Fixed-Treewidth-Efficient Algorithms for Edge-Deletion to Interval Graph Classes

Toshiki Saitoh[1](✉), Ryo Yoshinaka[2], and Hans L. Bodlaender[3]

[1] Kyushu Institute of Technology, Kitakyushu, Japan
toshikis@ces.kyutech.ac.jp
[2] Tohoku University, Sendai, Japan
ryoshinaka@tohoku.ac.jp
[3] Utrecht University, Utrecht, The Netherlands
H.L.Bodlaender@uu.nl

Abstract. For a graph class $\mathcal{C}$, the $\mathcal{C}$-Edge-Deletion problem asks for a given graph G to delete the minimum number of edges from G in order to obtain a graph in $\mathcal{C}$. We study the $\mathcal{C}$-Edge-Deletion problem for $\mathcal{C}$ the class of interval graphs and other related graph classes. It follows from Courcelle's Theorem that these problems are fixed parameter tractable when parameterized by treewidth. In this paper, we present concrete FPT algorithms for these problems. By giving explicit algorithms and analyzing these in detail, we obtain algorithms that are significantly faster than the algorithms obtained by using Courcelle's theorem.

Keywords: Parameterized algorithms · Treewidth · Edge-Deletion · Interval graphs

1 Introduction

Intersection graphs are represented by geometric objects aligned in certain ways so that each object corresponds to a vertex and two objects intersect if and only if the corresponding vertices are adjacent. Intersection graphs are well-studied in the area of graph algorithms since there are many important applications and we can solve many NP-hard problems in general graphs in polynomial time on such graph classes. Interval graphs are intersection graphs which are represented by intervals on a line. Clique, Independent Set, and Coloring on interval graphs can be solved in linear time and interval graphs have many applications in bioinformatics, scheduling, and so on. See [3,14,26] for more details of interval graphs and other intersection graphs.

Graph modification problems on a graph class $\mathcal{C}$ are to find a graph in $\mathcal{C}$ by modifying a given graph in certain ways. $\mathcal{C}$-Vertex-Deletion, $\mathcal{C}$-Edge-Deletion, and $\mathcal{C}$-Completion are to find a graph in $\mathcal{C}$ by deleting vertices, deleting edges, and adding edges, respectively, with the minimum cost.

These problems can be seen as generalizations of many NP-hard problems. Clique is equivalent to Complete-Vertex-Deletion: we find a complete graph by deleting the smallest number of vertices. Modification problems on intersection graph classes also have many applications. For example, Interval-Vertex/Edge-Deletion problems have applications to DNA (physical) mapping [12,13,27]. Lewis and Yannakakis showed that $\mathcal{C}$-Vertex-Deletion is NP-complete for any nontrivial hereditary graph class [18]. A graph class $\mathcal{C}$ is hereditary if for any graph in $\mathcal{C}$, every induced subgraph of the graph is also in $\mathcal{C}$. Since the class of intersection graphs are hereditary, $\mathcal{C}$-Vertex Deletion is NP-complete for any nontrivial intersection graph class $\mathcal{C}$. The problems $\mathcal{C}$-Edge-Deletion are also NP-hard when $\mathcal{C}$ is the class of perfect, chordal, split, circular arc, chain [24], interval, proper interval [13], trivially perfect [25], threshold [21], permutation, weakly chordal, or circle graphs [4]. See the lists in [4,20].

Parameterized complexity is well-studied in the area of computer science. A problem with a parameter k is *fixed parameter tractable*, *FPT* for short, if there is an algorithm running in $f(k)n^c$ time where n is the size of input, f is a computable function and c is a constant. Such an algorithm is called an *FPT algorithm*. The *treewidth* $tw(G)$ of a graph G represents treelikeness and is one of the most important parameters in parameterized complexity concerning graph algorithms. For many NP-hard problems in general, there are tons of FPT algorithms with parameter $tw(G)$ by dynamic programming on tree decompositions. Finding the treewidth of an input graph is NP-hard and it is known that Chordal-Completion with minimizing the size of the smallest maximum clique is equivalent to the problem. There is an FPT algorithm for computing the treewidth of a graph by Bodlaender [2] which runs in $O(f(tw(G))(n+m))$ time where n and m are the numbers of vertices and edges of a given graph: i.e., the running time is linear in the size of input. Courcelle showed that every problem that can be expressed in monadic second order logic (MSO_2) has a linear time algorithm on graphs of bounded treewidth [9]. Some intersection graph classes, for example interval graphs, proper interval graphs, chordal graphs, and permutation graphs, can be represented by MSO_2 [8] and thus there are FPT algorithms for Edge-Deletion problems on such graph classes. However, the algorithms obtained by Courcelle's theorem have a very large hidden constant factor even when the treewidth is very small, since the running time is the exponential tower of the coding size of the MSO_2 expression.

