Theory of Computing Systems (2019) 63:450–465
https://doi.org/10.1007/s00224-018-9882-1

On Tradeoffs Between Width- and Fill-like Graph Parameters

Dariusz Dereniowski[1] · **Adam Stański**[2]

Published online: 28 July 2018

Abstract

In this work we consider two two-criteria optimization problems: given an input graph, the goal is to find its interval (or chordal) supergraph that minimizes the number of edges and its clique number simultaneously. For the interval supergraph, the problem can be restated as simultaneous minimization of the pathwidth $\mathtt{pw}(G)$ and the profile $p(G)$ of the input graph G. We prove that for an arbitrary graph G and an integer $t \in \{1, \ldots, \mathtt{pw}(G)+1\}$, there exists an interval supergraph G' of G such that for its clique number it holds $\omega(G') \leq (1+\frac{2}{t})(\mathtt{pw}(G)+1)$ and the number of its edges is bounded by $|E(G')| \leq (t+2)p(G)$. In other words, the pathwidth and the profile of a graph can be simultaneously minimized within the factors of $1+\frac{2}{t}$ (plus a small constant) and $t+2$, respectively. Note that for a fixed t, both upper bounds provide constant factor approximations. On the negative side, we show an example that proves that, for some graphs, there is no solution in which both parameters are optimal. In case of finding a chordal supergraph, the two corresponding graph parameters that reflect its clique size and number of edges are the treewidth and fill-in. We obtain that the treewidth and the fill-in problems are also 'orthogonal' in the sense that for some graphs, a solution that minimizes one of those parameters cannot minimize the other. As a motivating example, we recall graph searching games which illustrates a need of simultaneous minimization of these pairs of graph parameters.

Keywords Fill-in · Graph searching · Node search · Pathwidth · Profile · Treewidth

Author partially supported by National Science Centre (Poland) grant number 2015/17/B/ST6/01887

This research has been conducted prior to the employment of this author in Amazon.

✉ Dariusz Dereniowski
deren@eti.pg.edu.pl

Adam Stański
stanskia@amazon.com

[1] Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

[2] Amazon.com, Gdańsk, Poland

1 Introduction

Multi-criteria optimization problems can be of interest for several reasons, including theoretical insights their study provides or potential practical applications. The selection of the parameters to be simultaneously optimized is dictated by those and can lead to challenging research questions. Our selection is motivated in two ways. First, the choice of the parameters themselves is made according to their importance in graph theory and algorithm design. Second, we paired the parameters according to a potential application that we describe in detail. The first pair of parameters that we minimize is the pathwidth and profile, which can be viewed as computations of linear graph layouts of certain characteristics. The second pair is the treewidth and fill-in, which is a tree-like graph layout counterpart of the former.

1.1 Related Work

We point out several optimization problems in which pathwidth or treewidth is paired with another parameter or with additional conditions that need to be satisfied. For an example consider a problem of computing a path decomposition with restricted width and length (defined as the number of bags in the path decomposition). It has been first studied in [2] as a problem motivated by an industrial application and called the *partner unit problem* but finds applications also in scheduling and register allocation [36] or graph searching games [15]. It turns out that there exists a polynomial-time algorithm for computing a path decomposition of width $k \leq 3$ and minimum length for an arbitrary input graph, but the problem becomes NP-hard for any width $k \geq 4$ [15]. If we fix the length to be 2 and ask for a minimum width path (or tree) decomposition, then the problem is also NP-hard [24]. Also, a minimum length path decomposition of a width k can be computed efficiently for k-connected graphs [22, 23]. A natural treewidth counterpart of the 'length' minimization problem can be seen as minimizing, besides of the width of a tree decomposition, the number of its bags [33]. It is known that such problem is NP-complete for any fixed $k \geq 4$, it is polynomial for $k \leq 2$ and for $k = 3$ it is polynomial for trees and 2-connected outerplanar graphs [33]. See [10] for an algorithm of running time $2^{O(n/\log n)}$ that solves both problems for a fixed k. For a more general approach through using a cost function on tree decompositions see [7].

We refer the reader to a related problem of minimization of width and the diameter of the underlying tree-structure of the decomposition [6, 9]. (The *diameter* of a tree decomposition is defined to be the maximum distance between any two nodes of the three decomposition.)

A very closely related research area includes several graph searching games. We now restrict ourselves to a short and informal introduction and an overview, and in Section 6 we give a formal statement of a graph searching problem, which gives a motivation and a potential application of our results. The problem of graph searching can be informally stated as one in which an agent called the *fugitive* is moving around the graph with the goal to escape a group of agents called guards or *searchers*. There are many variations of this problem specifying behavior of the fugitive and the searchers, phase restrictions, speeds of both parties or their other capabilities like

visibility, radius of capture etc. Numerous optimization criteria have been studied for these games as well. However, the tradeoffs between different optimization parameters have not yet been throughly analyzed. In this work we refer to one of the two classical formulations of the graph searching problem, namely the *node search* (see a formal definition in Section 6).

In the original statement of the problem the fugitive is considered invisible (i.e., the searchers can deduce its potential locations only based on the history of their moves) and *active*, i.e., constantly moving with unbounded speed to counter the searchers' strategy. It turns out that the minimum number of searchers sufficient to guarantee the capture of the fugitive corresponds to the pathwidth of the underlying graph [12]. Later on, the lazy, also referred to as *inert*, fugitive variant has been defined in which the fugitive only moves when the searchers are one move apart from catching it. The latter version was first introduced in [12] where the authors show that minimizing the number of searchers precisely corresponds to finding the treewidth of the input graph. Seymour and Thomas proposed in [37] a variant of the game in which the fugitive was visible and active. In the same paper they prove that the visible active variant of the problem is equivalent to the invisible inert variant.

