k-Gap Interval Graphs

Fedor V. Fomin[1], Serge Gaspers[2], Petr Golovach[3], Karol Suchan[45], Stefan Szeider[2], Erik Jan van Leeuwen[1], Martin Vatshelle[1], and Yngve Villanger[1]

[1] Department of Informatics, University of Bergen, Bergen, Norway.
{fomin, e.j.van.leeuwen, vatshelle, yngvev}@ii.uib.no
[2] Inst. of Information Systems, Vienna University of Technology, Vienna, Austria.
gaspers@kr.tuwien.ac.at, stefan@szeider.net
[3] School of Engineering and Computing Sciences, Durham University, Durham, UK.
petr.golovach@durham.ac.uk
[4] Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile.
karol.suchan@uai.cl
[5] WMS, AGH - University of Science and Technology, Krakow, Poland.

Abstract. We initiate the study of a new parameterization of graph problems. In a multiple interval representation of a graph, each vertex is associated to at least one interval of the real line, with an edge between two vertices if and only if an interval associated to one vertex has a nonempty intersection with an interval associated to the other vertex. A graph on n vertices is a k-gap interval graph if it has a multiple interval representation with at most $n + k$ intervals in total. In order to scale up the nice algorithmic properties of interval graphs (where $k = 0$), we parameterize graph problems by k, and find FPT algorithms for several problems, including Feedback Vertex Set, Dominating Set, Independent Set, Clique, Clique Cover, and Multiple Interval Transversal. The Coloring problem turns out to be W[1]-hard and we design an XP algorithm for the recognition problem.

1 Introduction

A multiple interval representation f of a graph $G = (V, E)$ is a mapping which assigns to each vertex of G a non-empty collection of intervals on the real line so that two distinct vertices u and v are adjacent if and only if there are intervals $I \in f(u)$ and $J \in f(v)$ with $I \cap J \neq \emptyset$. For a vertex v, $|f(v)|$ denotes the number of intervals that f assigns to v. The *interval number* of G is defined as

$$i(G) = \min \left\{ \max_{v \in V} \{|f(v)|\} : f \text{ is a multiple interval representation of } G \right\} .$$

The *total interval number* of a graph $G = (V, E)$ is defined as

$$I(G) = \min \left\{ \sum_{v \in V} \{|f(v)|\} : f \text{ is a multiple interval representation of } G \right\} .$$

The class of *t-interval graphs* is defined as the class of all graphs G with $i(G) \leq t$. This natural generalization of interval graphs was independently introduced by Trotter and Harary [46], and by Griggs and West [29].

Even for small fixed $t \geq 2$, these graph classes are much richer than interval graphs. For example, the class of 2-interval graphs includes circular-arc graphs, outerplanar graphs, cubic graphs, and line graphs, and the class of 3-interval graphs includes all planar graphs [43]. Unfortunately, many problems remain NP-hard on 2-interval graphs (for example, their recognition [51], 3-Coloring, Dominating Set, Independent Set, and Hamiltonian Cycle) or 3-interval graphs (for example Clique [15], whose complexity on 2-interval graphs is open [15, 44]). Parameterized by solution size, Independent Set, Dominating Set, and Independent Dominating Set are $W[1]$-hard on 2-interval graphs, even when all intervals have unit length, whereas Clique is FPT [23].

With the objective to generalize interval graphs while maintaining their nice algorithmic properties, we define *k-gap interval graphs* as graphs that have a multiple interval representation whose total number of intervals exceeds the number of vertices by at most k. Parameterizing problems by k becomes then a reasonable attempt to scale up the nice algorithmic properties of interval graphs to more general graphs.

Definition 1. *A graph G on n vertices is a* k-gap interval graph *if $I(G) \leq n+k$.*

Throughout this paper, we assume that problems that have a k-gap interval graph as input also have access to the corresponding multiple interval representation. An alternative definition of k-gap interval graphs is via the identification operation.

Definition 2. *Given a graph $G = (V, E)$ and two vertices $a, b \in V$, the graph obtained from G by* identifying *a and b is the graph obtained from $G - b$ by adding all edges $\{va : vb \in E\}$.*

Observation 1. *A graph is a k-gap interval graph if and only if it can be obtained from an interval graph by a sequence of at most k operations of identifying pairs of vertices.*

Several of our FPT results do not make use of the special structure of the vertices with gaps, and also hold for the graph class interval+kv.

Definition 3. *A graph $G = (V, E)$ is an* interval+kv graph *if there is a vertex set $X \subseteq V$, with $|X| \leq k$, such that $G \setminus X$ is an interval graph. We refer to such a vertex set X as the* interval deletion set *of G.*

When discussing the complexity of problems on interval+kv graphs, we assume that an interval deletion set is provided as a part of the input, as it is an open question whether Interval Vertex Deletion is FPT [38]. As the set of vertices that are associated to more that one interval in a multiple interval representation is an interval deletion set, FPT results for interval+kv graphs translate to FPT results for k-gap interval graphs. When the generalization of the FPT result for k-gap interval graphs to interval+kv graphs is straightforward, we state the stronger result.

Related work. The class of t-interval graphs has been studied from the view point of approximation algorithms [7, 8, 15], graph theory (see, for example [2, 6, 22, 29, 43, 46, 50] and references), classical complexity [51], and parameterized complexity [11, 23]. Applications have been identified in scheduling and resource allocation [7, 8], communication protocols [15], computational biology [4, 5, 11, 17, 20, 25, 48, 49], and monitoring [15]. The total interval number was introduced in [29] and studied in [3, 16, 18, 34–36, 41].