Our Results: We propose concrete FPT algorithms for Edge-Deletion to interval graphs and other related graph classes, when parameterized by the treewidth of the input graph. Our algorithms virtually compute a set of edges S with the minimum size such that $G - S$ is in a graph class $\mathcal{C}$ by using dynamic programming on a tree-decomposition. We maintain possible alignments of geometric objects corresponding to vertices in the bag of each node of the tree-decomposition. Alignments of the objects of forgotten vertices are remembered only relatively to the objects of the current bag. If two forgotten objects have the same relative position to the objects of the current bag, we remember only the fact that there is at least one forgotten object at that position. In this way, we achieve the fixed-parameter-tractability, while guaranteeing that no object pairs of non-adjacent vertices of the input graph will intersect in our dynamic programming algorithm. Our algorithms run in $O(f(tw(G))\cdot(n+m))$ time where n and m are the numbers of vertices and edges of the input graph. Our explicit algorithms are significantly faster than those obtained by using Courcelle's theorem. We also analyze the time complexity of our algorithms parameterized by pathwidth which is analogous to treewidth. The relation among the graph classes for which this paper provides $\mathcal{C}$-Edge-Deletion algorithms is shown in Fig. 1.

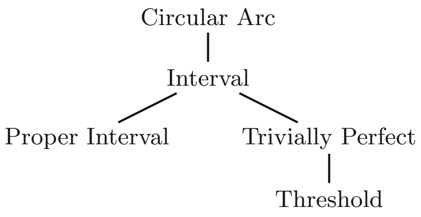


Fig. 1. The graph classes of which this paper presents algorithms for the edge-deletion problems.

Related Works: Another kind of common parameters considered in parameterized complexity of graph modification problems is the number of vertices or edges to be removed or to be added. Here we review preceding studies on those problems for intersection graphs with those parameters.

Concerning parameterized complexity of $\mathcal{C}$-Vertex-Deletion, Hof et al. proposed an FPT algorithm for Proper-Interval-Vertex-Deletion [16], and Marx proposed an FPT algorithm for Chordal-Vertex-Deletion [22]. Heggernes et al. showed Perfect-Vertex-Deletion and Weakly-Chordal-Vertex-Deletion are W[2]-hard [15]. Cai showed that C-Vertex/Edge-Deletion are FPT when $\mathcal{C}$ is characterized by a finite set of forbidden induced subgraphs [5].

For modification problems on interval graphs, Villanger et al. presented an FPT algorithm for Interval-Completion [28], and Cao and Marx presented an FPT algorithm for Interval-Vertex-Deletion [7]. Cao improved these algorithms and developed an FPT algorithm for Edge-Deletion [6].

It is known that Threshold-Edge-Deletion, Chain-Edge-Deletion and Trivially-Perfect-Edge-Deletion are FPT, since threshold graphs, chain graphs and trivially perfect graphs are characterized by a finite set of forbidden induced subgraphs [5]. Nastos and Gao presented faster algorithms for the problems [23], and Liu et al. improved their algorithms to $O(2.57^k(n+m))$ and $O(2.42^k(n+m))$ using modular decomposition trees [19], where k is the number of deleted edges. There are algorithms to find a polynomial kernel for Chain-Edge-Deletion and Trivially Perfect-Edge-Deletion [1,11].

Organization of this Article: Section 2 prepares the notation and definitions used in this paper. We propose an FPT algorithm for Interval-Edge-Deletion in Sect. 3. We then extend the algorithm related to the interval graphs in Sect. 4. We conclude this paper and provide some open questions in Sect. 5.

2 Preliminaries

For a set X, its cardinality is denoted by $|X|$. A *partition* of X is a tuple $(X_1, \ldots, X_k)$ of subsets of X such that $X = X_1 \cup \cdots \cup X_k$ and $X_i \cap X_j = \emptyset$ if $1 \leq i < j \leq k$, where we allow some of the subsets to be empty. For entities $x, y, z \in X$, we let $x[y/z] = y$ if $x = z$, and $x[y/z] = x$ otherwise. For a subset $Y \subseteq X$, define $Y[y/z] = \{\, x[y/z] \mid x \in Y \,\}$.

For a linear order π over a finite set X, by $\mathsf{suc}_\pi(x)$ we denote the successor of $x \in X$ w.r.t. π, i.e., $x <_\pi \mathsf{suc}_\pi(x)$ and $\mathsf{suc}_\pi(x) \leq_\pi y$ for all y with $x <_\pi y$. The maximum element of X w.r.t. π is denoted by $\max_\pi X$. Note that $\mathsf{suc}_\pi(\max_\pi X)$ is undefined. Similarly $\mathsf{pred}_\pi(x)$ is the predecessor of x and $\min_\pi X$ is the least element of X.

A simple graph $G = (V, E)$ is a pair of vertex and edge sets, where each element of E is a subset of V consisting of exactly two elements.

A *tree-decomposition* of $G = (V, E)$ is a tree T such that[1]

- to each node of T a subset of V is assigned,
- if the assigned sets of two nodes of T contain a vertex $u \in V$, then so does every node on the path between the two nodes,
- for each $\{u, v\} \in E$, there is a node of T whose assigned set includes both u and v.

The *width* of a tree-decomposition is the maximum cardinality of the assigned sets minus one and the *treewidth* of a graph is the smallest width of its tree-decompositions. A tree-decomposition is said to be *nice* if it is rooted, the root is assigned the empty set, and its nodes are grouped into the following four:

- *leaf nodes*, which have no children and are assigned the empty set,
- *introduce nodes*, each of which has just one child, where the set assigned to the parent has one more vertex than the child's set,
- *forget nodes*, each of which has just one child, where the set assigned to the parent has one less vertex than the child's set,
- *join nodes*, each of which has just two children, where the same vertex set is assigned to the parent and its two children.

