Theory Comput Syst (2016) 58:273–286
DOI 10.1007/s00224-015-9616-6

Exact Algorithms for Intervalizing Coloured Graphs

Hans L. Bodlaender · Johan M. M. van Rooij

Received: 20 October 2014 / Accepted: 23 February 2015 / Published online: 15 March 2015

Abstract In the INTERVALIZING COLOURED GRAPHS problem, one must decide for a given graph $G = (V, E)$ with a proper vertex colouring of G whether G is the subgraph of a properly coloured interval graph. For the case that the number of colors is fixed, we give an exact algorithm that uses $2^{\mathcal{O}(n/\log n)}$ time. We also give an $\mathcal{O}^*(2^n)$ algorithm for the case that the number of colors is not fixed.

Keywords Graph algorithms · Exact algorithms · Interval graphs · Subexponential time · Intervalizing coloured graphs · Pathwidth

1 Introduction

The area of exact algorithms for NP-hard problems is an old area in the field of design and analysis of algorithms, but also one with many important new

The research of this author was partially funded by the Networks programme, funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research.

H. L. Bodlaender (✉) · J. M. M. van Rooij
Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, Netherlands
e-mail: h.l.bodlaender@uu.nl

J. M. M. van Rooij
e-mail: jmmrooij@cs.uu.nl

H. L. Bodlaender
Department of Mathematics and Computer Science, University of Technology Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands

J. M. M. van Rooij
Consultants in Qualitative Methods, P.O. Box 414, 5600 AK Eindhoven, Netherlands

developments. A recent overview of the area can be found in the book by Fomin and Kratsch [14]. In this paper, we consider exact algorithms for the problem called INTERVALIZING COLOURED GRAPHS. This problem is defined in the following way. Given a graph $G = (V, E)$ together with a proper vertex colouring $c : V \to \{1, \ldots, k\}$ of G (a colouring c is proper if for all edges $\{v, w\} \in E$: $c(v) \neq c(w)$), one must decide if G is subgraph of a properly coloured interval graph, i.e., can we add edges, such that each edge is between vertices of different colors and the result is an interval graph? The problem has its original motivation in DNA physical mapping [13].

This problem is NP-complete [13] (see also [18]), even when the number of colors equals four [5, 6], and in addition, inputs are restricted to caterpillar trees [1]. We denote the version of the problem where the number k of colors is fixed by INTERVALIZING k-COLOURED GRAPHS, and the version with a potentially unbounded number of colors by INTERVALIZING COLOURED GRAPHS.

If the number of colors equals two, the problem is trivially solvable in linear time. For three colors, the problem is solvable in quadratic time with a complicated algorithm [7]; the case for three colors and biconnected graphs is described in [6].

In this paper, we give two algorithms: one for the case that the number of colors is bounded by a constant, and one for the case that the number of colors is unbounded. The former result is relevant if the number of colors is a constant that is at least four, and forms a somewhat curious exception to a general pattern that can be observed in most exact algorithms for graph problems. Namely, the algorithm uses slightly less than exponential time. Most NP-hard problems that have subexponential algorithms deal with planar graphs and generalizations of planar graphs, see e.g., [12, 16, 23]. Typically, the running time of such algorithms is of the form $2^{\mathcal{O}(\sqrt{n})}$. Our result is unlike most results in two ways: first, our inputs are general graphs (but a positive answer implies bounded pathwidth of the input; see Proposition 3 and its discussion), and secondly, the running time is 'just subexponential': it is bounded by $2^{\mathcal{O}(n/\log n)}$.

Our algorithm for INTERVALIZING k-COLOURED GRAPHS for fixed k can be viewed as a dynamic programming algorithm in Held-Karp style [21], resembling algorithms for some graph layout problems given e.g., in [8], but with one additional improvement: an isomorphism test on certain parts of the graph during the dynamic programming. Important concepts that facilitate the presentation of our results are the notions of *path decomposition* and *nice path decomposition*.

Our second result is an algorithm for INTERVALIZING COLOURED GRAPHS (with no bound on the number of colors); this algorithm runs in $\mathcal{O}^*(2^n)$. It is a rather simple dynamic programming algorithm, also in Held-Karp style, and is the first exact algorithm for the problem.

This paper is organized as follows. In Section 2, preliminary definitions and results are given. In Section 3, the notion of *partial path decomposition* and some related notions are introduced, and a few structural results on these notions are derived. In Section 4, we give the algorithm for INTERVALIZING k-COLOURED GRAPHS, and analyse its running time. In Section 5, the algorithm for INTERVALIZING COLOURED GRAPHS is given. Some final remarks are made in Section 6.

2 Preliminaries

In this section, we introduce some standard notations, and give a few preliminary results on path decompositions.

The graphs in this paper are considered to be undirected and simple. We make a distinction between labelled and unlabelled graphs. In a labelled graph, each vertex has a unique label, and two isomorphic graphs with different labels are considered to be different. In contrast, two isomorphic unlabelled graphs are considered to be the same object. We also consider graphs given with a vertex colouring. A vertex colouring of a graph $G = (V, E)$ is a function $c : V \to C$, for some finite set C. $|C|$ is the number of colours. A vertex colouring c is *proper*, if for all edges $\{v, w\} \in E$, we have that $c(v) \neq c(w)$.

Throughout the paper, we assume that the input graph G is connected; if not, we can run the algorithm separately on each connected component of G.