All previously mentioned problems considered the number of searchers used by a strategy to be the optimization criterion. In [16], the authors analyzed the cost defined (informally) as the sum of the guard counts over all steps of the strategy. This graph searching parameter is the one that corresponds to the profile minimization.

Not much is known in terms of two-criteria optimization in the graph searching games. To mention some, examples, there is an analysis of simultaneous minimization of time (number of 'parallel' steps) and the number of searchers for the visible variant (this corresponds precisely to the above-mentioned width and length minimization of path decompositions) [15] and for the inert one [33] of the node search. For more examples of very closely related two-criteria problems that can be found in the graph searching games see e.g. [8, 14, 30].

The pathwidth or treewidth parameters have been also studied with additional constraints which can be most generally stated as requiring certain connectivity structures to be induced by the bags. These include the parameter of connected pathwidth introduced in [4] in the context of graph searching games and studied further e.g. in [3, 13]. (A path decomposition is *connected* if the union of the bags from each prefix of the path decomposition induced a connected subgraph.) For a relation with the graph searching games we point out that pathwidth problem is equivalent to the *node search game*, is equivalent up to an additive difference of 1 to the *edge search game* and up to a multiplicative factor of 2 (plus a $o(1)$ additive term) to the *connected search game*, see e.g. [3, 13, 18]. Another example includes the connected treewidth [19].

1.2 Outline

This work mostly deals with simultaneous minimization of width-like (namely pathwidth and treewidth) and fill-like (namely profile and fill-in) graph parameters. In order to state our results for pathwidth and profile formally, we introduce the necessary notation in Section 2. For pathwidth and profile we give an upper bound (to be

precise, a class of upper bounds that results in a tradeoff between the two parameters) in Section 3 (Theorem 1) and, in Section 4, an example that shows that the two cannot be simultaneously minimized in general (Theorem 2). The latter example also is valid for the tradeoffs between the two corresponding parameters, treewidth and fill-in and for this reason we introduce the two in Section 5 and state this result as a corollary (Corollary 1). Section 6 recalls two classical graph searching problems which serve as an example that illustrates a case in which it is natural to optimize the two selected pairs of parameters. These connections are summarized there in Remarks 1 and 2. Thus, this part of the work serves as an additional motivation for this research.

2 Preliminaries

We start with recalling some basic graph-theoretic terms used in this work. For a graph G, we write $V(G)$ and $E(G)$ to denote the sets of its vertices and edges, respectively. We say that a graph G' is a *subgraph* of a graph G (and in such case G is a *supergraph* of G') if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. Moreover, G' is a subgraph of G *induced* by $X \subseteq V(G)$ and denoted $G[X]$ (or G' is an *induced subgraph* of G for short) if $V(G') = X$ and $E(G') = \{\{u, v\} \in E(G) \mid u, v \in X\}$. A *clique* is a graph in which any two vertices are adjacent. For a vertex v of a graph G, $N_G(v)$ is the set of neighbors of v in G.

We now recall the graph parameters studied in this work. For a permutation $f : V(G) \to \{1, \ldots, |V(G)|\}$ of the vertices of G, define

$$p_f(G) = \sum_{v \in V(G)} \left(f(v) - \min_{u \in \{v\} \cup N_G(v)} f(u) \right).$$

Informally, $f(v) - \min_{u \in \{v\} \cup N_G(v)} f(u)$ can be interpreted as the maximum distance, according to the permutation f, between v and its neighbors appearing in the permutation prior to v (if all neighbors of v are ordered in f after v, then this difference is by definition zero). Then, a *profile* of a graph G [21], denoted by $p(G)$, is defined as

$$p(G) = \min \left\{ p_f(G) \mid f \text{ is a permutation of } V(G) \right\}. \tag{1}$$

A *tree decomposition* [1, 35] of a simple graph $G = (V(G), E(G))$ is a pair $(\mathcal{X}, \mathcal{T})$, where $\mathcal{X} = \{X_1, \ldots, X_d\}$ is a collection of subsets of $V(G)$, i.e., $X_i \subseteq V(G)$ for each $i \in \{1, \ldots, d\}$, and $\mathcal{T} = (\{1, \ldots, d\}, F)$ is a tree whose vertices correspond to the elements of $\mathcal{X}$, such that the following conditions are satisfied:

- $\bigcup_{i=1,\ldots,d} X_i = V(G)$,
- for each $\{u, v\} \in E(G)$ there exists $i \in \{1, \ldots, d\}$ such that $u, v \in X_i$,
- for each i, j, k, if j is on the path from i to k in $\mathcal{T}$, then $X_i \cap X_k \subseteq X_j$.

The *width* of the tree decomposition equals $\max_{i=1,\ldots,d} |X_i| - 1$, and the *treewidth* of G, denoted by $\mathtt{tw}(G)$, is the smallest width of all path decompositions of G. A tree decomposition is called a *path decomposition* if $\mathcal{T}$ is a path, and the corresponding graph parameter that minimizes the width over all path decomposition is called the *pathwidth* of G and is denoted by $\mathtt{pw}(G)$.

2.1 Interval Graphs

A graph G is an *interval graph* if and only if for each $v \in V(G)$ there exists an interval $I_v = (l_v, r_v)$ such that for each edge $u, v \in V(G)$ it holds: $\{u, v\} \in E(G)$ if and only if $I_u \cap I_v \neq \emptyset$. The collection $\mathcal{I} = \{I_v \mid v \in V(G)\}$ is called an *interval representation of* G. An interval representation $\mathcal{I}$ of G is said to be *canonical* if the endpoints of I_v are integers for each $v \in V(G)$ and $\{l_v \mid v \in V(G)\} = \{1, \ldots, n\}$. This in particular implies that the left endpoints are pairwise different. Denote by $\mathcal{R}(G)$ the set of all canonical interval representations of G. We will write $\mathcal{R}(G)$ for a graph G that is not an interval graph to denote the set $\bigcup_{G' \in X} \mathcal{R}(G')$, where X is the set of all interval supergraphs of G with the same vertex set as G. If $\mathcal{I}$ is an interval representation of an interval graph G and $v \in V(G)$, then $\mathcal{I}(v)$ denotes the interval in $\mathcal{I}$ that corresponds to v. For any interval I, we write $\text{left}(I)$ and $\text{right}(I)$ to denote its left and right endpoint, respectively. Note that we consider without loss of generality only open intervals in the interval representations.