It is known that every tree-decomposition has a nice tree-decomposition of the same width whose size is $O(k|V|)$, where k is the treewidth of the tree-decomposition [10,17]. Hereafter, under a fixed graph G and a fixed nice tree-decomposition T, we let X_s denote the subset of V assigned to a node s of a tree-decomposition and $X_{\leq s}$ denote the union of all the subsets assigned to the

[1] We use the terms "vertices" for an input graph and "nodes" for a tree-decomposition.

node s and its descendant nodes. We call vertices in X_s and in $X_{\leq s} - X_s$ *active* and *forgotten*, respectively. Moreover, we define $E_s = \{ \{u, v\} \in E \mid u, v \in X_s\}$ and $E_{\leq s} = \{ \{u, v\} \in E \mid u, v \in X_{\leq s}\}$. Given a tree decomposition of treewidth k, we can compute a nice tree-decomposition with treewidth k and $O(kn)$ nodes in $O(k^2(n + m))$ time [10].

A tree-decomposition is called a *path-decomposition* if the tree is a path. The *pathwidth* of a graph is the smallest width of its path-decompositions. Every path-decomposition has a nice path-decomposition of the same pathwidth, which consists of leaf, introduce, forget, but not join nodes.

The problem we tackle in this paper is given as follows.

Definition 1. *For a graph class $\mathcal{C}$, the $\mathcal{C}$-*EDGE-DELETION *is a problem to find the minimum natural number c such that there is a subgraph $G' = (V, E')$ of G with $G' \in \mathcal{C}$ and $|E| - |E'| = c$ for an input simple graph $G = (V, E)$.*

In the succeeding sections, for different classes $\mathcal{C}$ of intersection graphs, we present algorithms for $\mathcal{C}$-EDGE-DELETION that run in linear time in the input graph size when the treewidth is bounded. We assume that the algorithm takes a nice tree-decomposition T of G in addition as input [2,17]. Our algorithms are dynamic programming algorithms that recursively compute solutions (and some auxiliary information) for the subproblems on $(X_{\leq s}, E_{\leq s})$ for each node s in the given tree-decomposition from leaves to the root.

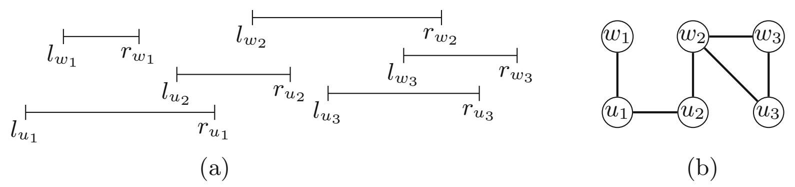


Fig. 2. (a) Interval representation ρ. (b) Interval graph G_ρ. We have $\mathscr{A}(\rho, s) = (\pi, I, J, K, c)$ for $X_s = \{u_1, u_2, u_3\}$ and $X_{\leq s} - X_s = \{w_1, w_2, w_3\}$ where $I = \{l_{u_1}, r_{u_1}, r_{u_2}, l_{u_3}, r_{u_3}\}$, $J = \{r_{u_2}, l_{u_3}, r_{u_3}\}$, and $K = \{r_{u_2}, l_{u_3}\}$.

3 Finding a Largest Interval Subgraph

An *interval representation* π over a set X is a linear order over the set $LR_X = L_X \cup R_X \cup \{\bot, \top\}$ with $L_X = \{ l_x \mid x \in X \}$ and $R_X = \{ r_x \mid x \in X \}$ such that $\bot <_\pi l_x <_\pi r_x <_\pi \top$ for all $x \in X$. Let $\langle\langle p_1, p_2\rangle\rangle_\pi = \{ q \in LR_X \mid p_1 \leq_\pi q <_\pi p_2 \}$ and for $Y \subseteq X$, let

$$[\![Y]\!]_\pi = \bigcup_{u \in Y} \langle\langle l_u, r_u\rangle\rangle_\pi = \{ q \in LR_X \mid l_u \leq_\pi q <_\pi r_u \text{ for some } u \in Y \},$$

which may contain some elements of $LR_X - LR_Y$. The *interval graph G_π of an interval representation π on V* is (V, E_π) where

$$E_\pi = \{\, \{u, v\} \subseteq V \mid \langle\!\langle l_u, r_u \rangle\!\rangle_\pi \cap \langle\!\langle l_v, r_v \rangle\!\rangle_\pi \neq \emptyset \text{ and } u \neq v \,\}.$$

Figure 2 shows an example of an interval graph.

This section presents an FPT algorithm for the interval edge deletion problem w.r.t. the treewidth. Let $G = (V, E)$ be an input graph and s a node of a nice tree-decomposition T of G. On each node s of T, for each interval representation ρ over $X_{\le s}$ that gives an interval subgraph of $(X_{\le s}, E_{\le s})$, we would like to remember some pieces of information about ρ, which we call the "abstraction" of ρ. The abstraction includes the linear order π over LR_{X_s} that restricts ρ. In addition, we remember how the intervals of the forgotten vertices intersect the points of the active vertices using three sets $I, J, K \subseteq LR_X$. Figure 3 explains the meaning of those sets. When $p \in I$, $(p, \mathsf{suc}_\pi(p))$ intersects with a forgotten interval. If $p \in J$, p is within a forgotten interval. Moreover if $p \in K$, then the interval $(p, \mathsf{suc}_\pi(p))$ is properly covered by some forgotten intervals. Using those sets, we can introduce new interval without making it intersect with forgotten intervals.