Notation and definitions. Let $G = (V, E)$ be a graph, $u \in V$ be a vertex, and $S \subseteq V$ be a subset of vertices. The *open neighborhood* of v is defined as $N(u) = \{v : uv \in E\}$, its *closed neighborhood* is $N[u] = N(u) \cup \{u\}$, and its degree is $d(u) = |N(u)|$. Also, $N(S) = \bigcup_{v \in S} N(v) \setminus S$ and $N[S] = N(S) \cup S$. The graph $G \setminus S$ is obtained from G by removing all vertices in S and all edges that have at least one endpoint in S. The graph induced on S is $G[S] = G \setminus (V \setminus S)$. By n and m we generally denote the number of vertices and edges of G.

In a k-gap interval graph $G = (V, E)$ with multiple interval representation f we say that a vertex $v \in V$ has a gap if $|f(v)| \geq 2$. We denote by $\mathsf{gap}_f(G)$ the set of vertices that have gaps and omit the subscript if the context specifies the multiple interval representation. We say that a multiple interval representation of G has k gaps if $\sum_{v \in V} |f(v)| = |V| + k$.

A *tree decomposition* of a graph G is a pair (B, T) where T is a tree and $B = \{B_i \mid i \in V(T)\}$ is a collection of subsets (called *bags*) of $V(G)$ such that:

1. $\bigcup_{i \in V(T)} B_i = V(G)$,
2. for each edge $\{x, y\} \in E(G)$, $x, y \in B_i$ for some $i \in V(T)$, and
3. for each $x \in V(G)$ the set $\{i \in V(T) : x \in B_i\}$ induces a connected subtree of T.

The *width* of a tree decomposition $(\{B_i \mid i \in V(T)\}, T)$ is $\max_{i \in V(T)} \{|B_i| - 1\}$. The *treewidth* of a graph G (denoted $\mathbf{tw}(G)$) is the minimum width over all tree decompositions of G. If, in the above definitions, we restrict T to be a path then we define the notions of *path decomposition* and *pathwidth* of G (denoted $\mathbf{pw}(G)$).

We refer to [21, 24, 39] for texts on parameterized complexity, the theoretical framework of our investigations. Proofs of statements marked with $(\star)$ have been moved to Appendix A.

2 Recognition

The problem of recognizing k-gap interval graphs is to determine whether for a graph G on n vertices, $I(G) \leq n + k$. If k is part of the input, the problem is NP-hard, as it is NP-hard to decide whether $I(G) \leq n + (m + 1 - n)$, even if G is a planar, 3-regular, triangle-free graph [36]. We show however that the problem is polynomial-time decidable if k is a constant. First, we need a bound on the number of maximal cliques in k-gap interval graphs.

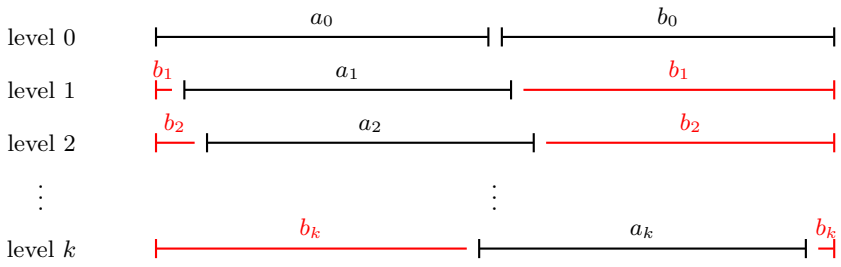

Fig. 1. A multiple interval representation with k gaps of a graph with 2^{k+1} maximal cliques.

2.1 Maximal Cliques

A *clique* in a graph G is a set of vertices that are all pairwise adjacent in G. A clique is *maximal* if it is not a subset of another clique.

Lemma 1. *An interval+kv graph on n vertices has $O(2^k \cdot n)$ maximal cliques.*

Proof. Let $G = (V, E)$ be an interval+kv graph and X be its interval deletion set. Let $Y \subseteq X$ and consider all maximal cliques of G whose intersection with X is exactly Y. For any such maximal clique S, $S \setminus Y$ is a maximal clique of $G[\bigcap_{v \in Y} N(v) \setminus X]$. As $G[\bigcap_{v \in Y} N(v) \setminus X]$ is an interval graph, there are $O(n)$ choices for S. As there are $O(2^k)$ choices for Y, the lemma follows. □

On the other hand, Lemma 1 cannot be substantially improved, even for k-gap interval graphs, as there are k-gap interval graphs with $\Omega(2^k)$ maximal cliques. Figure 1 represents a multiple interval representation with k gaps of a graph $G = (V, E)$ with vertex set $V = \{a_0, \cdots, a_k, b_0, \cdots, b_k\}$ and an edge between every pair of distinct vertices except a_i and b_i, $0 \leq i \leq k$. Any vertex set containing exactly one of a_i, b_i, $0 \leq i \leq k$ forms a maximal clique. Thus, this graph has 2^{k+1} maximal cliques.

2.2 PQ-trees

To recognize k-gap interval graphs, we make use of PQ-trees. A *PQ-tree* is a rooted tree T that represents allowed permutations over a set U. Each leaf corresponds to a unique element of U. Internal nodes are labeled P or Q. The children of an internal node v appear in a particular order, which can be modified depending on the label of v. The order can be reversed if the label is Q, and it can be arbitrarily changed if the label is P. In this way, the tree imposes various restrictions on the order in which the leaves appear. PQ-trees were famously used to provide a linear-time recognition algorithm for interval graphs [13].

Booth and Lueker [13] introduced PQ-trees, together with a *reduction algorithm*. This algorithm, given a PQ-tree T and a collection $\mathbb{S}$ of sets, restricts

the set of permutations represented by T to those in which the elements of each $S \in \mathbb{S}$ appear consecutively. It runs in time $O(|U| + |\mathbb{S}| + \sum_{S \in \mathbb{S}} |S|)$.