Let G be an interval graph. Given an interval representation $\mathcal{I}$ of G and an integer i, define

$$m_i(\mathcal{I}) = |\{I \in \mathcal{I} \mid i \in I\}|$$

to be the cardinality of the set of all intervals that contain the point i. Let $I_1, \ldots, I_n$ be the intervals in $\mathcal{I}$. Then, let $f_i(\mathcal{I}) = m_j(\mathcal{I})$, where $j = \text{left}(I_i)$, $i \in \{1, \ldots, n\}$. In other words, $f_i(\mathcal{I})$ is the number of intervals in $\mathcal{I}$ containing the point $\text{left}(I_i)$. (Note that $\text{left}(I_i) \notin I_i$ and hence I_i does not contribute to the value of $f_i(\mathcal{I})$.)

Given a canonical interval representation $\mathcal{I}$ of an interval graph G, define the *interval cost* of $\mathcal{I}$ as

$$ic(\mathcal{I}) = \sum_{i=1}^{n} f_i(\mathcal{I}), \qquad \text{where } n = |\mathcal{I}|.$$

It turns out that $ic(\mathcal{I})$ equals the number of edges of the interval graph with interval representation $\mathcal{I}$ (see e.g. [16]). For a graph G, we define its *interval cost* as

$$ic(G) = \min\{ic(\mathcal{I}) \mid \mathcal{I} \in \mathcal{R}(G)\}.$$

The next fact follows from [5] and [16].

Proposition 1 *Let G be any graph and let k be an integer. The following inequalities are equivalent:*

(i) $\left|E(G')\right| \leq k$, *where G' is an interval supergraph of G having the minimum number of edges,*
(ii) $ic(G') \leq k$,
(iii) $p(G) \leq k$.

Let G be an interval graph and let $\mathcal{I} \in \mathcal{R}(G)$. We define the *width of* $\mathcal{I}$ as $w(\mathcal{I}) = \max\{m_i(\mathcal{I}) \mid i \in \mathbb{R}\}$. The *interval width* of any simple graph G is then

$$iw(G) = \min\{w(\mathcal{I}) \mid \mathcal{I} \in \mathcal{R}(G)\}.$$

We have the following fact [26–28, 31, 34]:

Proposition 2 *Let G be any graph and let k be an integer. The following inequalities are equivalent:*

(i) $iw(G) \leq k$,
(ii) $\mathrm{pw}(G) \leq k - 1$.

2.2 Problem Formulation

For the purposes of this work we need an 'uniform' formulation of the two graph-theoretic problems that we study, namely pathwidth and profile, in order to be able to formally apply the two optimization criteria to a single solution to a problem instance. In view of Propositions 1 and 2, we can state the optimization version of our problem as follows:

Problem PPM (***Pathwidth & Profile Minimization***)**:**

Input: a graph G, integers k and c.
Question: does there exist an interval supergraph G' of G such that $iw(G') \leq k$ and $\left|E(G')\right| \leq c$?

3 Pathwidth and Profile Tradeoffs

In this section we prove that for an arbitrary graph G there exists its interval supergraph G' with width at most $(1 + \frac{2}{t})(\mathrm{pw}(G) + 1)$ and the number of edges at most $(t + 2)p(G)$ for each $t \in \{1, \ldots, iw(G)\}$. This is achieved by providing a procedure that finds a desired interval supergraph (the procedure returns an interval representation of this supergraph). Since the goal is to prove an upper bound and not to provide an efficient algorithm, this procedure relies on optimal algorithms for finding a minimum width and minimum cost interval supergraph of a given graph. (The latter problems are NP-complete, see [20, 25, 32, 39].) Therefore, the running time of this procedure is exponential.

We first give some intuitions on our method. We start by computing a 'profile-optimal' (canonical) interval representation $\mathcal{I}$ of some interval supergraph G'' of G, that is, $ic(\mathcal{I}) = p(G)$. Then, in the main loop of the procedure this initial interval representation is iteratively refined. Each refinement targets an interval (i, j) selected in such a way that the width of $\mathcal{I}$ exceeds k/t at each point in (i, j), i.e., $m_{i'}(\mathcal{I}) > k/t$ for each $i' \in \{i, \ldots, j\}$ and i, j are taken so that the interval is maximum with respect to this condition. The refinement on $\mathcal{I}$ in (i, j) is done as follows (see the pseudocode below for detail and Fig. 1 for an example):

- intervals that cover (i, j) entirely or have an empty intersection with it do not change (see Case (i) in Fig. 1),
- intervals that contain one of i or j will be extended to cover entire interval, except that we ensure that they have pairwise different left endpoints as required in canonical representations (Cases (ii) and (iii) in Fig. 1),

- for the intervals that originally are contained in (i, j), we recompute the interval representation; while doing so we take care of the following: first, the neighborhood relation in the initial graph is respected so that the new interval representation provides an interval supergraph of G, and second, the width of the new interval representation inside (i, j) is minimal (Case (iv) in Fig. 1).

The above refinement is performed for each interval (i, j) that satisfies given conditions. Each refinement potentially increases the interval cost of $\mathcal{I}$ but narrows down its width appropriately in the interval for which the refinement is done.

