

k -Gap Interval Graphs*

Fedor V. Fomin¹, Serge Gaspers², Petr Golovach³, Karol Suchan^{4,5},
Stefan Szeider², Erik Jan van Leeuwen¹,
Martin Vatshelle¹, and Yngve Villanger¹

¹ Department of Informatics, University of Bergen, Bergen, Norway
{fedor.fomin,e.j.van.leeuwen,martin.vatshelle,yngve.villanger}@ii.uib.no,
{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 Faculty of Applied MathematicsWMS,
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

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

* Serge Gaspers and Stefan Szeider acknowledge support from the European Research Council (COMPLEX REASON, 239962). Petr Golovach acknowledges the support by EPSRC (EP/G043434/1), Royal Society (JP100692). Karol Suchan acknowledges support from Conicyt Chile (Anillo ACT-88, Basal-CMM, Fondecyt 11090390).

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 [47], 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 [44]. Unfortunately, many problems remain NP-hard on 2-interval graphs (for example, their recognition [52], 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, 45]). 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 $\text{interval}+kv$.

Definition 3. *A graph $G = (V, E)$ is an $\text{interval}+kv$ 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 [40]. As the set of vertices that are associated to more than 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, 44, 47, 51] and references), classical complexity [52], and parameterized complexity [11, 23, 32, 33]. Applications have been identified in scheduling and resource allocation [7, 8], communication protocols [15], computational biology [4, 5, 11, 17, 20, 26, 49, 50], and monitoring [15]. The total interval number was introduced in [29] and studied in [3, 16, 18, 36–38, 42].

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 $\text{gap}_f(G)$ 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 $\text{tw}(G)$) 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 $\text{pw}(G)$).

We refer to [21, 24, 41] for texts on parameterized complexity, the theoretical framework of our investigations. Proofs of statements marked with (\star) can be found in [25].

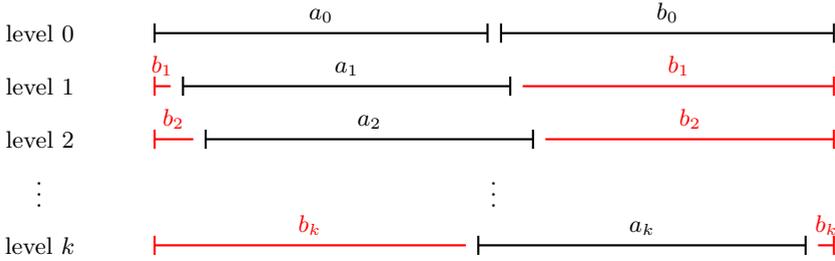


Fig. 1. A multiple interval representation with k gaps of a graph with 2^{k+1} maximal cliques

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 [38]. 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.

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 (\star). *An interval+ kv graph on n vertices has at most $2^k \cdot (n - k)$ maximal cliques.*

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 $V = \{a_0, \dots, a_k, b_0, \dots, b_k\}$ 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 $B = (i, u_L^1, u_R^1, \dots, u_L^i, u_R^i, S)$, where $\{u_L^1, u_R^1, \dots, u_L^i, u_R^i\} \subseteq S \subseteq U$, if S can be partitioned into S_1, \dots, 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 $S \setminus \{u_L^1, u_R^1, \dots, u_L^i, u_R^i\}$.

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

Lemma 2 (\star). *Let $\mathcal{Z} = \{B_1, \dots, B_\ell\}$ 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 refer to [25] 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 an interval graph we can find all the maximal cliques in polynomial time using a perfect elimination order [43]. We also find all maximal cliques of G using Lemma 1 and a polynomial delay enumeration algorithm [48].

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, 34, 46]). 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 constraints has as input a set of variables $X = \{x_1, \dots, x_{n'}\}$, a totally ordered set of values D , a map $dom : X \rightarrow 2^D$ 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 $(u, w) \in \text{dom}(x) \times \text{dom}(x)$, such that there is a value $v \in D \setminus \text{dom}(x)$ with $u < v < w$ and there is no value $v' \in \text{dom}(x)$ with $u < v' < w$. The problem has a kernel with $O(k'^2)$ variables and domain values and can be solved in time $O(1.6181^{k'} k'^2 + n' + |D|)$ [27].

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 constraints. 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 $X := V$, $D := F$, $\text{dom}(x) := \bigcup_{I \in f(x)} I \cap F$, 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 [35] 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 $1, \dots, r$ 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 $1, \dots, r$. Set $B'_0 := \emptyset$. We construct a path decomposition of G with bags B_0, \dots, B_r where $B_i = B'_i \cup X$ for $i \in \{0, \dots, r\}$. Now we apply a dynamic programming algorithm over this path decomposition.