Our recognition algorithm for k-gap interval graphs will construct a PQ-tree T and add additional constraints to T. We describe these constraints now and propose an algorithm to check whether they can be met by T. First, we give some notation. We say that $u \in U$ is to the *left* of $v \in U$ in T if the order of the leaves induced by T is such that u comes before v. We can then also define *right*, *leftmost*, and *rightmost* in a natural way. We say that a set of leaves is *consecutive* in T if they appear consecutively in the order of the leaves induced by the tree.

We now give the type of constraints that we will impose on PQ-trees. A PQ-tree T over U *satisfies a partition constraint* $B = (i, u_L^1, u_R^1, \ldots, u_L^i, u_R^i, S)$, where $\{u_L^1, u_R^1, \ldots, u_L^i, u_R^i\} \subseteq S \subseteq U$ if S can be partitioned into $S_1, \ldots, S_i$ such that each S_j is consecutive, u_L^j is the leftmost leaf of S_j, and u_R^j is the rightmost leaf of S_j. Moreover, S_j appears to the left of S_{j+1} for all $1 \leq j < i$. We use S'_B to denote the set $S \backslash \{u_L^1, u_R^1, \ldots, u_L^i, u_R^i\}$.

We show that, given a PQ-tree and a set of partition constraints, we can decide in polynomial time whether the PQ-tree can be ordered to satisfy these constraints. If so, our algorithm finds the order and the partitions $S_1, \ldots, S_i$ for each of the constraints.

Lemma 2 ($\star$). *Let $\mathcal{Z} = \{B_1, \cdots, B_\ell\}$ be a set of partition constraints such that the sets S'_{B_j} are pairwise disjoint. It can be decided in $(|\mathcal{Z}| \cdot n)^{O(1)}$ time if there exists a valid ordering of the leaves of a PQ-tree T satisfying all constraints in $\mathcal{Z}$.*

2.3 Recognition Algorithm

We now show how to use Lemma 2 to recognize k-gap interval graphs. The algorithm tries to construct a multiple interval representation for G with at most k gaps. It guesses an interval deletion set X for G and a multiple interval representation of $G[X]$. Then, it constructs a PQ-tree T for $G \setminus X$ and adds partition constraints to T that need to be fulfilled by an interval representation of $G \setminus X$ to be merged with the multiple interval representation of $G[X]$. Lemma 2 can then check whether the guesses led to a multiple interval representation of G with k gaps. We only sketch the proof here and refer to Appendix A for the full proof.

Theorem 1 ($\star$). *Given a graph G, one can decide whether $I(G) \leq n + k$ in polynomial time if k is a constant.*

Proof (Sketch). As a first step, the algorithm guesses an interval deletion set X of G of size at most k and it guesses the number of intervals that are assigned to each vertex of X, such that the total number of intervals is at most $|X| + k$. In total there are $O(n^k)$ choices. For each choice, the algorithm checks that $G \setminus X$ is an interval graph, because otherwise it can immediately move to the next

choice for X. The algorithm also guesses the order of all endpoints of intervals associated with vertices in X. There are at most $(4k)!$ different permutations. The ordering defines a multiple interval representation f of $G[X]$ and determines the way the vertices of X overlap. If this ordering does not match with the edges of $G[X]$, disregard the current guess.

As $G \setminus X$ is a interval graph we can find all the maximal cliques in polynomial time using a perfect elimination order [42]. We also find all maximal cliques of G using Lemma 1 and a polynomial delay enumeration algorithm [31].

Suppose f can be extended into a multiple interval representation f' for G by assigning exactly one interval to each vertex from $V \setminus X$. Consider some endpoint p of an interval in f. Then, p defines a clique of $G \setminus X$, contained within the neighborhoods of some vertices from X. For each endpoint of an interval in f, the algorithm guesses this clique and the clique that comes just before p. Build a PQ-tree of the maximal cliques of G restricted to $G \setminus X$ plus the cliques corresponding to endpoints of intervals in f. Then, partition all the cliques in the PQ-tree into sets depending on what subset of intervals from f they will belong to.

Finally we use this partition to add partition constraints to the PQ-tree and apply Lemma 2. Once we have the order of the cliques in the PQ-tree a multiple interval representation with k gaps can easily be obtained. □

3 FPT Results

The Multiple Interval Transversal problem is specific to multiple interval graphs. This problem and its variants is well studied for t-interval graphs (see for example [1, 30, 32, 45]). Given a graph G, a multiple interval representation f of G, and a positive integer p, the problem asks whether there is a set P of p points on the real line such that each vertex of G is associated to an interval containing a point from P. By relating this problem to a problem from Constraint Satisfaction, we obtain the following result.

Theorem 2. *The* Multiple Interval Transversal *problem, parameterized by k has a $O(k^2)$-vertex kernel and can be solved in time $O(1.6181^k k^2 + n)$ on k-gap interval graphs, where n is the number of vertices of the input graph.*

Proof. The Consistency problem for AtMost-NValue contraints has as input a set of variables $X = \{x_1, \ldots, x_{n'}\}$, a totally ordered set of values D, a map $dom : X \to 2^D$ assigning a non-empty domain $dom(x) \subseteq D$ to each variable $x \in X$, and an integer N. The question is whether there exists an assignment of the variables from X to values from their domain such that the number of distinct values taken by variables from X is at most N.

Bessière *et al.* [10] were the first to parameterize this problem by the total number k' of holes in the domains of the variables. Here, a hole in the domain of a variable x is a couple $(u, w) \in dom(x) \times dom(x)$, such that there is a value $v \in D \setminus dom(x)$ with $u < v < w$ and there is no value $v' \in dom(x)$ with

$u < v' < w$. The problem has a kernel with $O(k'^2)$ variables and domain values and can be solved in time $O(1.6181^{k'}k'^2 + n' + |D|)$ [26].