More formally, for an interval representation ρ over $X_{\le s}$ such that G_ρ is a subgraph of $(X_{\le s}, E_{\le s})$, we define the *abstraction $\mathscr{A}(\rho, s)$ of ρ for s* to be the quintuple (π, I, J, K, c) such that

- π is the restriction of ρ to LR_{X_s},
- $I = \{\, p \in LR_{X_s} \mid \langle\!\langle p, \mathsf{suc}_\pi(p) \rangle\!\rangle_\rho \cap [\![X_{\le s} - X_s]\!]_\rho \neq \emptyset \,\}$,
- $J = \{\, p \in LR_{X_s} \mid p \in [\![X_{\le s} - X_s]\!]_\rho \,\}$,
- $K = \{\, p \in LR_{X_s} \mid \langle\!\langle p, \mathsf{suc}_\pi(p) \rangle\!\rangle_\rho \subseteq [\![X_{\le s} - X_s]\!]_\rho \,\}$,
- $c = |E_{\le s} - E_\rho - E_s|$
  $= |\{\, \{u, v\} \in E_{\le s} \mid u \notin X_s \text{ and } \langle\!\langle l_u, r_u \rangle\!\rangle_\rho \cap \langle\!\langle l_v, r_v \rangle\!\rangle_\rho = \emptyset \,\}|$.

Note that $p \in K$ implies $p, \mathsf{suc}_\pi(p) \in J$. Moreover, if $p \in J$ or $\mathsf{suc}_\pi(p) \in J$, then $p \in I$. We say that $\mathscr{A}(\rho', s) = (\pi', I', J', K', c')$ *dominates* $\mathscr{A}(\rho, s) = (\pi, I, J, K, c)$ iff $\pi' = \pi$, $I' \subseteq I$, $J' \subseteq J$, $K' \subseteq K$, and $c' \le c$. In this case, every possible way of introducing new intervals to ρ is also possible for ρ' by cheaper or equivalent cost. Therefore, it is enough to remember $\mathscr{A}(\rho', s)$ discarding $\mathscr{A}(\rho, s)$. We call a set of abstractions *reduced* if it has no pair of distinct elements such that one dominates the other.

Our algorithm calculates a reduced set $\mathscr{I}_s$ of abstractions of interval representations of interval subgraphs of $(X_{\le s}, E_{\le s})$ for each node s of T which satisfies the following invariant.

Condition 1

- *Every element $(\pi, I, J, K, c) \in \mathscr{I}_s$ is the abstraction of some interval representation of an interval subgraph of $(X_{\le s}, E_{\le s})$ for X_s,*
- *Any interval representation ρ of any interval subgraph of $(X_{\le s}, E_{\le s})$ has an element of $\mathscr{I}_s$ that dominates its abstraction $\mathscr{A}(\rho, X_s)$.*

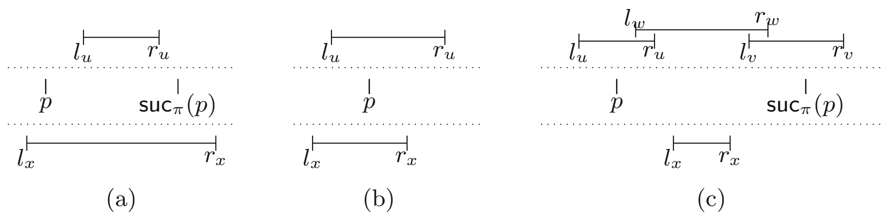


Fig. 3. Typical situations with (a) $p \in I$, (b) $p \in J$, (c) $p \in K$, where $\mathscr{A}(\rho, s) = (\pi, I, J, K, c)$ and $u, v, w \in X_{\leq s} - X_s$ are forgotten vertices. In the respective cases, we cannot introduce a new interval (l_x, r_x) that (a) covers $(p, \mathsf{suc}_\pi(p))$, (b) includes p, (c) overlaps $(p, \mathsf{suc}_\pi(p))$.

Since $\mathscr{I}_s$ is reduced, if $X_s = \emptyset$, we have $\mathscr{I}_s = \{(o, I, \emptyset, \emptyset, c)\}$ for some $I \subseteq \{\bot\}$ and $c \in \mathbb{N}$, where o is the trivial order such that $\bot <_o \top$. Particularly for the root node s, the number c is the least number such that one can obtain an interval subgraph by removing c edges from G. That is, c is the solution to our problem. If s is a leaf, $\mathscr{I}_s = \{(o, \emptyset, \emptyset, \emptyset, 0)\}$ by definition. It remains to show how to calculate $\mathscr{I}_s$ from the child(ren) of s, while preserving the invariant (Condition 1).