A formal pseudocode is given below as Procedure IC (*Interval Completion*) that as an input takes any graph G and an integer $t \in \{1, \ldots, iw(G)\}$, and returns an $\mathcal{I}' \in \mathcal{R}(G)$. Then, in Lemma 1, we prove that the procedure is correct and in Theorem 1 we estimate the width and interval cost of $\mathcal{I}'$. See Fig. 1 for an example that illustrates the transition performed in a single iteration of the procedure.

Procedure IC (*Interval Completion*)

Input: A graph G and an integer $t \in \{1, \ldots, iw(G)\}$.
Output: An interval representation of some interval supergraph of G.
Compute an interval representation $\mathcal{I} \in \mathcal{R}(G)$, where $ic(\mathcal{I}) = p(G)$.
Set $q \leftarrow 1$ and $\mathcal{I}'' \leftarrow \mathcal{I}$.
while $m_{i'}(\mathcal{I}'') > (\mathrm{pw}(G) + 1)/t$ for some $i' \geq q$ **do**
Find minimum integers i, j such that $q \leq i < j$, $m_{i-1}(\mathcal{I}'') \leq (\mathrm{pw}(G) + 1)/t$, $m_{j+1}(\mathcal{I}'') \leq (\mathrm{pw}(G)+1)/t$ and $m_{i'}(\mathcal{I}'') > (\mathrm{pw}(G)+1)/t$ for each $i' \in \{i, \ldots, j\}$.
Set $q \leftarrow j + 1$.
Let $\bar{G}$ be the subgraph of G induced by all vertices v such that $\mathcal{I}''(v) \subseteq (i, j)$.
Let $\bar{\mathcal{I}}$ be a minimum width canonical interval representation of some interval supergraph of $\bar{G}$.
Construct an interval representation $\mathcal{I}'$ of G as follows:
(i) Let initially $\mathcal{I}' = \{I \in \mathcal{I}'' \mid I \cap (i, j) = \emptyset \text{ or } (i, j) \subsetneq I\}$.
(ii) For each $v \in V(G)$ such that $i \in \mathcal{I}''(v)$ and $j \notin \mathcal{I}''(v)$, add $\mathcal{I}'(v) \leftarrow (\mathrm{left}(\mathcal{I}''(v)), j)$ to $\mathcal{I}'$.
(iii) Let $v_1, \ldots, v_p$ be all vertices of G such that $i \notin \mathcal{I}''(v_s)$ and $j \in \mathcal{I}''(v_s)$ for each $s \in \{1, \ldots, p\}$.
For each $s \in \{1, \ldots, p\}$, add $\mathcal{I}'(v_s) \leftarrow (i + s - 1, j)$ to $\mathcal{I}'$.
(iv) For each $v \in V(\bar{G})$, set $\mathcal{I}'(v) \leftarrow (i + p + \mathrm{left}(\bar{\mathcal{I}}(v)) - 1, i + p - 2 + \mathrm{right}(\bar{\mathcal{I}}(v)))$.
Set $\mathcal{I}'' \leftarrow \mathcal{I}'$.
end while
return $\mathcal{I}'$

Lemma 1 *Let G be any graph and let $t \in \{1, \ldots, iw(G)\}$. Procedure IC for the given G and t returns a canonical interval representation of some interval supergraph of G.*

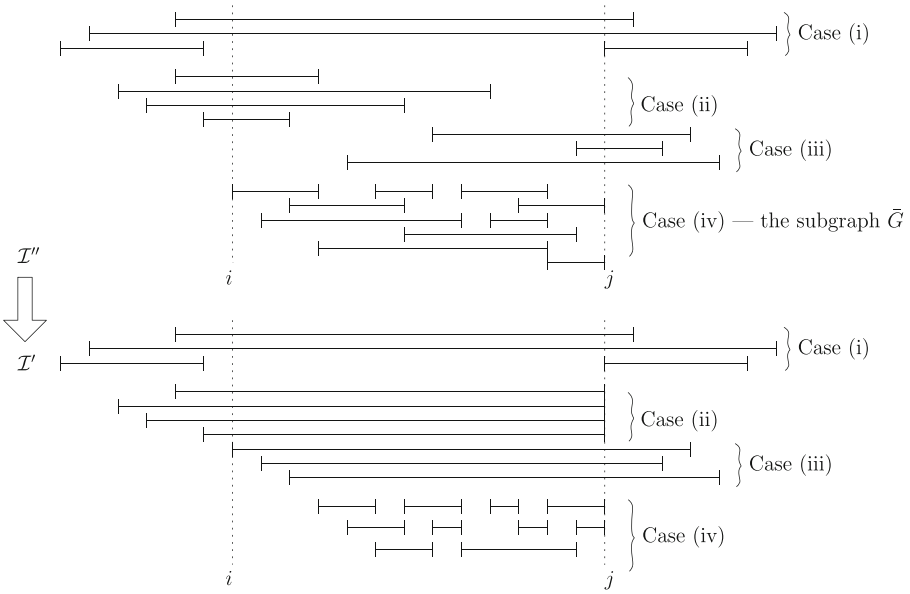

Fig. 1 A single iteration of Procedure IC: transition from interval representation $\mathcal{I}''$ to $\mathcal{I}'$ with particular intervals marked according in which of the cases (i)-(iv) they are processed in the pseudocode

Proof We start by noting that the execution of the procedure completes at some point. This follows from an observation that in an iteration of the main loop, a modification to $\mathcal{I}''$ is made by changing the endpoints of some vertex corresponding intervals that are contained in an interval (i, j), where i and j are selected specifically for this iteration. Also, the variable q is set to be $j + 1$ in this iteration. By the choice of (i, j), and in particular by $i \geq q$, we obtain that the subsequent iteration will modify an interval that is to the right of (i, j). This, by an inductive argument, implies that the number of iterations is bounded by the number of endpoints in a canonical representation, which is $O(n)$.