The theorem will follow by a simple reduction of a Multiple Interval Transversal instance $(G = (V, E), f, p)$ with parameter k to an instance (X, D, dom, N) with parameter $k' = k$ of the Consistency problem for AtMostNValue contraints. Let $F := \{l, r : [l, r] \in f(v), v \in V\}$ denote the set of all left and right endpoints of intervals in f. The reduction sets $X := V$, $D := F$, $dom(x) := \bigcup_{I \in f(x)} I \cap F$, and $N := p$. It is easy to see that both instances are equivalent and that $k' = k$. □

A vertex subset U is a *feedback vertex set* in a graph G if $G \setminus U$ has no cycle. The Feedback Vertex Set problem has as input a graph G and a positive integer p, and the question is whether G has a feedback vertex set of size at most p.

Theorem 3. Feedback Vertex Set *can be solved in time* $2^{O(k \log k)} \cdot n^{O(1)}$ *on interval+kv graphs with* n *vertices.*

Proof. We design a dynamic-programming algorithm to solve Feedback Vertex Set on interval+kv graphs. The key observation is that any feedback vertex set misses at most two vertices of any clique of G.

Any interval graph (see e.g. [28]) has a path decomposition whose set of bags is exactly the set of maximal cliques. Kloks [33] showed that every path decomposition of a graph G can be converted in linear time to a *nice path decomposition*, such that the size of the largest bag does not increase, and the total size of the path is linear in the size of the original path. A path decomposition (B, P) is *nice* if P is a path with nodes $1, \ldots, r$ such that the nodes of P are of two types:

1. an *introduce node* i with $B_i = B_{i-1} \cup \{v\}$ for some vertex $v \in V$ (we assume that $X_0 = \emptyset$) ;
2. a *forget node* i with $B_i = B_{i-1} \setminus \{v\}$ for some vertex $v \in V$.

Thus, an interval graph G has a nice path decomposition with the additional property that each bag is a clique in G.

Now we are ready to describe our algorithm for Feedback Vertex Set. Let G be an interval+kv graph with interval deletion set X. Using an interval representation of $G' = G \setminus X$, we construct a path decomposition of G' whose set of bags is the set of maximal cliques of G', and then we construct in linear time a nice path decomposition (B', P') of G' where P' is a path on nodes $1, \ldots, r$. Set $B'_0 := \emptyset$. We construct a path decomposition of G with bags $B_0, \ldots, B_r$ where $B_i = B'_i \cup X$ for $i \in \{0, \ldots, r\}$. Now we apply a dynamic programming algorithm over this path decomposition.

We first describe what is stored in the tables corresponding to the nodes $0, \ldots, r$ of the path. For any $i \in \{0, \ldots, r\}$, we denote by G_i the subgraph of G induced by $\cup_{j=0}^{i} B_j$. For $i \in \{0, \ldots, r\}$, the table stores the records $R = (F, F_i, \mathcal{P}, s)$, where

– $F \subseteq X$;

- $F_i \subseteq B'_i$;
- $\mathcal{P}$ is a partition of $B_i \setminus (F \cup F_i)$; and
- $s \leq n$ is a positive integer;

with the property that there is a feedback vertex set U_i of G_i such that

- $|U_i| \leq s$;
- $U_i \cap X = F$ and $U_i \cap B'_i = F_i$;
- for any set S in $\mathcal{P}$, $x, y \in S$ if and only if x, y are in the same component of $G_i \setminus U_i$.

Clearly, G has a feedback vertex set of size at most p if and only if the table for r contains a record R with $s = p$. The tables are created and maintained in a straightforward way.

It remains to estimate the running time. Since $|X| \leq k$, there are at most 2^k subsets F of X. Each B'_i is a clique. Hence, $|B'_i \setminus F_i| \leq 2$, since otherwise $G_i \setminus U_i$ has a cycle. It follows that we consider at most $\frac{1}{2}n(n+1)+1$ sets F_i. Each set $B_i \setminus (F \cup F_i)$ has size at most $k+2$, and the number of partitions is upper bounded by B_{k+2}, where B_t is the t^{th} Bell number. Finally, s can have at most n values. We conclude that for each $i \in \{0, \ldots, r\}$, the table for i contains at most $O(2^k B_{k+2} \cdot n^3)$ records. It follows that our algorithm runs in time $2^{O(k \log k)} \cdot n^{O(1)}$. □

A *clique cover* of size t of a graph $G = (V, E)$ is a partition of V into $Z_1, Z_2, \ldots, Z_t$ where Z_i is a clique in G, for $1 \leq i \leq t$. The Clique Cover problem has as input a graph G and a positive integer p, and the question is whether G has a clique cover of size p.

Theorem 4. Clique Cover *can be solved in time* $O(2^k \cdot n^{O(1)})$ *and polynomial space on interval+kv graphs with* n *vertices.*

Proof. Before starting we observe that there is a minimum clique cover where Z_1 is a maximal clique of G and in general Z_i is a maximal clique of $G[Z_i \cup Z_{i+1} \cup \ldots Z_t]$. I.e. stealing a vertex from a higher numbered clique will not increase the number of cliques in the cover.

Let G be a interval+kv graph with interval deletion set X. Using an interval representation of $G' = G \setminus X$, we construct a path decomposition of G' whose set of bags $B_1, \ldots, B_r$ is the set of maximal cliques of G' (see e.g. [28]). As each bag of the path decomposition corresponds to the vertex set of a maximal clique in G', there is a vertex $v \in B_1 \setminus (B_2 \cup B_3 \cup \ldots B_r)$.