For a graph $G = (V, E)$ and a set of vertices $W \subseteq V$, we denote $G[W]$ as the subgraph induced by W: $G[W] = (W, \{\{v, w\} \in E \mid v, w \in W\})$.

A graph $G = (V, E)$ is an *interval graph*, if we can associate to each vertex $v \in V$ an interval on the real line $I_v = [\ell_v, r_v]$, such that for all $v, w \in V,\ v \neq w$: $\{v, w\} \in E$, if and only if $I_v \cap I_w \neq \emptyset$.

A graph $H = (V, F)$ is an *interval completion* of a graph $G = (V, E)$, if G and H have the same vertex set, $E \subseteq F$, and H is an interval graph. More background can be found in [17]; see also [20].

A *path decomposition* of a graph $G = (V, E)$ is a sequence of subsets of V called *bags*, $(X_1, X_2, \ldots, X_r)$ such that:

- $\bigcup_{1\leq i\leq r} X_i = V$;
- for all $\{v, w\} \in E$, there is an $i,\ v, w \in X_i$;
- for all $i_0,\ i_1,\ i_2$: if $1 \leq i_0 \leq i_1 \leq i_2 \leq r$, then $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$.

The *width* of a path decomposition $(X_1, X_2, \ldots, X_r)$ is $\max_{1\leq i\leq r} |X_i| - 1$. The *pathwidth* of a graph G is the minimum width of a path decomposition of G.

A path decomposition $(X_1, X_2, \ldots, X_r)$ is *nice*, if for all $i,\ 1 \leq i < r$, exactly one of the following two cases holds:

- There is a vertex $v \in V$ with $X_{i+1} = X_i \cup \{v\}$. We call X_{i+1} an *introduce* node.
- There is a vertex $v \in V$ with $X_{i+1} = X_i - \{v\}$. We call X_{i+1} a *forget* node.

If $|X_1| = 1$, we also call X_1 an introduce node. The following proposition is well known. We give the proof for later reference.

Proposition 1 (Folklore) *Each graph $G = (V, E)$ with pathwidth k has a nice path decomposition of width k with $2n$ bags, with $|X_1| = 1$, and $X_r = \emptyset$.*

Proof Suppose we have a path decomposition $(X_1, X_2, \ldots, X_r)$. We can turn it in a nice path decomposition as follows. First, remove all bags that are empty. If for some $i,\ 1 \leq i < r,\ i + 1$ is not an introduce or forget bag, then we insert some new bags between i and $i + 1$: first forget nodes, one for each vertex in $X_i - X_{i+1}$, and then we have one introduce node for each vertex in $X_{i+1} - X_i$. Similarly, we add introduce nodes before X_1 when $|X_1| \neq 1$, and add forget nodes at the end of the procedure till $X_r = \emptyset$. We have one introduce and one forget node per vertex, so we have $2n$ bags. □

Proposition 2 *There are at most $(k + 2^k + 1)^{2n-1}$ unlabelled graphs with pathwidth at most k that are pairwise non isomorphic.*

Proof Consider a nice path decomposition of a graph with n vertices, with $|X_1| = 1$, and with $2n$ bags. For each of the bags X_i, $i > 1$, there are at most $k + 2^k + 1$ possibilities: we can have a forget node, where we have the choice which of the at most $k + 1$ vertices in X_i we forget, or we can have an introduce node, where we have the choice to which of the at most k vertices in X_i the introduced vertex has an edge, i.e., at most 2^k choices for an introduce node. If we have two graphs with two path decompositions that we can construct while always making the same choices, then these graphs are isomorphic. □

Proposition 3 *Let $G = (V, E)$ be a graph with proper vertex colouring $c : V \to \{1, 2, \ldots, k\}$. The following are equivalent.*

1. *G has a properly coloured interval completion, using colouring c.*
2. *G has a path decomposition $(X_1, X_2, \ldots, X_r)$, such that for all $v, w \in V$: if $v \neq w$ and there is an i with $v, w \in X_i$, then $c(v) \neq c(w)$.*
3. *G has a nice path decomposition $(X_1, X_2, \ldots, X_{2|V|})$ of width at most $k - 1$, such that for all $v, w \in V$: if $v \neq w$ and there is an i with $v, w \in X_i$, then $c(v) \neq c(w)$.*

This proposition is well known. We sketch some of the proofs, as they provide some intuition for several of our later notions and results. Suppose $H = (V, F)$ is a properly coloured interval completion of G. Consider the interval model of H. We can assume that all endpoints of intervals are different in this model. Use a scanline that moves in this model from left to right. While moving the scanline, we build a nice path decomposition with the required property. We start with an empty bag. Each time the scanline meets a left or right endpoint, we add a new node to the path decomposition. If we meet a right endpoint of an interval representing a vertex v, we have a forget node, where we forget v. If we meet a left endpoint of an interval representing a vertex v, we have an introduce node, where we introduce the vertex v. In this way, the vertices in a bag are exactly the vertices represented by the intervals that intersect the scanline. One easily verifies that we have obtained a nice path decomposition of H, and, as G is a subgraph of H, this is also a nice path decomposition of G. As H is properly coloured, vertices in the same bag are differently coloured.