Let $\mathcal{I}$ be the canonical interval representation of some interval supergraph of G computed at the beginning of Procedure IC. We proceed by induction on the number of iterations of the main loop of Procedure IC, namely, we prove that the interval representation $\mathcal{I}''$ obtained in the s-th iteration is a canonical interval representation of some interval supergraph of G. For the purpose of the proof, we use the symbol $\mathcal{I}_s$ to denote the interval representation obtained in the s-th iteration, taking $\mathcal{I}_0 = \mathcal{I}$.

For the base case of $s = 0$ we have that $\mathcal{I}_0 = \mathcal{I}$ and the claim follows. Hence, let $s > 0$. Since $\mathcal{I}_s$ consists of $|V(G)|$ intervals, $\mathcal{I}_s$ is an interval representation of an interval graph G' on $|V(G)|$ vertices. We need to prove that G is a subgraph of G' and that $\mathcal{I}_s$ is canonical.

By the induction hypothesis, there exists an interval supergraph G'' of G and $\mathcal{I}_{s-1} \in \mathcal{R}(G'')$. Note that $V(G) = V(G') = V(G'')$.

To prove that G is a subgraph of G', we argue that $N_G(v) \subseteq N_{G'}(v)$ for each $v \in V(G)$. Denote

$$X_1 = \{x \in V(G) \mid i \in \mathcal{I}_{s-1}(x) \text{ and } j \notin \mathcal{I}_{s-1}(x)\} \text{ and}$$
$$X_2 = \{x \in V(G) \mid i \notin \mathcal{I}_{s-1}(x) \text{ and } j \in \mathcal{I}_{s-1}(x)\},$$

where i and j have values as in the s-th iteration. Also, $\bar{G}$ refers to the subgraph computed in the s-th iteration. For $v \in V(G) \setminus (X_1 \cup X_2 \cup V(\bar{G}))$ we have that $N_{G'}(v) = N_{G''}(v)$ and hence, since G'' is a supergraph of G, $N_G(v) \subseteq N_{G''}(v)$ gives the claim. If $v \in X_1 \cup X_2$, then

$$\begin{aligned} N_{G''}(v) &= \big(N_{G''}(v) \cap (X_1 \cup X_2 \cup V(\bar{G}))\big) \cup \big(N_{G''}(v) \setminus (X_1 \cup X_2 \cup V(\bar{G}))\big) \\ &\subseteq \big(X_1 \cup X_2 \cup V(\bar{G})\big) \cup \big(N_{G''}(v) \setminus (X_1 \cup X_2 \cup V(\bar{G}))\big) = N_{G'}(v). \end{aligned}$$

Thus, for $v \in X_1 \cup X_2$ we also have $N_G(v) \subseteq N_{G''}(v) \subseteq V_{G'}(v)$ as required. If $v \in V(\bar{G})$, then

$$N_G(v) \subseteq N_{\bar{G}}(v) \cup X_1 \cup X_2 \cup X_3 = N_{G'}(v),$$

where X_3 is the set of vertices $x \in V(G)$ such that both i and j belong to $\mathcal{I}_{s-1}(x)$.

Now we argue that $\mathcal{I}_s$ is canonical. Since both endpoints of each interval in $\mathcal{I}_s$ are clearly integers, it is sufficient to prove that for each $i' \in \{1, \ldots, |V(G)|\}$ there exists exactly one interval $I \in \mathcal{I}_s$ whose left endpoint equals i'. If $i' \in \{1, \ldots, i-1\} \cup \{j, \ldots, |V(G)|\}$, then the claim follows from

$$\big\{v \in V(G) \mid (i', i'+1) \subseteq \mathcal{I}_{s-1}(v)\big\} = \big\{v \in V(G) \mid (i', i'+1) \subseteq \mathcal{I}_s(v)\big\},$$

i.e., the interval representations $\mathcal{I}_{s-1}$ and $\mathcal{I}_s$ are identical 'outside' of the interval (i, j). For each $i' \in \{i, \ldots, i+p-1\}$ the claim holds, since $\text{left}(\mathcal{I}_s(v_{i'-i+1})) = i'$. Finally, let $i' \in \{i+p, \ldots, j-1\}$. Since, $\mathcal{I}_{s-1}$ is canonical, $\big|V(\bar{G})\big| = j-i-p-1$. This implies that $\{\text{left}(\mathcal{I}) \mid \mathcal{I} \in \bar{\mathcal{I}}\} = \{1, \ldots, \big|V(\bar{G})\big|\}$ because $\bar{\mathcal{I}}$ is canonical. Thus, there exists $v \in V(\bar{G})$ such that $\text{left}(\bar{\mathcal{I}}(v)) = i' - i - p + 1$. By the construction of $\mathcal{I}_s$, $\text{left}(\mathcal{I}_s(v)) = i'$ as required. □

Theorem 1 *Let G be any graph and let $t \in \{1, \ldots, \mathtt{pw}(G)+1\}$ be an integer. There exists an interval supergraph G' of G and $\mathcal{I}' \in \mathcal{R}(G')$ such that $w(\mathcal{I}') \leq (1+\frac{2}{t})(\mathtt{pw}(G)+1)$ and $ic(\mathcal{I}') \leq (t+2)p(G)$.*

Proof Suppose that Procedure `IC` is executed for the input G and t. Let $\mathcal{I}$ be the canonical interval representation of some interval supergraph of G computed at the beginning of Procedure `IC`. Moreover, take such an $\mathcal{I}$ that satisfies $ic(\mathcal{I}) = ic(G)$. Let r be the number of iterations performed by the main loop. Let $\bar{G}_q$ and $\bar{\mathcal{I}}_q$, $q \in \{1, \ldots, r\}$, be the graph $\bar{G}$ and its interval representation, respectively, computed in the q-th iteration of the main loop. Also, let (i_q, j_q) be the interval used to select $\bar{G}_q$, i.e., $\bar{G}_q$ is the subgraph of G induced by all vertices v such that $\mathcal{I}''(v) \subseteq (i_q, j_q)$ for each $q \in \{1, \ldots, r\}$.