The algorithm considers all choices for the intersection of X with the clique from the clique cover containing v. Each such choice is a clique X_1 such that $N(v) \subseteq X_1 \subseteq X$. Given X_1 and v, the clique $c(X_1, v)$ of the clique cover containing $X_1 \cup \{v\}$ can be chosen greedily by the maximality argument mentioned above. Indeed, there is a unique maximal clique containing $X_1 \cup \{v\}$: we set $c(X_1, v) := X_1 \cup \{v\} \cup Y_1$, where $u \in Y_1$ if and only if $u \in B_1$ and $X_1 \subseteq N(u)$. Let $mcc(G)$ be the size of a minimum clique cover for G. Then $mcc(G) = 1 + \min\{mcc(G[V \setminus c(X_1, v)]) : X_1$ is a clique and $N(v) \subseteq X_1 \subseteq X\}$.

As the X_1 minimizing the above equation is one of the 2^k subsets of X we can conclude that clique cover is computed correctly in time $O(2^k \cdot n^{O(1)})$ and polynomial space. □

The *boolean-width* of graphs is a recently introduced graph parameter [14]. It will enable us to obtain FPT results for several problems. As interval graphs have boolean-width at most $\log n$ [9] and adding a vertex to a graph increases its boolean-width by at most 1, we have the following lemma.

Lemma 3. *Any interval+kv graph G has boolean width at most* $\log n + k$*, where n is the number of vertices of G.*

As several problems can be solved in time $2^{O(b)}n^{O(1)}$ on graphs with boolean-width b and n vertices [14], they are FPT on interval+kv graphs.

Corollary 1. Independent Set*,* Dominating Set*, their weighted and counting versions, and* Independent Dominating Set*, are* FPT *on interval+kv graphs.*

We also provide simple polynomial-space algorithms for Independent Set and Clique on interval+kv graphs and for Dominating Set on k-gap interval graphs in Appendix B.

4 W[1]-Hardness Result

A *coloring* of a graph $G = (V, E)$ is a mapping $c\colon V \to \{1, 2, \ldots\}$ such that $c(u) \neq c(v)$ whenever $uv \in E$. A *p-coloring* of G is a coloring c of G with $c(v) \in \{1, \ldots, p\}$ for $v \in V$. The p-Coloring problem asks for a graph G and a positive integer p, whether G has a p-coloring. The problem p-Precoloring Extension is to decide whether a given mapping $c\colon U \to \{1, \ldots, p\}$ defined on a (possibly empty) subset $U \subseteq V$ of *precolored* vertices can be extended to a p-coloring of G. We refer to these problems as Coloring and Precoloring Extention if p is assumed to be a part of the input.

First, we make the followig observation.

Proposition 1. *The parameterization of* Coloring *by $p + k$ is* FPT *on interval+kv graphs.*

Proof. We use a Win-Win approach. Let G be an interval+kv graph with interval deletion set X. If G has a clique of size $p + 1$, then it cannot be colored by p colors. By Theorem 6 it can be determined whether such a clique exists in time $2^k \cdot n^{O(1)}$. Otherwise, the interval graph $G \setminus X$ has pathwidth at most p [12]. Thus, $\mathbf{pw}(G) \leq p + k$. It remains to observe that p-Coloring is FPT on graphs of bounded pathwidth by Courcelle's Theorem [19]. □

However, the parameterization by k of this problem is W[1]-hard, even for k-gap interval graphs.

Theorem 5. Coloring, *parameterized by k, is* W*[1]-hard on k-gap interval graphs.*

Proof. We reduce from the Precoloring Extension problem. Marx [37] proved that Precoloring Extension is W[1]-hard on interval graphs, parameterized by the number of precolored vertices. Let $G = (V, E)$ be an interval graph with a set of precolored vertices $U \subseteq V$ and a precoloring $c\colon U \to \{1, \ldots, p\}$. Let $k = |U|$, and denote by $X_1, \ldots, X_p$ (some sets can be empty) the partition of U into the color classes induced by c. We construct the graph H as follows:

- construct a disjoint union of G and a complete graph K_p with the vertices $v_1, \ldots, v_p$;
- for each $i \in \{1, \ldots, p\}$, identify all the vertices of X_i and v_i.

By Observation 1, H is a k-gap interval graph. It remains to observe that H has a p-coloring if and only if c can be extended to a p-coloring of G. □

5 Conclusion

While multiple interval graphs have a large number of applications, many problems remain intractable on t-interval graphs, even for small constant t. On the other hand, the total number of gaps, k, in a multiple interval representation seems to be a more useful parameterization of problems on multiple interval graphs. Indeed, we have seen that this parameter captures some of the intractibility of graph problems and the parameterization by k of many problems turns out to be FPT.

While this first paper on the parameterization of graph problems by the total number of gaps classifies some important problems as FPT or W[1]-hard, it raises more questions than it answers. There is the question of investigating other problems that are polynomial time solvable on interval graphs but hard on t-interval graphs for small constant t. One example is Hamiltonian Cycle. Further considerations worth investigating are kernelization algorithms and improvements on the running time of our (rather simple) algorithms. The most important open problem for k-gap interval graphs is, in our eyes, to pinpoint the parameterized complexity of the recognition problem.

Acknowledgment. We thank Mathieu Chapelle for interesting discussions about this work.

References

1. N. Alon. Piercing d-intervals. *Discret. Comput. Geom.*, 19(3):333–334, 1998.
2. T. Andreae. On an extremal problem concerning the interval number of a graph. *Discrete Appl. Math.*, 14(1):1–9, 1986.
3. T. Andreae and M. Aigner. The total interval number of a graph. *J. Comb. Theory Ser. B*, 46(1):7–21, 1989.