For the reverse direction, suppose that we have a (nice) path decomposition $(X_1, X_2, \ldots, X_r)$ from Proposition 3 (ii) or (iii), one obtains the corresponding interval graph by making each X_i a clique. The corresponding interval graph model is obtained by taking for a vertex v the interval $[\min_{v \in X_i} i, \max_{v \in X_i} i]$. As all colors in a bag X_i are different, the width of the path decompositions is bounded by $k - 1$.

Note that it follows directly from Proposition 3, that a graph with a proper vertex colouring with k colors has pathwidth at most $k-1$.

Proposition 3 motivates the definition of a *properly coloured path decomposition*: $(X_1, \ldots, X_r)$ is a properly coloured path decomposition of G, if and only if it is a path decomposition of G, and for all $v, w \in V$, if $v \neq w$ and there is an i with $v, w \in X_i$, then $c(v) \neq c(w)$.

3 Partial Path Decompositions

In this section, we introduce a number of notions that will be used for our dynamic programming algorithm in the next section.

A *partial path decomposition* of a graph $G = (V, E)$ is a sequence of subsets of V $(X_1, X_2, \ldots, X_s)$ such that:

- $(X_1, X_2, \ldots, X_s)$ is a path decomposition of $G[\bigcup_{1\leq i\leq s}, and X_i]$
- For each connected component of $G[V - X_s]$ with vertex set W, either $W \subseteq \bigcup_{1\leq i\leq s-1} X_i$ or $W \cap \left(\bigcup_{1\leq i\leq s-1} X_i\right) = \emptyset$.

The following proposition follows from well known facts about path and tree decompositions. For completeness we give the proof.

Proposition 4 *Let $(X_1, X_2, \ldots, X_r)$ be a path decomposition of G. Then, for each s, $1 \leq s \leq r$, $(X_1, X_2, \ldots, X_s)$ is a partial path decomposition of G.*

Proof Suppose $(X_1, \ldots, X_s)$ is not a partial path decomposition. Write $A = \bigcup_{1\leq i\leq s-1} X_i$ and $B = \bigcup_{s+1\leq i\leq r} X_i$. There is a connected component of $G[V - X_s]$ with vertex set W, such that there is a $v \in W \cap A$ and a $w \in W \cap B$. There is a path from v to w with all vertices in W, i.e., it avoids X_s. This path contains two neighbouring vertices $x \in A$ and $y \in B$. There is a bag X_i with $\{x, y\} \subseteq X_i$. If $i \leq s$, then $y \in X_s$ by the definition of path decomposition; if $i > s$, then $x \in X_s$ by definition of path decomposition. In both cases we have a contradiction. □

Consider a partial path decomposition $(X_1, X_2, \ldots, X_r)$ and a vertex set X. Later, X will typically be the set X_r for some partial path decomposition $(X_1, X_2, \ldots, X_r)$. A *component* of X is a vertex set that forms a connected component of the graph $G[V - X]$. We say that two components Y and Z of X are *isomorphic components* of X, if there is a graph isomorphism f of $G[Y \cup X]$ to $G[Z \cup X]$ that preserves colors and is the identity when restricted to X, i.e., f is a bijective function, such that the following conditions hold:

1. For all $v, w \in Y \cup X$: $\{v, w\} \in E \Leftrightarrow \{f(v), f(w)\} \in E$.
2. For all $v \in Y \cup X$: $c(v) = c(f(v))$.
3. For all $v \in X$: $f(v) = v$.

A component W of X_r is said to be a *left component* of the partial path decomposition $(X_1, \ldots, X_r)$, if $W \subseteq \bigcup_{1 \le i \le r-1} X_i$, and a *right component* of $(X_1, \ldots, X_r)$, if $W \cap \left(\bigcup_{1 \le i \le r-1} X_i\right) = \emptyset$.

The following proposition follows directly from the definition of partial path decomposition, see also Proposition 4.

Proposition 5 *Let $(X_1, X_2, \ldots, X_r)$ be a partial path decomposition of G. Each component of X_r is either a left or a right component of $(X_1, X_2, \ldots, X_r)$.*

We say that a partial path decomposition $(X_1, X_2, \ldots, X_s)$ of $G = (V, E)$ is *properly coloured*, if for all $v, w \in V$, if $v \neq w$ and there exists an i with $v, w \in X_i$, then $c(v) \neq c(w)$. We say that a (partial) path decomposition $(Y_1, Y_2, \ldots, Y_s)$ is an *extension* of a partial path decomposition $(X_1, X_2, \ldots, X_r)$ if $r \le s$ and for all i, $1 \le i \le r$, $Y_i = X_i$, i.e., $(X_1, \ldots, X_r, Y_{r+1}, \ldots, Y_s)$ extends $(X_1, \ldots X_r)$.

We define an equivalence relation on partial path decompositions as follows. We say that the partial path decomposition $(X_1, X_2, \ldots, X_r)$ is *equivalent* to the partial path decomposition $(Y_1, Y_2, \ldots, Y_s)$, if the following two conditions hold:

1. $X_r = Y_s$.
2. Suppose $W_1, W_2, \ldots, W_q$ are the components of X_r, i.e., the connected components of $G[V - X_r]$. There is a bijective function $g : \{1, \ldots, q\} \to \{1, \ldots, q\}$, such that for all i, $1 \le i \le q$:

   - W_i is a left component of $(X_1, X_2, \ldots, X_r)$, if and only if $W_{g(i)}$ is a left component of $(Y_1, Y_2, \ldots, Y_s)$ and
   - W_i and $W_{g(i)}$ are isomorphic components of X_r.