Note that Procedure IC does not specify how $\bar{\mathcal{I}}_q$ is selected for each $q \in \{1, \ldots, r\}$ and therefore for the purpose of this proof of upper bounds we may assume that the interval representation $\bar{\mathcal{I}}_q$ satisfies

$$w(\bar{\mathcal{I}}_q) = iw(\bar{G}_q). \tag{2}$$

By Lemma 1, $\mathcal{I}'$ returned by Procedure IC is a canonical representation of some interval supergraph of G. By construction,

$$m_p(\mathcal{I}') = m_p(\mathcal{I}) \text{ for each } p \notin \bigcup_{q=1}^{r} \{i_q, \ldots, j_q\}. \tag{3}$$

Since $\bar{G}_q$ is a subgraph of G, $iw(\bar{G}_q) \leq iw(G)$ for each $q \in \{1, \ldots, r\}$ and hence by (2) we obtain $w(\bar{\mathcal{I}}_q) \leq iw(G)$. By the choice of i_q and j_q, we have $m_{i_q-1}(\mathcal{I}) \leq iw(G)/t$ and $m_{j_q+1}(\mathcal{I}) \leq iw(G)/t$. Hence, we obtain

$$m_p(\mathcal{I}') \leq m_{i_q-1}(\mathcal{I}) + m_{j_q+1}(\mathcal{I}) + w(\bar{\mathcal{I}}_q) \leq \left(1 + \frac{2}{t}\right) iw(G) \tag{4}$$

for each $p \in \{i_q, \ldots, j_q\}$ and for each $q \in \{1, \ldots, r\}$. (3) and (4) give that $w(\mathcal{I}') \leq (1 + \frac{2}{t})iw(G)$. Observe that, by (4), for each $q \in \{1, \ldots, r\}$ and for each $p \in \{i_q, \ldots, j_q\}$ it holds

$$m_p(\mathcal{I}') \leq (t+2)m_p(\mathcal{I}) \tag{5}$$

because $m_p(\mathcal{I}) \geq iw(G)/t$. By (3) and (5), $ic(\mathcal{I}') \leq (t+2)ic(\mathcal{I})$. Finally note that by Proposition 1, $ic(\mathcal{I}) = p(G)$ and by Proposition 2, $iw(G) = \mathrm{pw}(G) + 1$, which completes the proof. □

4 Pathwidth and Profile are 'Orthogonal'

In this section we prove that the two optimization criteria studied in this work cannot be minimized simultaneously for some graphs. In other words, we prove by example, that there exist graphs G such that any interval supergraph G' of G that has the minimum number of edges (i.e., $\left|E(G')\right| = p(G)$) cannot have minimum width (i.e., $iw(G) > \mathrm{pw}(G) - 1$) and vice versa. The example that we construct will be also used in the next section and for this reason we present it here in terms of chordal graphs, which is a class of graphs that generalizes interval graphs. For that we need some additional definitions.

We say that C is an *induced cycle of length* $k \geq 3$ in a graph G if C is a subgraph of G and $\{\{u, v\} \in E(G) \mid u, v \in V(C)\} = V(C)$, i.e., the only edges in G between vertices in $V(C)$ are the ones in $E(C)$. A graph is *chordal* if there is no induced cycle of length greater than 3 in G. Any edge that does not belong to a cycle C and connects two vertices of C is called a *chord* of C.

In this section we consider a graph G with vertex set $V(G) = A \cup B \cup B' \cup C$ and edges placed in such a way that $A \cup B$, $A \cup B'$, $B \cup C$ and $B' \cup C$ form cliques. In our construction we take any sets that satisfy

$$|A| < |B| = \left|B'\right| < |C| \quad \text{and} \quad |A|\,|C| > |B|^2. \tag{6}$$

We have the following observation.

Lemma 2 *If G' is a minimal chordal supergraph of G, then each of the subgraphs $G'[A \cup C]$ or $G'[B \cup B']$ is either a clique or an union of two disconnected cliques.*

Proof We prove that the subgraph of G' induced by $A \cup C'$ is either a clique or is disconnected and the proof for $B \cup B'$ is identical due to the symmetry. If $A \cup C'$ induces a clique, then the claim follows so suppose that there exist two vertices $a \in A \cup C$ and $c \in A \cup C$ that are not adjacent in G'. Without loss of generality let $a \in A$ and $c \in C$ — this is due to the fact that $G'[A]$ and $G'[C]$ are cliques. Take any two vertices $b \in B$ and $b' \in B'$. Since G' is chordal, the cycle induced by a, b, b' and c has a chord in G'. Thus, there is an edge between b and b' in G'. Since b and b' are selected arbitrarily, $G'[B \cup B']$ is a clique. Note that a supergraph of G that has no edge between any vertex in A and any vertex in C and in which $B \cup B'$ induces a clique is chordal. Thus, by the minimality of G', $G'[A \cup C]$ consists of two cliques $G'[A]$ and $G'[C]$ with no edges between them, as required. □

Theorem 2 *There exists a graph G such that no interval supergraph G' of G satisfies $iw(G') = iw(G)$ and $ic(G') = ic(G)$.*

Proof Consider the graph $G = (A \cup B \cup B' \cup C, E(G))$ constructed at the beginning of this section. For any minimal chordal supergraph G' of G, we say that it is (A, C)-connected ((B, B')-connected) if $G'[A \cup C]$ is a clique ($G'[B \cup B']$ is a clique, respectively). Denote by $G_{(A,C)}$ (respectively, $G_{(B,B')}$) the minimal chordal supergraph of G that is (A, C)-connected ((B, B')-connected, respectively) but has no edge joining a vertex in B (respectively A) with a vertex in B' (respectively C).