4. Y. Aumann, M. Lewenstein, O. Melamud, R. Y. Pinter, and Z. Yakhini. Dotted interval graphs and high throughput genotyping. In *SODA 2005*, 339–348, 2005.
5. V. Bafna, B. O. Narayanan, and R. Ravi. Nonoverlapping local alignments (weighted independent sets of axis-parallel rectangles). *Discrete Appl. Math.*, 71(1-3):41–53, 1996.
6. J. Balogh, P. Ochem, and A. Pluhár. On the interval number of special graphs. *J. Graph Theor.*, 46(4):241–253, 2004.
7. R. Bar-Yehuda, M. M. Halldórsson, J. Naor, H. Shachnai, and I. Shapira. Scheduling split intervals. *SIAM J. Comput.*, 36(1):1–15, 2006.
8. R. Bar-Yehuda and D. Rawitz. Using fractional primal-dual to schedule split intervals with demands. *Discrete Optim.*, 3(4):275–287, 2006.
9. R. Belmonte and M. Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. In *WG 2011*, LNCS 6986, 2011. To appear.
10. C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, C.-G. Quimper, and T. Walsh. The parameterized complexity of global constraints. In *AAAI 2008*, 235–240, 2008.
11. G. Blin, G. Fertin, and S. Vialette. Extracting constrained 2-interval subsets in 2-interval sets. *Theor. Comput. Sci.*, 385(1-3):241–263, 2007.
12. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.
13. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. System Sci.*, 13(3):335–379, 1976.
14. B.-M. Bui-Xuan, J. A. Telle, and M. Vatshelle. Boolean-width of graphs. *Theor. Comput. Sci.*, 412(39):5187–5204, 2011.
15. A. Butman, D. Hermelin, M. Lewenstein, and D. Rawitz. Optimization problems in multiple-interval graphs. *ACM Trans. Algorithms*, 6(2), 2010.
16. P. A. Catlin. Supereulerian graphs: A survey. *J. Graph Theor.*, 16(2):177–196, 1992.
17. E. Chen, L. Yang, and H. Yuan. Improved algorithms for largest cardinality 2-interval pattern problem. *J. Comb. Optim.*, 13(3):263–275, 2007.
18. M. Chen and G. J. Chang. Total interval numbers of complete r-partite graphs. *Discrete Appl. Math.*, 122:83–92, 2002.
19. B. Courcelle. The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *Rairo - Theor. Inform. Appl.*, 26:257–286, 1992.
20. M. Crochemore, D. Hermelin, G. M. Landau, D. Rawitz, and S. Vialette. Approximating the 2-interval pattern problem. *Theor. Comput. Sci.*, 395(2-3):283–297, 2008.
21. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer, 1999.
22. P. Erdös and D. B. West. A note on the interval number of a graph. *Discrete Math.*, 55(2):129–133, 1985.
23. M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410:53–61, 2009.
24. J. Flum and M. Grohe. *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series XIV. Springer, 2006.
25. P. Gambette and S. Vialette. On restrictions of balanced 2-interval graphs. In *WG 2007*, LNCS 4769, 55–65, 2007.
26. S. Gaspers and S. Szeider. Kernels for global constraints. In *IJCAI 2011*, 540–545, 2011.
27. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Comb. Theory Ser. B*, 16(1):47–56, 1974.

28. M. C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, 1980.
29. J. R. Griggs and D. B. West. Extremal values of the interval number of a graph. *SIAM J. Algebra. Discr.*, 1(1):1–7, 1980.
30. R. Hassin and D. Segev. Rounding to an integral program. *Oper. Res. Lett.*, 36(3):321–326, 2008.
31. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Inform. Process. Lett.*, 27(3):119–123, 1988.
32. T. Kaiser. Transversals of d-intervals. *Discret. Comput. Geom.*, 18(2), 1997.
33. T. Kloks. *Treewidth, Computations and Approximations*, LNCS 842. Springer, 1994.
34. A. V. Kostochka and D. B. West. Total interval number for graphs with bounded degree. *J. Graph Theor.*, 25(1):79–84, 1997.
35. T. M. Kratzke and D. B. West. The total interval number of a graph, I: Fundamental classes. *Discrete Math.*, 118(1-3):145–156, 1993.
36. T. M. Kratzke and D. B. West. The total interval number of a graph II: Trees and complexity. *SIAM J. Discrete Math.*, 9(2):339–348, 1996.
37. D. Marx. Parameterized coloring problems on chordal graphs. *Theor. Comput. Sci.*, 351(3):407–424, 2006.
38. D. Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.
39. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.
40. G. Ramalingam and C. Pandu Rangan. A unified approach to domination problems on interval graphs. *Inform. Process. Lett.*, 27(5):271–274, 1988.
41. A. Raychaudhuri. The total interval number of a tree and the hamiltonian completion number of its line graph. *Inform. Process. Lett.*, 56(6):299–306, 1995.
42. D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976.
43. E. R. Scheinerman and D. B. West. The interval number of a planar graph: Three intervals suffice. *J. Comb. Theory Ser. B*, 35(3):224–239, 1983.
44. J. P. Spinrad. *Efficient Graph Representations*, Fields Institute Monographs 19. AMS, 2003.
45. G. Tardos. Transversals of 2-intervals, a topological approach. *Combinatorica*, 15(1):123–134, 1995.
46. W. T. Trotter and F. Harary. On double and multiple interval graphs. *J. Graph Theor.*, 3(3):205–2011, 1979.
47. S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.*, 6(3):505–517, 1977.
48. S. Vialette. On the computational complexity of 2-interval pattern matching problems. *Theor. Comput. Sci.*, 312(2-3):224–239, 2004.
49. S. Vialette. Two-interval pattern problems. In *Encyclopedia of Algorithms*. Springer, 2008.
50. D. B. West. A short proof of the degree bound for interval number. *Discrete Math.*, 73(3):309–310, 1989.
51. D. B. West and D. B. Shmoys. Recognizing graphs with fixed interval number is NP-complete. *Discrete Appl. Math.*, 8:295–305, 1984.