The main insight behind our dynamic programming algorithm is the following result.

Proposition 6 *If $(X_1, X_2, \ldots, X_r)$ and $(Y_1, Y_2, \ldots, Y_s)$ are equivalent coloured partial path decompositions, then $(X_1, X_2, \ldots, X_r)$ has an extension that is a properly coloured path decomposition of G, if and only if $(Y_1, Y_2, \ldots, Y_s)$ has an extension that is a properly coloured path decomposition of G.*

Proof Suppose $(X_1, X_2, \ldots, X_r, Z_1, Z_2, \ldots, Z_{r'})$ is a properly coloured path decomposition of G that is an extension of $(X_1, X_2, \ldots, X_r)$. Suppose that $W_1, \ldots, W_q$ are the components of X_r. Let g be the bijective function as in the definition of equivalence. Let f_i be a color preserving graph isomorphism from $G[X_r \cup W_i]$ to $G[X_r \cup W_{g(i)}]$ that is the identity on X_r, as implied by the definition of equivalence.

Let $f : V \to V$ be the function defined in the following way.

- for $v \in W_i,\ 1 \le i \le r$: $f(v) = f_i(v)$;
- for $v \in X_r,\ f(v) = v$.

Claim 7 *f is a color preserving automorphism of G.*

Proof f is a color preserving bijection: if $v \in X_r$, then v does not belong to a component W_i, so $f^{-1}(v) = \{v\}$. Otherwise, suppose $v \in X_{g(i)}$. If $f(w) = v$, then $w \in W_i$ and $f_i(w) = v$; as f_i is an isomorphism, there is only one such w; and as f_i is color preserving, the color of w equals the color of v.

Consider an edge $\{v, w\} \in E$. There must be an i, with $v, w \in W_i \cup X_r$. Now, $f(v) = f_i(v)$, and $f(w) = f_i(w)$. As f_i is an isomorphism, $\{f(v), f(w)\} = \{f_i(v), f_i(w)\} \in E$. Similarly, a pair of non-adjacent vertices is mapped to a pair of non-adjacent vertices. □

Define for $i,\ 1 \le i \le r',\ Z'_i = \{f(v) \mid v \in Z_i\}$.

Claim 8 *$(Y_1, Y_2, \ldots, Y_s, Z'_1, Z'_2, \ldots, Z'_{r'})$ is a properly coloured path decomposition.*

Proof We first prove that $(Y_1, Y_2, \ldots, Y_s, Z'_1, Z'_2, \ldots, Z'_{r'})$ is a path decomposition.

First, we show that every edge $\{v, w\} \in E$ is contained in some bag of $(Y_1, Y_2, \ldots, Y_s, Z'_1, Z'_2, \ldots, Z'_{r'})$. If $v, w \in Y_s$, then we can take the bag Y_s; so w.l.o.g., let $v \notin Y_s$. If v belongs to a left component W_i of $(Y_1, \ldots, Y_s)$, then there must be a bag $Y_j,\ 1 \le j \le s-1$ that contains v and w as $(Y_1, Y_2, \ldots, Y_s)$ is a partial path decomposition. Now suppose that v belongs to a right component W_i of $(Y_1, \ldots, Y_s)$. We have that $W_{g^{-1}(i)}$ is a right component of $(X_1, \ldots, X_r)$. As $\{f^{-1}(v), f^{-1}(w)\} \in E$, there is a bag in $(X_1, X_2, \ldots, X_r, Z_1, Z_2, \ldots, Z_{r'})$ that contains both $f^{-1}(v)$ and $f^{-1}(w)$. As $W_{g^{-1}(i)}$ is a right component, this bag must be one of the $Z_j,\ 1 \le j \le r'$, and thus $v, w \in Z'_j$.

Second, it now directly follows that $\left(\bigcup_{1\le i\le s} Y_i\right) \cup \left(\bigcup_{1\le i\le r'} Z'_i\right) = V$: as G is connected, each vertex is endpoint of an edge.

Third, we show that every $v \in V$ only occurs in a series of consecutive bags. For a vertex $v \in Y_s = X_r$, we note that there are $1 \le \alpha \le s$, $0 \le \beta \le r'$, such that v belongs to bags $Y_\alpha, Y_{\alpha+1}, \ldots, Y_s$, and v belongs to bags $Z_1, Z_2, \ldots, Z_\beta$, and no other bags. As $f(v) = v,\ v$ also belongs to bags $Z'_1, Z'_2, \ldots, Z'_\beta$, and no later bags. So, for a vertex $v \in Y_s = X_r$, we are done.

If $v \in W_{g(i)}$ where $W_{g(i)}$ is a left component of $(Y_1, Y_2, \ldots, Y_s)$. Then, $f^{-1}(v) \in W_i$ with W_i a left component of $(X_1, X_2, \ldots, X_r)$. Thus, $f^{-1}(v)$ belongs to one or more consecutive bags in $(X_1, X_2, \ldots, X_{r-1})$, and, as $f^{-1}(v)$ does not belong to X_r, $f^{-1}(v)$ does not belong to $Z_1, Z_2, \ldots, Z_{r'}$ because otherwise $(X_1, X_2, \ldots, X_r, Z_1, Z_2, \ldots, Z_{r'})$ is not a path decomposition. So, v belongs to one or more consecutive bags in $(Y_1, Y_2, \ldots, Y_{s-1})$ and no others. And, if $v \in W_{g(i)}$ where $W_{g(i)}$ is a right component of $(Y_1, Y_2, \ldots, Y_s)$, then the required result follows from a similar analysis.