We first describe what is stored in the tables corresponding to the nodes $0, \dots, r$ of the path. For any $i \in \{0, \dots, r\}$, we denote by G_i the subgraph of G induced by $\bigcup_{j=0}^i B_j$. For $i \in \{0, \dots, r\}$, 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 $i \in \{0, \dots, r\}$, 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, \dots, 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 $G[Z_i \cup Z_{i+1} \cup \dots \cup Z_t]$. 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, \dots, 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 $v \in B_1 \setminus (B_2 \cup B_3 \cup \dots \cup B_r)$.

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

$mcc(G) = 1 + \min\{mcc(G[V \setminus c(X_1, v)]) : X_1 \text{ is a clique and } N(v) \subseteq X_1 \subseteq X\}$. 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. \square

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 [25].

4 W[1]-Hardness Result

A *coloring* of a graph $G = (V, E)$ is a mapping $c: V \rightarrow \{1, 2, \dots\}$ such that $c(u) \neq c(v)$ whenever $uv \in E$. A *p -coloring* of G is a coloring c of G with $c(v) \in \{1, \dots, p\}$ 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 $c: U \rightarrow \{1, \dots, p\}$ 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 EXTENSION if p is assumed to be a part of the input.

First, we make the following observation that was independently made in [31].

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 the CLIQUE algorithm from [25] 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]. \square

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 [39] 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 $c: U \rightarrow \{1, \dots, p\}$. Let $k = |U|$, and denote by X_1, \dots, 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, \dots, v_p ;
- for each $i \in \{1, \dots, p\}$, 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 intractability 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. Alon, N.: Piercing *d*-intervals. *Discret. Comput. Geom.* 19(3), 333–334 (1998)
2. Andreae, T.: On an extremal problem concerning the interval number of a graph. *Discrete Appl. Math.* 14(1), 1–9 (1986)
3. Andreae, T., Aigner, M.: The total interval number of a graph. *J. Comb. Theory Ser. B* 46(1), 7–21 (1989)
4. Aumann, Y., Lewenstein, M., Melamud, O., Pinter, R.Y., Yakhini, Z.: Dotted interval graphs and high throughput genotyping. In: *SODA 2005*, pp. 339–348 (2005)

5. Bafna, V., Narayanan, B.O., Ravi, R.: Nonoverlapping local alignments (weighted independent sets of axis-parallel rectangles). *Discrete Appl. Math.* 71(1-3), 41–53 (1996)
6. Balogh, J., Ochem, P., Pluhár, A.: On the interval number of special graphs. *J. Graph Theor.* 46(4), 241–253 (2004)
7. Bar-Yehuda, R., Halldórsson, M.M., Naor, J., Shachnai, H., Shapira, I.: Scheduling split intervals. *SIAM J. Comput.* 36(1), 1–15 (2006)
8. Bar-Yehuda, R., Rawitz, D.: Using fractional primal-dual to schedule split intervals with demands. *Discrete Optim.* 3(4), 275–287 (2006)
9. Belmonte, R., Vatshelle, M.: Graph Classes with Structured Neighborhoods and Algorithmic Applications. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 47–58. Springer, Heidelberg (2011)
10. Bessièrè, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C.-G., Walsh, T.: The parameterized complexity of global constraints. In: AAAI 2008, pp. 235–240 (2008)
11. Blin, G., Fertin, G., Vialette, S.: Extracting constrained 2-interval subsets in 2-interval sets. *Theor. Comput. Sci.* 385(1-3), 241–263 (2007)
12. Bodlaender, H.L.: A partial k -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.* 209(1-2), 1–45 (1998)
13. Booth, K.S., Lueker, G.S.: 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. Bui-Xuan, B.-M., Telle, J.A., Vatshelle, M.: Boolean-width of graphs. *Theor. Comput. Sci.* 412(39), 5187–5204 (2011)
15. Butman, A., Hermelin, D., Lewenstein, M., Rawitz, D.: Optimization problems in multiple-interval graphs. *ACM Trans. Algorithms* 6(2) (2010)
16. Catlin, P.A.: Supereulerian graphs: A survey. *J. Graph Theor.* 16(2), 177–196 (1992)
17. Chen, E., Yang, L., Yuan, H.: Improved algorithms for largest cardinality 2-interval pattern problem. *J. Comb. Optim.* 13(3), 263–275 (2007)
18. Chen, M., Chang, G.J.: Total interval numbers of complete r -partite graphs. *Discrete Appl. Math.* 122, 83–92 (2002)
19. Courcelle, B.: The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *Rairo - Theor. Inform. Appl.* 26, 257–286 (1992)
20. Crochemore, M., Hermelin, D., Landau, G.M., Rawitz, D., Vialette, S.: Approximating the 2-interval pattern problem. *Theor. Comput. Sci.* 395(2-3), 283–297 (2008)
21. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer (1999)
22. Erdős, P., West, D.B.: A note on the interval number of a graph. *Discrete Math.* 55(2), 129–133 (1985)
23. Fellows, M.R., Hermelin, D., Rosamond, F., Vialette, S.: On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.* 410, 53–61 (2009)
24. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series XIV. Springer (2006)
25. Fomin, F.V., Gaspers, S., Golovach, P., Suchan, K., Szeider, S., van Leeuwen, E.J., Vatshelle, M., Villanger, Y.: k -gap interval graphs. arXiv CoRR 1112.3244 (2011)
26. Gambette, P., Vialette, S.: On Restrictions of Balanced 2-Interval Graphs. In: Brandstädt, A., Kratsch, D., Müller, H. (eds.) WG 2007. LNCS, vol. 4769, pp. 55–65. Springer, Heidelberg (2007)
27. Gaspers, S., Szeider, S.: Kernels for global constraints. In: IJCAI 2011, pp. 540–545 (2011)
28. Golumbic, M.C.: Algorithmic graph theory and perfect graphs. Academic Press (1980)