A Omitted Proofs

Lemma 2 ($\star$). *Let $\mathcal{Z} = \{B_1, \cdots, B_\ell\}$ be a set of partition constraints such that the sets S'_{B_j} are pairwise disjoint. It can be decided in $(|\mathcal{Z}| \cdot n)^{O(1)}$ time if there exists a valid ordering of the leaves of a PQ-tree T satisfying all constraints in $\mathcal{Z}$.*

Proof. For a node $w \in V(T)$, let T_w denote the subtree of T rooted at w and let L_w denote the set of leaves of T_w. We call a partition constraint $(i, \ldots, S)$ *relevant* at w if $1 \leq |S \cap L_w| < |S|$. We call a constraint *active* at w if it is relevant at w or at a child of w, and *inactive* otherwise. By preprocessing the tree in polynomial time, we can easily decide whether a given constraint is active for a vertex $w \in V(T)$ or not.

We fix a correct ordering bottom-up. Let w be an internal vertex of type P. First, observe that we only need to permute relevant children of w. Consider first all relevant children of w that have a leaf u^i_L or u^i_R for some i and some constraint as a descendant. We permute these relevant children so that they respect the ordering prescribed by the constraints. That is, u^i_L comes immediate before u^i_R and before $u^j_\cdot$ for $j > i$ and after $u^j_\cdot$ for all $j < i$ for each constraint in $\mathcal{Z}$. As a consequence a leaf u^i_R in T_w with the corresponding u^i_L not in T_w would have to be the first in the ordering, likewise we might find the last leaf. If it is not possible to find such an ordering, we answer No. Now consider all remaining relevant children. Note that all leaves that are a descendant of such a child must belong to some set S', or we can immediately answer No. Since the sets S' are disjoint, it is easy to permute them and place them properly within the ordering.

If w is an internal vertex of type Q, we apply the same idea, but there are only two possible orderings to check. If neither helps to satisfy the constraints, we simply answer No.

By applying this procedure in a bottom-up fashion, we can correctly decide whether we can satisfy all constraints. There are $O(n)$ nodes in the PQ-tree. The ordering of the children can be done in $O(|\mathcal{Z}| \cdot n^2)$ time. □

Theorem 1 ($\star$). *Given a graph G, one can decide whether $I(G) \leq n + k$ in polynomial time if k is a constant.*

Proof. As a first step, the algorithm guesses how many intervals are associated to each vertex of $G = (V, E)$, with a total of at most $n + k$ intervals. There are at most $O(n^k)$ such choices. Let $X \subseteq V$ denote the set of vertices to which more than one interval is associated. The algorithm verifies that $G \setminus X$ is an interval graph, otherwise it immediately moves to the next guess.

Then the algorithm enumerates all permutations of the set of all endpoints of intervals associated with vertices in X. There are at most $(4k)!$ such permutations, and each permutation corresponds to a multiple interval representation f. Next, verify that f is indeed a multiple interval representation for $G[X]$, otherwise move on to the next permutation.

Consider the set $\mathcal{K}'$ of maximal cliques of $G \setminus X$. Since $G \setminus X$ is an interval graph, $|\mathcal{K}'| \leq |V \setminus X|$ and $\mathcal{K}'$ can be found in polynomial time using a perfect

elimination order [42]. Construct a new set $\mathcal{K}$ of cliques, where $\mathcal{K}$ is obtained from $\mathcal{K}'$ by adding all pairwise intersections of cliques in $\mathcal{K}'$. Compute the set $\mathcal{G}$ of all maximal cliques of G. By Lemma 1, constructing $\mathcal{G}$ takes $2^k n^{O(1)}$ time using the polynomial-delay enumeration algorithm of [31].

Consider the multiple interval representation f that we have guessed before. A *display* of a clique C is a maximal open interval (a, b) such that (a, b) is contained in an interval of each vertex of C and is disjoint from the intervals of all other vertices. We say that a clique is *displayable* if it has a display. Let C be a displayable clique of $G[X]$. First we find $\mathcal{K}_C = \{K \cap (\bigcap_{v \in C} N(v)) \mid K \in \mathcal{K}'\}$. For each display of each displayable clique C, we guess its *outer cliques*, the two cliques of $\mathcal{K}_C$ that appear first and last under the display. Denote the multiset of chosen cliques by $\mathcal{K}_C^* = \{K_C^1, \bar{K}_C^1, \ldots, K_C^t, \bar{K}_C^t\}$, numbered in the order in which they were chosen, where $K_C^i, \bar{K}_C^i$ are the outer cliques of the i-th display. Since $\mathcal{I}$ has at most $4k$ displays, it takes $n^{O(k)}$ time in total to guess $\mathcal{K}_C^*$ for all displayable cliques C.

Let $\mathcal{K}^*$ denote the multiset that is the union of all the guessed $\mathcal{K}_C^*$, and let $\bar{\mathcal{K}}$ denote the multiset obtained after adding all cliques that are in $\{K \setminus X \mid K \in \mathcal{G}\}$ and not already in $\mathcal{K}^*$. Initialize a PQ-tree T with $\bar{\mathcal{K}}$ as ground set. Run the reduction algorithm of Booth and Lueker [13] on T with the set $\mathbb{S} = \{\{K \in \bar{\mathcal{K}} \mid v \in K\} \mid v \in V(G) \setminus X\}$. If it fails, continue to the next guess of outer cliques.