Finally, by assumption all vertices in a bag Y_i have a different color, and, as f is color preserving, as all vertices in a bag Z_i have a different color, also all vertices in a bag Z'_i have a different color. We thus have shown Claim 8. □

So, $(Y_1, Y_2, \ldots, Y_s)$ has an extension that is a properly coloured path decomposition of G. This shows one direction of implication of the proposition; the proof of the other direction is identical. Thus, Proposition 6 holds. □

We assume some ordering on the vertices. The *characteristic* of a partial path decomposition $(X_1, X_2, \ldots, X_r)$ is the following pair:

$$(X_r, \bigcup_{1 \le i \le r-1} X_i - X_r),$$

where we assume that both vertex sets are given as an ordered list of vertices.

Two properly coloured partial path decompositions with the same characteristic are trivially equivalent, using the identity for g. The algorithm in Section 5 basically tabulates all different characteristics of properly coloured partial path decompositions, and thus gives an $\mathcal{O}(2^n)$ algorithm for INTERVALING COLOURED GRAPHS; this is somewhat similar to the Held-Karp algorithm for TSP [21]. In case the number of colors is bounded, we can obtain a faster algorithm by applying an isomorphism check for components; this is the main ingredient of the faster algorithm described in the next section.

4 An Exact Algorithm for Intervalizing k-Coloured Graphs

In this section, we give the algorithm for INTERVALIZING k-COLOURED GRAPHS, building upon the notions and preliminary results of the previous sections.

First, we note that a positive instance has a path decomposition in which each bag has size at most k (all vertices in a bag have a different color and there are k colors). Thus, as a first step we use the linear time algorithm (for fixed k), that tests if the pathwidth of the input graph is at most $k-1$ from [4, 10]. If not, we are done, and can decide negatively. Thus, we can assume that G has pathwidth at most k in the remainder. We consider k to be a constant.

We introduce some further notions.

We define the *progress* of a partial path decomposition $(X_1, X_2, \ldots, X_r)$ to be $2 \cdot |\bigcup_{1 \le i \le r} X_i| - |X_r|$. Note that when we extend a nice partial path decomposition with one additional introduce or one additional forget node, then the progress always increases by exactly one. As the characteristic of $(X_1, \ldots, X_r)$ is $(X_r, \bigcup_{1 \le i < r} X_i - X_r)$, it follows that if a partial path decomposition has characteristic (X, Z), its progress equals $2|Z| - |X|$.

The *canonical characteristic* of a properly coloured partial path decomposition is the lexicographically minimal characteristic over all characteristics of equivalent properly coloured partial path decompositions.

Proposition 9 *Given a characteristic of a properly coloured partial path decomposition, we can compute in polynomial time its canonical characteristic.*

Proof The GRAPH ISOMORPHISM problem is polynomial time solvable on graphs of bounded treewidth, and thus also on graphs of bounded pathwidth [3, 15]. It is straightforward to modify the algorithms of [3] or [15] such that it also works on coloured graphs while using the same running time. (More precisely, the very recent result of Fomin et al. [15] shows that GRAPH ISOMORPHISM is fixed parameter tractable, with the treewidth as parameter; this improvement is however suppressed by other factors in the running time of our algorithm.)

Given a characteristic (X_r, Z), we first compute (with depth first search) the connected components of $G[V - X_r]$, say $W_1, W_2, \ldots, W_q$. For each pair $W_i,\ W_j$, we can check in polynomial time if they are isomorphic: use the isomorphism algorithm on coloured graphs of bounded pathwidth discussed above, and take a new, different color for each vertex in X_r. (Note the definition of isomorphism for components, as given in Section 3).

Thus, we can partition the components in equivalence classes dictated by isomorphism. We can sort each component lexicographically, and then each class lexicographically. Then, for each class, we determine how many components from the class are a subset of Z (i.e., left components). In the canonical characteristic, we take the same number of left components from the class, but now take this number of lexicographically smallest elements. A simple last sorting step gives the desired result. □

We can now describe our algorithm.

- Check if the pathwidth of G is at most $k - 1$. If not, answer no and terminate.
- Otherwise, for $\alpha = 1 \cdots 2n$, compute a table T_α of all canonical characteristics of partial path decompositions of progress α.
- If T_{2n} is empty, then answer no; otherwise, answer yes.

The output of the algorithm clearly is correct as a partial path decomposition is a path decomposition, if and only if, its progress equals $2n$.

We now describe how the tables T_i are computed. Computing T_1 is simple: for all $v \in V$, we have an entry in T_1 of the form $(\{v\}, \emptyset)$. Given a table $T_\alpha,\ 1 \leq \alpha < 2n$, we compute table $T_{\alpha+1}$ as follows. Initialize $T_{\alpha+1}$ as empty set. For each entry (X, Z) from T_α, do the following:

- Compute the new characteristics that result when the next node in the partial path decomposition is an introduce node: for each $v \in V - Z - X$ such that there is no $x \in X$ with $c(v) = c(x)$, compute the canonical characteristic of $(X \cup \{v\}, Z)$ and put it in $T_{\alpha+1}$.
- Compute the new characteristics that result when the next node in the partial path decomposition is a forget node: for each $x \in X$ such that there is no $v \in Z - V - X$ with $\{v, x\} \in E$, compute the canonical characteristic of $(X - \{v\}, Z \cup \{v\})$.