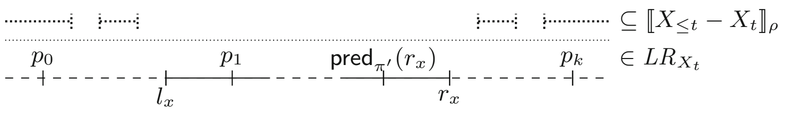
Introduce Node: Suppose that s has just one child t such that $X_s = X_t \cup \{x\}$. For (π, I, J, K, c) in $\mathscr{I}_t$, we say an extension π' of π to X_s *respects* E, I, J, K if

- $\{x, u\} \notin E$ for $u \in X_t$ implies $\langle\!\langle l_x, r_x \rangle\!\rangle_{\pi'} \cap \langle\!\langle l_u, r_u \rangle\!\rangle_{\pi'} = \emptyset$,
- $p \in I$ implies $\langle\!\langle p, \mathsf{suc}_\pi(p) \rangle\!\rangle_{\pi'} \nsubseteq \langle\!\langle l_x, r_x \rangle\!\rangle_{\pi'}$,
- $p \in J$ implies $p \notin \langle\!\langle l_x, r_x \rangle\!\rangle_{\pi'}$,
- $p \in K$ implies $\langle\!\langle p, \mathsf{suc}_\pi(p) \rangle\!\rangle_{\pi'} \cap \langle\!\langle l_x, r_x \rangle\!\rangle_{\pi'} = \emptyset$,

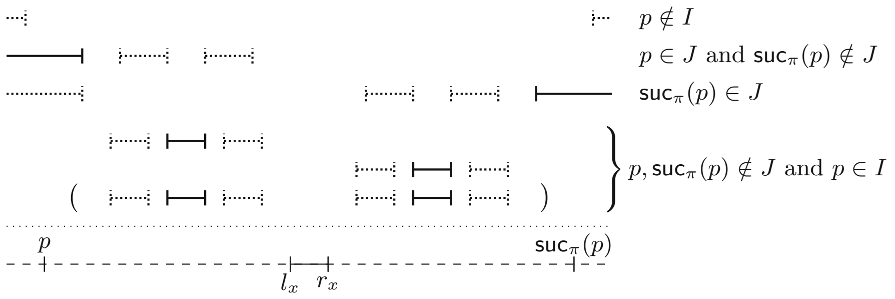
respectively. If π' does not respect some of E, I, J, K, then it means that we are creating an edge between two vertices which are not connected in the input graph G. For each interval representation π' extending π to LR_{X_s} that respects E, I, J, K, we put one or two elements into $\mathscr{I}'_s$ by the following manner. If $r_x \neq \mathsf{suc}_{\pi'}(l_x)$, we add $(\pi', I[r_x/\mathsf{pred}_{\pi'}(r_x)], J, K, c)$ to $\mathscr{I}'_s$ (see Fig. 4 (a)). Otherwise, let $p = \mathsf{pred}_{\pi'}(l_x)$, for which it holds that $p <_{\pi'} l_x <_{\pi'} r_x <_{\pi'} \mathsf{suc}_\pi(p)$. We have four exhaustive cases shown in Fig. 4 (b). If either $p \notin I$ or $J \cap \{p, \mathsf{suc}_\pi(p)\} = \{p\}$, we add (π', I, J, K, c) to $\mathscr{I}'_s$. If $\mathsf{suc}_\pi(p) \in J$, we add $(\pi', I \cup \{r_x\}, J, K, c)$ to $\mathscr{I}'_s$. If $p \in I$ and $J \cap \{p, \mathsf{suc}_\pi(p)\} = \emptyset$, we add both (π', I, J, K, c) and $(\pi', I[r_x/p], J, K, c)$ to $\mathscr{I}'_s$. Those exhaust all the possibilities. We then obtain $\mathscr{I}_s$ by reducing $\mathscr{I}'_s$.

Forget Node: Suppose that s has just one child t such that $X_t = X_s \cup \{x\}$. For each (π, I, J, K, c) in $\mathscr{I}_t$, in accordance with the definition of abstractions, we add to $\mathscr{I}'_s$ the quintuple (π', I', J', K', c) where

- π' is the restriction of π,
- $I' = \{\, p \in LR_{X_s} \mid \langle\!\langle p, \mathsf{suc}_{\pi'}(p) \rangle\!\rangle_\pi \cap (I \cup \langle\!\langle l_x, r_x \rangle\!\rangle_\pi) \neq \emptyset \,\}$,
- $J' = \{\, p \in LR_{X_s} \mid p \in J \cup \langle\!\langle l_x, r_x \rangle\!\rangle_\pi \,\}$,
- $K' = \{\, p \in LR_{X_s} \mid \langle\!\langle p, \mathsf{suc}_{\pi'}(p) \rangle\!\rangle_\pi \subseteq K \cup \langle\!\langle l_x, r_x \rangle\!\rangle_\pi \,\}$,


$\subseteq [\![X_{\leq t} - X_t]\!]_\rho$
p_0
p_1
$\mathsf{pred}_{\pi'}(r_x)$
p_k
$\in LR_{X_t}$
l_x
r_x

(a) When $r_x \neq \mathsf{suc}_{\pi'}(l_x)$. We assume possible forgotten intervals should appear left to l_x and right to r_x to prevent (l_x, r_x) from intersecting forgotten intervals.


$p \notin I$
$p \in J$ and $\mathsf{suc}_\pi(p) \notin J$
$\mathsf{suc}_\pi(p) \in J$
$p, \mathsf{suc}_\pi(p) \notin J$ and $p \in I$
p
$\mathsf{suc}_\pi(p)$
l_x
r_x

(b) When $r_x = \mathsf{suc}_{\pi'}(l_x)$. Our algorithm does not consider all the admissible extensions ρ' of π'. For example, we do not put $(\pi, I \cup \{p, r_x\}, J, K, c)$ into $\mathscr{I}'_s$ as illustrated in the parentheses above, since it is dominated by other possibilities and will be absent in $\mathscr{I}_s$ anyway.