Each interval graph is also chordal. On the other hand, both $G_{(A,C)}$ and $G_{(B,B')}$ are interval graphs.

Consider a minimal chordal supergraph G' of G. By Lemma 2, G' is (A, C)-connected, (B, B')-connected, both (A, C)- and (B, B')-connected or neither (A, C)- and (B, B')-connected. Since it is minimal, it is not (A, C)- and (B, B')-connected simultaneously. Also, it must be (A, C)- or (B, B')-connected for otherwise it is not chordal. This implies that G' equals either to $G_{(A,C)}$ or $G_{(B,B')}$. We have that G' is an interval graph and hence, by (6), we obtain

$$iw(G_{(A,C)}) - iw(G_{(B,B')}) = (|A| + |B| + |C|) - \left(|B| + \left|B'\right| + |C|\right) < 0,$$

and

$$ic(G_{(A,C)}) - ic(G_{(B,B')}) = |A|\,|C| - |B|\left|B'\right| > 0.$$

We conclude by noting that it is enough to consider minimal interval supergraphs when minimizing interval cost or interval width. □

5 Treewidth and Fill-in

We refer the reader e.g. to [20, 38] for a definition of the NP-complete problem of fill-in. The treewidth for a given graph G, denoted by $\mathtt{tw}(G)$, can be defined as the

the minimum k such that there exists a chordal supergraph G' of G such that the maximum clique $\omega(G')$ of G' has size at most $k+1$. The *fill-in* of G is the minimum m such that there exists a chordal supergraph of G that can be constructed by adding m edges to G. Hence, our corresponding combinatorial problem can be stated as follows:

Problem `TFM` **(*Treewidth & Fill-in Minimization*):**

Input: a graph G, integers k and c.
Question: does there exist a chordal supergraph G' of G such that $\omega(G') \leq k$ and $\left|E(G')\right| \leq c$?

By the same proof as in Theorem 2, we obtain that for some graphs there is no solution to Problem `TFM` in which $k = \texttt{tw}(G) - 1$ and $c = |E(G)| + \texttt{fill-in}(G)$.

Corollary 1 *There exists a graph G such that no chordal supergraph G' of G satisfies* $\texttt{tw}(G) = \omega(G') - 1$ *and* $\texttt{fill-in}(G) + |E(G)| = \left|E(G')\right|$.

6 Applications to Graph Searching

6.1 Formal Definitions

The following definitions of the *node search problem* are taken from or based on [16] and [17]. An *active search strategy* $\mathcal{S}$ for a graph G is a sequence of pairs

$$(A_0, Z_0), (A_1, Z_1), \ldots, (A_m, Z_m)$$

that satisfies the following *axioms*:

(i) $A_i \subseteq V(G)$ and $Z_i \subseteq V(G)$ for each $i \in \{0, \ldots, m\}$,
(ii) $A_0 = Z_0 = \emptyset$, $A_m = V(G)$ and $Z_m = \emptyset$,
(iii) (*placing/removing searchers*) For each $i \in \{1, \ldots, m\}$ there exist $v_i \in V(G)$ such that $\{v_i\} = A_i \setminus A_{i-1}$, $v_i \in Z_i$ and $Z_i \subseteq A_{i-1} \cup \{v_i\}$.
(vi) (*possible recontamination*) For each $i \in \{1, \ldots, m\}$, A_i consists of v_i and each vertex u such that each path connecting u to a vertex in $V(G) \setminus A_{i-1}$ has an internal vertex in Z_i.

An *inert search strategy* $\mathcal{S}$ is one that satisfies axioms (i),(ii),(iii) and:

(vi') (*possible recontamination*) For each $i \in \{1, \ldots, m\}$, A_i consists of v_i and each vertex u such that each path connecting u to v_i has an internal vertex in Z_{i-1}.

We say that, in the i-th step of $\mathcal{S}$, the vertices in Z_i are *guarded*, the vertices in A_i are *cleared* and the vertices in $V(G) \setminus A_i$ are *contaminated*. The *search cost* of a search strategy $\mathcal{S}$ is defined as

$$\gamma(\mathcal{S}) = \sum_{i=0}^{m} |Z_i|$$

and the number of searchers it uses is

$$\mathtt{ns}(\mathcal{S}) = \max\{|Z_i| \mid i \in \{0, \ldots, m\}\}.$$

Informally speaking, in an active search strategy recontamination can 'spread' from any contaminated vertex while in inert strategies recontamination can only spread from v_i.

We say that a strategy (active or inert) $\mathcal{S} = ((A_0, Z_0), \ldots, (A_m, Z_m))$ is *monotone* if $A_i \subseteq A_{i+1}$ for each $i \in \{1, \ldots, m-1\}$.

6.2 Consequences of our Results

We have the following equivalences:

Theorem 3 ([16]) *For each graph G, if $\mathcal{S}$ an active monotone search strategy of minimum cost, then $\gamma(\mathcal{S}) = ic(G)$.*

Theorem 4 ([26–28, 34]) *For each graph G, if $\mathcal{S}$ an active search strategy that uses the minimum number of searchers, then $\mathtt{ns}(\mathcal{S}) = \mathtt{pw}(G) + 1$.*

Hence we obtain the following equivalence:

Remark 1 An optimal solution to Problem PPM corresponds to an active search strategy that simultaneously minimizes the number of searchers and the cost.