Proposition 10 *The procedure correctly computes table $T_{\alpha+1}$.*

Proof Note that the characteristic of a partial path decomposition remains the same when we apply the procedure of Proposition 1. So, we may assume that we compute the canonical characteristics of the properly coloured nice partial path decompositions $(X_1, X_2, \ldots, X_r)$ with progress $\alpha + 1$. Of these, we consider two cases: the last node X_r can be an introduce node or a forget node.

If X_r is an introduce node with $X_r = X_{r-1} \cup \{v\}$, then $(X_1, X_2, \ldots, X_{r-1})$ is a properly coloured partial path decomposition of progress α. If $(X_1, X_2, \ldots, X_{r-1})$

has characteristic (X_{r-1}, Z), then $(X_1, X_2, \ldots, X_r)$ has characteristic $(X_{r-1} \cup \{v\}, Z)$. v must have a color different from the colors of vertices in X_{r-1}.

If X_r is a forget node with $X_r = X_{r-1} - \{v\}$, then again $(X_1, X_2, \ldots, X_{r-1})$ is a properly coloured partial path decomposition of progress α. As v is forgotten, it cannot belong to bags right of X_r, and thus all neighbours of v must belong to $\bigcup_{1 \le i \le r} X_i$. If $(X_1, X_2, \ldots, X_{r-1})$ has characteristic (X_{r-1}, Z), then the characteristic of $(X_1, X_2, \ldots, X_r)$ is $(X_{r-1} - \{v\}, Z \cup \{v\})$. $\square$

This completes the description of the algorithm. From our discussion, we see that the algorithm indeed correctly decides if G has a properly coloured interval completion.

We now will analyse the running time of the algorithm. We remark that our algorithm uses polynomial time per entry in a table T_i. Thus, the running time of the algorithm equals the product of a polynomial in n and the number of canonical characteristics of properly coloured partial path decompositions. So, we need to establish an upper bound on this number of canonical characteristics. First, we obtain an upper bound on the number of non-isomorphic components of a set X.

Proposition 11 *Let $(X_1, X_2, \ldots, X_r)$ be a properly coloured partial path decomposition of G. There are at most $(k \cdot 2^{3k})^\ell$ equivalence classes of the isomorphism relation on components of $G[V - X_r]$ that contain components with ℓ vertices.*

Proof Each equivalence class can be identified by an uncoloured unlabelled graph on ℓ vertices of pathwidth at most $k - 1$, a colouring with at most k colors of the vertices of the graph, and the incidence relation between the vertices in the graph and the vertices in X_r. This gives at most the following number of equivalence classes:

$$(k - 1 + 2^{k-1} + 1)^{2\ell-1} \cdot k^\ell \cdot 2^{k\ell} \le k^\ell \cdot 2^{3k\ell} = (k \cdot 2^{3k})^\ell$$

because the first gives at most $(k - 1 + 2^{k-1} + 1)^{2\ell-1}$ possibilities by Proposition 2, the second at most k^ℓ possibilities, and the last at most $2^{k\ell}$ possibilities. $\square$

To show that the size of a table T_i is bounded by $2^{\mathcal{O}(n/\log n)}$, we consider three cases for the components of $G[V - X_r]$; for each we count the number of possibilities they give in the table T_i.

Let $c_k = \frac{1}{2\log k \cdot 3k}$.

- A component is *large*, if it has at least $c_k \cdot \log n$ vertices. For each, we have the possibility to be a left or a right component. As there are $\mathcal{O}(n/\log n)$ large components, this gives $2^{\mathcal{O}(n/\log n)}$ possibilities for the large components.
- A component is *frequent-small*, if it is not large and it has at least $\sqrt{n}$ isomorphic components of $G[V - X_r]$. Note that there are at most $\sqrt{n}$ equivalence classes of the isomorphism relation that contain frequent-small components. As there are less than n components, this gives at most $n^{\sqrt{n}} = 2^{(\log n)\sqrt{n}} = 2^{O(n/\log n)}$ possibilities.

- A component is *infrequent-small*, if it is not large and it has less than $\sqrt{n}$ isomorphic components of $G[V - X_r]$. There are less than

  $$(k \cdot 2^{3k})^{c_k \log n} = 2^{\log k \cdot 3k \cdot c_k \cdot \log n} = 2^{(\log n)/2} = \sqrt{n}$$

  equivalence classes of the isomorphism relation that contain small components, by Proposition 11. For each class with infrequent-small components, we have less than $\sqrt{n}$ components in the class, and thus at most $\sqrt{n}$ possibilities regarding the number of left components in the class. This gives a total of at most $\sqrt{n}^{\sqrt{n}} = 2^{O(n/\log n)}$ possibilities for infrequent-small components.

For a fixed set X, the number of characteristics of the form (X, Z) is obtained by multiplying the number of possibilities for large, frequent-small, and infrequent-small components. As each case is bounded by $2^{\mathcal{O}(n/\log n)}$, we obtain a bound of $2^{\mathcal{O}(n/\log n)}$. To obtain an upper bound on the total size of a table, we note that X is a subset of at most k vertices, and as $(n+1)^k = 2^{\mathcal{O}(n/\log n)}$, we have that each table T_i has a size bounded by $2^{\mathcal{O}(n/\log n)}$. As the running time of the algorithm is bounded by a polynomial times the size of these tables, we obtain our main result.