29. Griggs, J.R., West, D.B.: Extremal values of the interval number of a graph. *SIAM J. Algebra. Discr.* 1(1), 1–7 (1980)
30. Hassin, R., Segev, D.: Rounding to an integral program. *Oper. Res. Lett.* 36(3), 321–326 (2008)
31. Jansen, B.M.P., Kratsch, S.: Data Reduction for Graph Coloring Problems. In: Owe, O., Steffen, M., Telle, J.A. (eds.) *FCT 2011. LNCS*, vol. 6914, pp. 90–101. Springer, Heidelberg (2011)
32. Jiang, M., Zhang, Y.: Parameterized Complexity in Multiple-interval Graphs: Domination. In: Rossmanith, P. (ed.) *IPEC 2011. LNCS*, vol. 7112, pp. 27–40. Springer, Heidelberg (2012)
33. Jiang, M., Zhang, Y.: Parameterized Complexity in Multiple-Interval Graphs: Partition, Separation, Irredundancy. In: Fu, B., Du, D.-Z. (eds.) *COCOON 2011. LNCS*, vol. 6842, pp. 62–73. Springer, Heidelberg (2011)
34. Kaiser, T.: Transversals of d-intervals. *Discret. Comput. Geom.* 18(2) (1997)
35. Kloks, T.: *Treewidth, Computations and Approximations. LNCS*, vol. 842. Springer, Heidelberg (1994)
36. Kostochka, A.V., West, D.B.: Total interval number for graphs with bounded degree. *J. Graph Theor.* 25(1), 79–84 (1997)
37. Kratzke, T.M., West, D.B.: The total interval number of a graph, I: Fundamental classes. *Discrete Math.* 118(1-3), 145–156 (1993)
38. Kratzke, T.M., West, D.B.: The total interval number of a graph II: Trees and complexity. *SIAM J. Discrete Math.* 9(2), 339–348 (1996)
39. Marx, D.: Parameterized coloring problems on chordal graphs. *Theor. Comput. Sci.* 351(3), 407–424 (2006)
40. Marx, D.: Chordal deletion is fixed-parameter tractable. *Algorithmica* 57(4), 747–768 (2010)
41. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications.* Oxford University Press (2006)
42. Raychaudhuri, A.: The total interval number of a tree and the hamiltonian completion number of its line graph. *Inform. Process. Lett.* 56(6), 299–306 (1995)
43. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.* 5(2), 266–283 (1976)
44. Scheinerman, E.R., West, D.B.: The interval number of a planar graph: Three intervals suffice. *J. Comb. Theory Ser. B* 35(3), 224–239 (1983)
45. Spinrad, J.P.: *Efficient Graph Representations. Fields Institute Monographs*, vol. 19. AMS (2003)
46. Tardos, G.: Transversals of 2-intervals, a topological approach. *Combinatorica* 15(1), 123–134 (1995)
47. Trotter, W.T., Harary, F.: On double and multiple interval graphs. *J. Graph Theor.* 3(3), 205–211 (1979)
48. Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I.: A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.* 6(3), 505–517 (1977)
49. Vialette, S.: On the computational complexity of 2-interval pattern matching problems. *Theor. Comput. Sci.* 312(2-3), 224–239 (2004)
50. Vialette, S.: Two-interval pattern problems. In: *Encyclopedia of Algorithms.* Springer (2008)
51. West, D.B.: A short proof of the degree bound for interval number. *Discrete Math.* 73(3), 309–310 (1989)
52. West, D.B., Shmoys, D.B.: Recognizing graphs with fixed interval number is NP-complete. *Discrete Appl. Math.* 8, 295–305 (1984)