Fig. 4. Illustrating how we insert l_x and r_x into previously determined interval representation. Thick lines illustrate $[\![X_{\leq t} - X_t]\!]_\rho$ for a possible extension ρ such that $\mathscr{A}(\rho, X_t) = (\pi, I, J, K, c)$.

– $c' = c + |\{\, \{x,u\} \in E \mid \langle\!\langle l_u, r_u \rangle\!\rangle_\pi \cap \langle\!\langle l_x, r_x \rangle\!\rangle_\pi = \emptyset$ and $u \in X_s \,\}|$.

Then we obtain $\mathscr{I}_s$ by reducing $\mathscr{I}'_s$.

Join Node: Suppose that s has two children t_1 and t_2, where $X_s = X_{t_1} = X_{t_2}$. We say that $A_1 = (\pi_1, I_1, J_1, K_1, c_1) \in \mathscr{I}_{t_1}$ and $A_2 = (\pi_2, I_2, J_2, K_2, c_2) \in \mathscr{I}_{t_2}$ are *compatible* if $\pi_1 = \pi_2$ and $J_1 \cap J_2 = I_1 \cap K_2 = K_1 \cap I_2 = \emptyset$. If A_1 and A_2 are not compatible, any interval representation ρ on $X_{\leq s}$ which extends ρ_1 and ρ_2 will connect two vertices which are not adjacent in the input graph G for any interval representations ρ_i on $X_{\leq t_i}$ of which A_i is the abstraction for $i = 1, 2$. For each compatible pair (A_1, A_2), one can find an interval representation ρ on $X_{\leq s}$ that forms a subgraph of $(X_{\leq s}, E_{\leq s})$ which extends some ρ_1 and ρ_2 whose abstractions are A_1 and A_2, respectively. Then we add the quintuple $(\pi_1,\ I_1 \cup I_2,\ J_1 \cup J_2,\ K_1 \cup K_2,\ c_1 + c_2)$ to $\mathscr{I}'_s$. We obtain $\mathscr{I}_s$ by reducing $\mathscr{I}'_s$.

Theorem 1. *The edge deletion problem for interval graphs can be solved in $O(|V|N^2\mathrm{poly}(k))$ time where $N = (2k)! \cdot 2^{2k}$ for the treewidth k of G. If k is the pathwidth, it can be solved in $O(|V|N\mathrm{poly}(k))$ time.*

Proof. Let k be the maximum size of the assigned set X_s to a node of a nice tree-decomposition. Each $\mathscr{I}_s$ may contain at most $N = (2k)!/2^k \cdot (2^k)^3 = (2k)! \cdot 2^{2k}$

elements. To calculate $\mathscr{I}_s$ from $\mathscr{I}_t$ for children t of s, it takes $O(N^\ell \mathrm{poly}(k))$ time for some polynomial function poly, if it has at most ℓ children. Since the nice tree-decomposition has $O(|V|)$ nodes, we obtain the conclusion.

4 Algorithms for Other Graph Classes

We in this section present Edge-Deletion algorithms on some graph classes related to interval graphs by modifying the algorithm in the previous section.

4.1 Proper Interval Graphs

An interval representation π is said to be *proper* if there are no $u, v \in V$ such that $l_u <_\pi l_v <_\pi r_v <_\pi r_u$. An interval graph is *proper* if it admits a proper interval representation. The algorithm presented in Sect. 3 can easily be modified so that it solves the edge deletion problem for proper interval graphs. In accordance with the definition of a proper interval representation, we simply require π' in *Introduce Node* to be a proper interval representation. Under the restriction, we see that $I = \{\, p \mid p \in J \text{ or } \mathsf{suc}_\pi(p) \in J \,\}$ if $(\pi, I, J, K, c) \in \mathscr{I}_s$. Then, I can be discarded from each abstraction.

Corollary 1. *The edge deletion problem for proper interval graphs can be solved in $O(|V|N^2\mathrm{poly}(k))$ time where $N = (2k)! \cdot 2^k$ for the treewidth k of G. If k is the pathwidth, it can be solved in $O(|V|N\mathrm{poly}(k))$ time.*

4.2 Trivially Perfect Graphs

An interval representation π is said to be *nested* if there are no $u, v \in V$, such that $l_u <_\pi l_v <_\pi r_u <_\pi r_v$. A *trivially perfect graph* is an interval graph that admits a nested interval representation. The algorithm presented in Sect. 3 can easily be modified so that it solves the edge deletion problem for trivially perfect graphs. In accordance with the definition of the graph class, we simply require π' in *Introduce Node* to be a nested interval representation. Under the restriction, we see that $l_v \in J$ if and only if $r_v \in J$ if and only if $l_v \in K$. Therefore, we do not need to have the set J any more.