Theorem 12 *For every fixed $k \geq 4$, there is an algorithm for* INTERVALIZING k-COLOURED GRAPHS *that runs in time* $2^{\mathcal{O}(n/\log n)}$.

We remark that there are inputs on which the algorithm uses $\Omega(2^{n/\log n})$ time: suppose G has a vertex v that is a separator such that $G[V - \{v\}]$ has $\Omega(n/\log n)$ non-isomorphic components each of size $\lfloor \log n \rfloor$.

5 An Algorithm for Intervalizing Coloured Graphs with an Arbitrary Number of Coors

In this section, we consider the case that the number of colors is not fixed. We give a simple Held-Karp style dynamic programming algorithm for this problem.

Suppose we are given a properly coloured graph $G = (V, E)$. For a given set of vertices $W \subseteq V$, the *border* of W is the set of vertices in W with at least one neighbour in $V - W$, i.e., we denote

$$B(W) = \{v \in W \mid \exists w \in V - W : \{v, w\} \in E\}$$

A set of vertices $W \subseteq V$ is said to be *fine*, if there exists a properly coloured path decomposition $(X_1, X_2, \ldots, X_s)$ of $G[W]$, such that $B[W] \subseteq X_s$, i.e., the last bag contains all vertices in the border of W.

Lemma 13 *For all $W \subseteq V$, $W \neq \emptyset$, W is fine, if and only if, there exists a $v \in W$, such that $W - \{v\}$ is fine and all vertices in $B(W - \{v\}) \cup \{v\}$ have a different color.*

Proof Suppose W is fine. Suppose $(X_1, X_2, \ldots, X_s)$ is a properly coloured path decomposition of $G[W]$ with $B(W) \subseteq X_s$. If $s = 1$, the result follows directly (any vertex in X_1 can play the role of v). Suppose $s > 1$. If $X_s \subseteq X_{s-1}$, then $(X_1, \ldots, X_{s-1})$ is also a properly coloured path decomposition of $G[W]$ with $B(W) \subseteq X_s$, and we look at this path decomposition instead. Repeat the step till $X_s \not\subseteq X_{s-1}$ or $s = 1$. So, we may suppose that $X_s \not\subseteq X_{s-1}$.

Take a vertex $v \in X_s - X_{s-1}$. X_s must contain each vertex $w \in B(W - \{v\})$, as for each such w, either $w \in B(W)$ or $\{v, w\} \in E$. So all vertices in $B(W - \{v\}) \cup \{v\} \subseteq X_s$ have a different color. $W - \{v\}$ is fine, as $(X_1, X_2, \ldots, X_{s-1}, X_s - \{v\})$ fulfils the stated condition.

For the other direction, suppose that $W - \{v\}$ is fine, and all vertices in $B(W - \{v\}) \cup \{v\}$ have a different color. Let $(Y_1, Y_2, \ldots, Y_r)$ be a properly coloured path decomposition with $B(W - \{v\}) \subseteq Y_r$. A simple case analysis shows that $(Y_1, Y_2, \ldots, Y_r, B(W - \{v\}_\cup \{v\})$ is a properly coloured path decomposition of $G[W]$ with $B(W) \subseteq B(W - \{v\}) \cup \{v\}$. E.g., each neighbour in v that belongs to $W - \{v\}$ belongs to $B(W - \{v\})$, and thus to the last bag. □

Lemma 13 directly implies the existence of a dynamic programming algorithm that uses $\mathcal{O}^*(2^n)$ time. For $i = 0, 1, \ldots, n$, we compute the collection of fine sets W with $|W| = i$; call this collection $F(i)$. For $i = 0$, we note that the empty set is fine, i.e., $F(0) = \{\emptyset\}$. If $i > 0$, initialize $F(i)$ as an empty collection. Then, perform the following step for each fine set $Y \in F(i - 1)$:

- Compute the border of Y, $B[Y]$. This can be done in linear time using depth first search.
- If $B[Y]$ contains two vertices of the same color, we do not further process Y, otherwise continue with the next step.
- For all vertices $v \in V - Y$,
    - Check if $B[Y]$ contains a vertex with the same color as v.
    - If not, then $Y \cup \{v\}$ is a fine set of size i. If $F(i)$ does not yet contain $Y \cup \{v\}$, then add $Y \cup \{v\}$ as a new element to $F(i)$.

It is easy to see that the amount of work per fine set of vertices is polynomial. Finally, G has a properly coloured interval completion, if and only if $F(n) \neq \emptyset$. Thus, we have

Theorem 14 *The* INTERVALIZING COLOURED GRAPHS *problem can be solved in* $\mathcal{O}^*(2^n)$ *time.*

6 Conclusions

In this paper, we gave a dynamic programming algorithm for the INTERVALIZING k-COLOURED GRAPHS problem for a fixed number of colors k. It uses subexponential time of a somewhat unusual form, and thus, the result forms a somewhat curious exception to the types of results that are usually obtained in the field. The result is merely of theoretical interest, as values of n for which the algorithm can be run in practice can be expected to be rather small, say below 100. Experiments with a somewhat similar Held-Karp style algorithm for TREEWIDTH [9] suggest that our algorithm can also be practical for small values of n. In particular, it would be interesting to test a variant of the algorithm where the isomorphism test is applied heuristically, i.e., only on components that are very small, and with components, and with a usual graph isomorphism heuristic instead of the (complicated and probably only theoretically interesting) algorithms from [3, 15].