We build a set S_C which we will use for a partition constraint on the PQ-tree conform Lemma 2. Initially, S_C contains the outer cliques of C. Consider a maximal clique $K \in \mathcal{G}$. Suppose that $K \setminus X$ has a vertex v such that v does not appear in any outer clique of $C = K \cap X$. Since $v \in K$, v is a neighbor of every vertex in C. Hence the interval of v must be a strict subinterval of a display of C. Therefore we add $K \setminus X$ to S_C. If no such vertex v exists all vertices of $K \setminus X$ appear in the union of the outer cliques of C. As K is maximal, $K \setminus X$ is not a strict subset of any single outer clique. All vertices from $K \setminus X$ appear in the union of two outer cliques that are consecutive in the order of their endpoints in f, for otherwise an outer clique separates two vertices from $K \setminus X$ and hence contradicts that $K \setminus X$ is a clique. Suppose these consecutive outer cliques are $\bar{K}_C^i, K_C^{i+1}$ for some i. We may assume that $K \setminus X$ contains a vertex from $\bar{K}_C^i \setminus K_C^{i+1}$ and a vertex from $K_C^{i+1} \setminus \bar{K}_C^i$, because otherwise $K \setminus X \subseteq \bar{K}_C^i$ or $K \setminus X \subseteq K_C^{i+1}$. But then $K \setminus X$ appears between $\bar{K}_C^i$ and K_C^{i+1}, and is not added it to S_C. Suppose instead that the two outer cliques are $K_C^i, \bar{K}_C^i$. Using similar reasoning, we can then show that $K \setminus X$ appears between K_C^i and $\bar{K}_C^i$, and thus must be added to S_C. This finishes the construction of S_C.

For each displayable clique C, we have a set of cliques S_C that should appear under a display of C. Use Lemma 2 to partition them into sets S_C^i, where S_C^i appears between $K_C^i, \bar{K}_C^i$. Run the reduction algorithm of Booth and Lueker on T with the set $\mathbb{S}$, which is the union of all $\mathbb{S}_C = \{S_C^i, S_C^i \cup \{K_C^i\}, S_C^i \cup \{\bar{K}_C^i\} \mid 1 \leq i \leq t\}$ taken over all displayable cliques C.

Assuming all the above operations succeed, the PQ-tree represents admissible permutations of the maximal cliques that yield a multiple interval representation of G with k gaps. We obtain this multiple interval representation by going from

left to right through the leafs of the PQ-tree. When going from one clique K to the next K', we close all intervals for vertices in $K \setminus K'$ and we open all intervals for vertices in $K' \setminus K$.

This concludes the proof. □

B Polynomial-Space Algorithms for Independent Set, Clique, and Dominating Set

An *independent set* in a graph G is a set of vertices that are all pairwise non-adjacent in G. The Independent Set (Clique) problem has as input a graph G and a positive integer p, and the question is whether G has an independent set (clique) of size p.

Theorem 6. Independent Set *and* Clique *can be solved in time* $2^k \cdot n^{O(1)}$ *and polynomial space on interval+kv graphs, where* n *is the number of vertices.*

Proof. Let $G = (V, E)$ be the input graph with interval deletion set X.

To check whether G has an independent set of size p, the algorithm goes over all subsets $Y \subseteq X$, and checks whether Y is independent in G. If so, it checks whether $G \setminus (X \cup N[Y])$ has an independent set of size $p - |Y|$. The last check can be done in linear time [27], as $G \setminus (X \cup N[Y])$ is an interval graph.

To check whether G has a clique of size p, the algorithm uses a polynomial-delay polynomial-space algorithm enumerating all maximal cliques of G [47], and checks whether at least one such maximal clique has size at least p. As G has $O(2^k n)$ maximal cliques by Lemma 1, the running time follows. □

A vertex subset D is a *dominating set* in a graph $G = (V, E)$ if every vertex from $V \setminus D$ has a neighbor in D. The Dominating Set problem has as input a graph G and a positive integer p, and the question is whether G has a dominating set of size p.

Theorem 7. Dominating Set *can be solved in time* $O(3^k \cdot n^{O(1)})$ *and polynomial space on* k*-gap interval graphs.*

Proof. Let $G = (V, E)$ be the input graph, let f be a multiple interval representation of G with k gaps, and let $X = \mathsf{gap}_f(G)$.

The algorithm goes over all partitions (Y, Z) of X. For each such partition it will consider dominating sets containing Y. Each vertex $z \in Z$ needs to be dominated by a vertex that has is associated to an interval intersecting at least one interval of z. For each vertex $z \in Z$, the algorithm considers all possibilities of choosing exactly one interval $[l_z, r_z]$ from $f(z)$ by which it is to be dominated. It remains to check whether G has a dominating set D of size p such that $Y \subseteq D$ and such that each $[l_z, r_z]$ intersects at least one interval of a vertex from D, for each $z \in Z \setminus D$. This is done by a polynomial-time algorithm which solves a version of Dominating Set on interval graphs where some vertices do not need to be dominated [40]. Namely, we start from the interval model

$\{f(v) : v \in V \setminus X\} \cup \{[l_z, r_z] : z \in Z\}$, we mark the intervals associated to vertices from $N(Y)$ and check whether this interval graph has $p - |Y|$ vertices dominating every vertex in $V \setminus N[Y]$ by the algorithm from [40].

As choosing an interval for each vertex from Y can be reduced to deciding, for each gap, whether the interval is to the left or to the right of this gap, the total running time is within a polynomial factor of $\sum_{Y \subseteq X} \prod_{z \in X \setminus Y} 2^{|f(z)|-1} = O(3^k)$. □