. doi:10.1007/s00453-012-9661-3.

[26] Yngve Villanger. Proper interval vertex deletion. In Venkatesh Raman and Saket Saurabh, editors, Parameterized and Exact Computation - IPEC 2010, volume 6478 of LNCS, pages 228–238. Springer, 2010. doi:10.1007/978-3-642-17493-3_22.

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

[28] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. SIAM Journal on Algebraic and Discrete Methods, 2(1):77–79, 1981. doi:10.1137/0602010.