The result with an arbitrary number of colors from Section 5 follows more standard arguments; the algorithm is similar to Held-Karp style algorithms for several layout problems, see [8].

A minor generalization of the result can be obtained when we consider a small ($o(\log n)$) number of colors; in such cases the algorithm also uses subexponential time.

A generalization of the INTERVALIZING k-COLOURED GRAPHS problem is the INTERVAL GRAPH SANDWICH problem, in which we are given two graphs with G and H with the same vertex set, and ask whether there exists an interval graph G' that is a subgraph of H and contains G as a subgraph. A well studied variant has the additional condition that G' has maximum clique size k. See e.g., [19, 22]. The ideas of our paper seem not to give results better than an algorithm that uses $\Theta^*(2^n)$ time for this problem however, still assuming that k is fixed.

Other related problems are the version where we ask to find a properly coloured *proper interval graph*, which is polynomial for a fixed number of colors k [2], and the problem to find a properly coloured *chordal graph*, which is also polynomial for a fixed number of colors [24].

Very recently, Bodlaender and Nederlof [11] obtained a lowerbound for the problem: assuming the Exponential Time Hypothesis, an algorithm for INTERVALIZING k-COLOURED GRAPHS uses at least $2^{\Omega(n/\log n)}$ time, for each fixed number of colours at least six.

Acknowledgments We thank Jesper Nederlof for useful discussions.

References

1. Àlvarex, C., Díaz, J., Serna, M.: The hardness of intervalizing four colored caterpillars. Discret. Math. **235**, 19–27 (2001)
2. Àlvarex, C., Serna, M.: On the proper intervalization of colored caterpillar trees. Informatique Théorique et Applications **43**, 667–686 (2010)
3. Bodlaender, H.L.: Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. J. Algorithm. **11**, 631–643 (1990)
4. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**, 1305–1317 (1996)
5. Bodlaender, H.L., de Fluiter, B.: Intervalizing k-colored graphs. In: Fülöp, Z., Gécseg, F. (eds.) Proceedings of the 22nd International Colloquium on Automata, Languages and Programming, ICALP'95, Lecture Notes in Computer Science, vol. 944, pp. 87–98. Springer, Berlin Heidelberg New York (1995)

6. Bodlaender, H.L., de Fluiter, B.: On intervalizing k-colored graphs for DNA physical mapping. Discret. Appl. Math. **71**, 55–77 (1996)
7. Bodlaender, H.L., de Fluiter, B.L.E.: Intervalizing k-colored graphs. Technical Report UU-CS-1995-15. Department of Computer Science. Utrecht University, Utrecht (1995)
8. Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M.: A note on exact algorithms for vertex ordering problems on graphs. Theory Comput. Syst. **50**, 420–432 (2012)
9. Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M.: On exact algorithms for treewidth. ACM Trans. Algoritm. **9**(1), 12 (2012)
10. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithm. **21**, 358–402 (1996)
11. Bodlaender, H.L., Nederlof, J.: Unpublished results (2015)
12. Demaine, E.D., Hajiaghayi, M.: The bidimensionality theory and its algorithmic applications. Comput. J. **51**, 292–302 (2008)
13. Fellows, M.R., Hallett, M.T., Wareham, H.T.: DNA physical mapping: three ways difficult (extended abstract). In: Lengauer, T. (ed.) Proceedings of the 1st Annual European Symposium on Algorithms, ESA'93, Lecture Notes in Computer Science, vol. 726, pp. 157–168. Springer, Berlin Heidelberg New York (1993)
14. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Springer, Berlin Heidelberg New York (2010)
15. Fomin, F.V., Lokshtanov, D., Saurabh, S.: Efficient computation of representative sets with applications in parameterized and exact algorithms. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, pp. 142–151 (2014)
16. Fomin, F.V., Thilikos, D.M.: A simple and fast approach for solving problems on planar graphs. In: Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science, STACS 2004, Lecture Notes in Computer Science, vol. 2996, pp. 56–67. Springer, Berlin Heidelberg New York (2004)
17. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980)
18. Golumbic, M.C., Kaplan, H., Shamir, R.: On the complexity of DNA physical mapping. Adv. Appl. Math. **15**, 251–261 (1994)
19. Golumbic, M.C., Kaplan, H., Shamir, R.: Graph sandwich problems. J. Algorithm. **19**, 449–472 (1995)
20. Heggernes, P., Suchan, K., Todinca, I., Villanger, Y.: Minimal interval completions. In: Proceedings of the 13th Annual European Symposium on Algorithms, ESA 2005, Lecture Notes in Computer Science, vol. 3669, pp. 403–414. Springer, Berlin Heidelberg New York (2005)
21. Held, M., Karp, R.: A dynamic programming approach to sequencing problems. J. Soc. Ind. Appl. Math. **10**, 196–210 (1962)
22. Kaplan, H., Shamir, R.: Bounded degree interval sandwich problems. Algorithmica **24**, 96–104 (1999)
23. Lipton, R.J., Tarjan, R.E.: Applications of a planar separator theorem. SIAM J. Comput. **9**, 615–627 (1980)
24. McMorris, F.R., Warnow, T., Wimer, T.: Triangulating vertex-colored graphs. SIAM J. Discret. Math. **7**(2), 296–306 (1994)