Corollary 2. *The edge deletion problem for trivially perfect graphs can be solved in $O(|V|N^2\mathrm{poly}(k))$ time where $N = (2k)! \cdot 2^k$ for the treewidth k of G. If k is the pathwidth, it can be solved in $O(|V|N\mathrm{poly}(k))$ time.*

4.3 Circular-Arc Graphs

Circular-arc graphs are a generalization of interval graphs which have a "circular" interval representation. For a linear order π over a set S, we let

$$\langle\!\langle p_1, p_2 \rangle\!\rangle_\pi = \begin{cases} \{\, q \in S \mid p_1 \leq_\pi q <_\pi p_2 \,\} & \text{if } p_1 \leq_\pi p_2, \\ \{\, q \in S \mid p_1 \leq_\pi q \vee q <_\pi p_2 \,\} & \text{if } p_2 <_\pi p_1. \end{cases}$$

A *circular-arc graph* is a graph $G_\pi = (V, E)$ such that

$$E = \{\, \{u, v\} \subseteq V \mid \langle\!\langle l_u, r_u \rangle\!\rangle_\pi \cap \langle\!\langle l_v, r_v \rangle\!\rangle_\pi \neq \emptyset \,\}$$

for some linear order π over $L_V \cup R_V$. Note that this set contains neither $\top$ nor $\bot$. The algorithm presented in Sect. 3 can easily be modified so that it solves the edge deletion problem for circular-arc graphs by replacing the definition of $\langle\!\langle p_1, p_2 \rangle\!\rangle_\pi$ as above, and defining $\mathsf{suc}_\pi(\max_\pi LR_V) = \min_\pi LR_V$ and $\mathsf{pred}_\pi(\min_\pi LR_V) = \max_\pi LR_V$. Since we allow $r_u <_\pi l_u$, the number of admissible circular interval representations is bigger than that of (ordinary) interval representations. This affects the computational complexity.

Corollary 3. *The edge deletion problem for circular-arc graphs can be solved in $O(|V|N^2\mathrm{poly}(k))$ time where $N = (2k)! \cdot 2^{3k}$ for the treewidth k of G. If k is the pathwidth, it can be solved in $O(|V|N\mathrm{poly}(k))$ time.*

4.4 Threshold Graphs

Threshold graphs are special cases of trivially perfect graphs, which can be defined in several different ways. Here we use a pair of a vertex subset $W \subseteq V$ and a linear order π over R_V as a *threshold interval representation.* We say that vertices u and v *intersect* on (W, π) if and only if $u \in W$ and $r_v <_\pi r_u$ or the other way around. A *threshold graph* is a graph $G_{W,\pi} = (V, E_{W,\pi})$ where (W, π) is a threshold interval representation on V and $E_{W,\pi} = \{\, \{u, v\} \subseteq V \mid u$ and v intersect on $(W, \pi)\,\}$. By extending π to π' over LR_V so that $l_w <_{\pi'} r_v$ for all $w \in W$ and $v \in V$ and $\mathsf{suc}_{\pi'}(l_u) = r_u$ for all $u \in V - W$, then the induced interval graph coincides with the threshold graph. To attain drastic improvement on the complexity, we design an algorithm for the edge deletion problem for threshold graphs from scratch, rather than modifying the one for interval graphs.

For a threshold representation (Y, ρ) of a subgraph $G_{Y,\rho} = (X_{\le s}, E_{Y,\rho})$, we define its abstraction $\mathscr{A}((Y, \rho), s) = (Y', \pi, b, p, c)$ as follows: (1) π is the restriction of ρ to R_{X_s}, (2) $Y' = Y \cap X_s$, (3) if $Y' = Y$, then $b = 0$ and $p = \max_\pi\{\, p \in R_{X_s} \mid p <_\rho r_y$ for all $y \in X_{\le s} - X_s \,\}$, (4) if $Y' \neq Y$, then $b = 1$ and $p = \max_\pi\{\, p \in R_{X_s} \mid p <_\rho r_y$ for some $y \in Y - Y' \,\}$, and (5) $c = |E_{\le s} - E_{Y,\rho} - E_s| = |\{\, \{u, v\} \in E \mid \{u, v\} \nsubseteq X_s$ and u and v do not intersect on $(Y, \rho) \,\}|$.

We say that (Y', π', b', p', c') *dominates* (Y, π, b, p, c) if $Y' = Y$, $\pi' = \pi$, $c' \le c$, and either (a) $b' = b = 0$ and $p' \ge_\pi p$, (b) $b' = b = 1$ and $p' \le_\pi p$, or (c) $b' = 0$ and $b = 1$. Using the above invariant, we can provide an algorithm for Threshold-Edge-Deletion.

Theorem 2. *The edge deletion problem for threshold graphs can be solved in $O(|V|N^2\mathrm{poly}(k))$ time where $N = k! \cdot 2^k$ for the treewidth k of G. If k is the pathwidth, it can be solved in $O(|V|N\mathrm{poly}(k))$ time.*

5 Conclusion

We propose FPT algorithms for Edge-Deletion to some intersection graphs parameterized by treewidth in this paper. Our algorithms maintain partial intersection models on a node of a tree decomposition with some restrictions and extend the models consistently for the restrictions in the next step. We expect that the ideas in our algorithms can be applied to other intersection graphs whose intersection models can be represented as linear-orders, for example circle graphs, chain graphs and so on, and to Vertex-Deletion of intersection graphs.

We have the following questions as future work:

– Do there exist single exponential time algorithms for the considered problems, that is, $O^*(2^{tw(G)})$ time, or can we show matching lower bounds assuming the Exponential Time Hypothesis?
– Are there FPT algorithms parameterized by treewidth for $\mathcal{C}$-Completion which is to find the minimum number of adding edges to obtain a graph in an intersection graph class $\mathcal{C}$? We can naturally apply the idea of our algorithms to $\mathcal{C}$-Completion problems. While $\mathcal{C}$-Edge-Deletion algorithms do not allow introduced objects to intersect with forgotten objects, $\mathcal{C}$-Completion algorithms do allow it with the cost of addition of new edges. Thus $\mathcal{C}$-Completion algorithms based on this naive approach will be XP algorithms since we have to remember the number of forgotten objects in the representation to count the number of intersections between the introduced objects and forgotten objects.
– Are there FPT algorithms for Edge-Deletion to intersection graphs defined using objects on a plane, like unit disk graphs? The intersection graph classes discussed in this paper are all defined using objects aligned on a line. Going up to a geometric space of higher dimension is a challenging topic.

Acknowledgement. This work was supported in part by JSPS KAKENHI Grant Numbers JP18H04091 and JP19K12098.

References

1. Bessy, S., Perez, A.: Polynomial kernels for proper interval completion and related problems. Inf. Comput. **231**, 89–108 (2013)
2. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
3. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph Classes: A Survey. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1999)
4. Burzyn, P., Bonomo, F., Durán, G.: NP-completeness results for edge modification problems. Discrete Appl. Math. **154**(13), 1824–1844 (2006)
5. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inf. Process. Lett. **58**(4), 171–176 (1996)
6. Cao, Y.: Linear recognition of almost interval graphs. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, 10–12 January 2016, pp. 1096–1115 (2016)

7. Cao, Y., Marx, D.: Interval deletion is fixed-parameter tractable. ACM Trans. Algorithms **11**(3), 21:1–21:35 (2015)
8. Courcelle, B.: The monadic second-order logic of graphs XV: on a conjecture by D. Seese. J. Appl. Logic **4**(1), 79–114 (2006)
9. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach. Encyclopedia of Mathematics and Its Applications, vol. 138. Cambridge University Press (2012)
10. Cygan, M., et al.: Lower bounds for kernelization. In: Cygan, M., et al. (eds.) Parameterized Algorithms, pp. 523–555. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3_15
11. Drange, P.G., Pilipczuk, M.: A polynomial kernel for trivially perfect editing. Algorithmica **80**(12), 3481–3524 (2018). https://doi.org/10.1007/s00453-017-0401-6
12. Frenkel, Z., Paux, E., Mester, D.I., Feuillet, C., Korol, A.B.: LTC: a novel algorithm to improve the efficiency of contig assembly for physical mapping in complex genomes. BMC Bioinform. **11**, 584 (2010). https://doi.org/10.1186/1471-2105-11-584
13. Goldberg, P.W., Golumbic, M.C., Kaplan, H., Shamir, R.: Four strikes against physical mapping of DNA. J. Comput. Biol. **2**(1), 139–152 (1995)
14. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, vol. 57). North-Holland Publishing Co., Amsterdam (2004)
15. Heggernes, P., van't Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity of vertex deletion into perfect graph classes. Theor. Comput. Sci. **511**, 172–180 (2013)
16. van't Hof, P., Villanger, Y.: Proper interval vertex deletion. Algorithmica **65**(4), 845–867 (2013). https://doi.org/10.1007/s00453-012-9661-31
17. Kloks, T. (ed.): Treewidth, Computations and Approximations. LNCS, vol. 842. Springer, Heidelberg (1994). https://doi.org/10.1007/BFb0045375
18. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. Syst. Sci. **20**(2), 219–230 (1980)
19. Liu, Y., Wang, J., You, J., Chen, J., Cao, Y.: Edge deletion problems: Branching facilitated by modular decomposition. Theor. Comput. Sci. **573**, 63–70 (2015)
20. Mancini, F.: Graph modification problems related to graph classes. Ph.D. thesis, University of Bergen, May 2008
21. Margot, F.: Some complexity results about threshold graphs. Discrete Appl. Math. **49**(1–3), 299–308 (1994)
22. Marx, D.: Chordal deletion is fixed-parameter tractable. Algorithmica **57**(4), 747–768 (2010). https://doi.org/10.1007/s00453-008-9233-8
23. Nastos, J., Gao, Y.: Bounded search tree algorithms for parametrized cograph deletion: Efficient branching rules by exploiting structures of special graph classes. Discrete Math., Alg. and Appl. **4**(1) (2012)
24. Natanzon, A., Shamir, R., Sharan, R.: Complexity classification of some edge modification problems. Discrete Appl. Math. **113**(1), 109–128 (2001)
25. Sharan, R.: Graph Modification Problems and their Applications to Genomic Research. Ph.D. thesis, University of Bergen, May 2008
26. Spinrad, J.P.: Efficient Graph Representations. Fields Institute Monographs. American Mathematical Society, Providence (2003)
27. Waterman, S., Griggs, M.R.: Interval graphs and maps of DNA. J. Bull. Math. Biol. **48**, 189–195 (1986)
28. Villanger, Y., Heggernes, P., Paul, C., Telle, J.A.: Interval completion is fixed parameter tractable. SIAM J. Comput. **38**(5), 2007–2020 (2009)