For the second pair of parameters, we recall the following theorems.

Theorem 5 ([17]) *For each graph G, if $\mathcal{S}$ an inert monotone search strategy of minimum cost, then $\gamma(\mathcal{S}) = |E(G)| + \mathtt{fill\text{-}in}(G)$.*

Theorem 6 ([37]) *For each graph G, if $\mathcal{S}$ an inert search strategy that uses the minimum number of searchers, then $\mathtt{ns}(\mathcal{S}) = \mathtt{tw}(G) + 1$.*

This leads us to the following theorem:

Remark 2 An optimal solution to Problem TFM corresponds to an inert search strategy that simultaneously minimizes the number of searchers and the cost.

7 Conclusions and Open Problems

We note that the reason why Procedure IC is exponential is its first step, i.e., the computation of the profile-minimizing interval representation. Thus, having this as an input, our constructive method is of polynomial running time.

The first open problem we leave is the one of existence of a similar tradeoff between fill-in and treewidth to the one we have in Theorem 1. More particularly, is it

possible to find chordal supergraphs that approximate both parameters to within constant factors of their optimal values? Our approach used in Procedure `IC` most likely cannot be extended from interval graphs to chordal graphs as the latter have tree-like representations: the constant factor in our tradeoff relies on the fact that we iteratively 'reorganize' subintervals of the initial representation. More precisely, each modification performed for an interval (i, j) 'extends' the intervals intersecting the points i and j, while in case of chordal graphs we would have to deal with 'subtrees' of the corresponding representation. Since such subtrees have potentially many leaves (as opposed to just two represented previously by the endpoints i and j), we cannot ensure keeping constant width of the final representation.

A challenging open problem is the one that refers to the concept of recontamination in the graph searching games that has been posed in [17]: does recontamination help to obtain a minimum-cost inert search strategy? Formally, does there exist, for some graph G, an inert search strategy whose cost is smaller than $|E(G)| + \texttt{fill-in}(G)$? In yet other words, does there exist a graph for which an inert search strategy that minimizes the cost must necessarily allow for recontamination and as a result some vertex v is searched twice in step (iii), i.e., $v = v_i$ for two different indices i?

We remark that another example that shows that the problems of finding the minimum fill-in and the minimum clique size of an arbitrary graph are 'orthogonal' has been independently reported in [11] (see also [29] for some comments).

References

1. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. Algebraic Discrete Method **8**, 277–284 (1987)
2. Aschinger, M., Drescher, C., Gottlob, G., Jeavons, P., Thorstensen, E.: Structural decomposition methods and what they are good for. In: Thomas Schwentick and Christoph Dürr, editors, 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011), volume 9 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 12–28 (2011)
3. Barrière, L., Flocchini, P., Fomin, F.V., Fraigniaud, P., Nisse, N., Santoro, N., Thilikos, D.M.: Connected graph searching. Inf. Comput. **219**, 1–16 (2012)
4. Barrière, L., Flocchini, P., Fraigniaud, P., Santoro, N.: Capture of an intruder by mobile agents. In: SPAA'02: Proceedings of the Fourteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 200–209. ACM, New York (2002)
5. Billionnet, A.: On interval graphs and matrice profiles. RAIRO Rech. Opér. **20**(3), 245–256 (1986)
6. Bodlaender, H.L.: NC-Algorithms for Graphs with Small Treewidth. In: WG '88 14th International Workshop on Graph-Theoretic Concepts in Computer Science, Amsterdam, the Netherlands, June 15-17, 1988, Proceedings, pp. 1–10 (1988)

7. Bodlaender, H.L., Fomin, F.V.: Tree decompositions with small cost. Discret. Appl. Math. **145**(2), 143–154 (2005)
8. Bodlaender, H.L., Gilbert, J.R., Hafsteinsson, H., Kloks, T.: Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. J. Algorithms **18**, 238–255 (1995)
9. Bodlaender, H.L., Hagerup, T.: Tree Decompositions of Small Diameter. In: MFCS'98 23rd International Symposium on Mathematical Foundations of Computer Science 1998, Brno, Czech Republic, August 24-28, 1998, Proceedings, pp. 702–712 (1998)
10. Bodlaender, H.L., Nederlof, J.: Subexponential Time Algorithms for Finding Small Tree and Path Decompositions. In: Algorithms - ESA 2015 - 23Rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings, pp. 179–190 (2015)
11. Bodlaender, H.L., van der Gaag, L., Kloks, T.: Some remarks on minimum edge and minimum clique triangulations. Unpublished result
12. Dendris, N.D., Kirousis, L.M., Thilikos, D.M.: Fugitive-search games on graphs and related parameters. Theor. Comput. Sci. **172**(1-2), 233–254 (1997)
13. Dereniowski, D.: From pathwidth to connected pathwidth. SIAM J. Discrete Math. **26**(4), 1709–1732 (2012)
14. Dereniowski, D., Dyer, D.: On minimum cost edge searching. Theor. Comput. Sci. **495**, 37–49 (2013)
15. Dereniowski, D., Kubiak, W., Zwols, Y.: The complexity of minimum-length path decompositions. J. Comput. Syst. Sci. **81**(8), 1715–1747 (2015)
16. Fomin, F.V., Golovach, P.A.: Graph searching and interval completion. SIAM J. Discrete Math. **13**(4), 454–464 (2000)
17. Fomin, F.V., Heggernes, P., Telle, J.A.: Graph searching, elimination trees, and a generalization of bandwidth. Algorithmica **41**(2), 73–87 (2004)
18. Fomin, F.V., Thilikos, D.M.: An annotated bibliography on guaranteed graph searching. Theor Comput. Sci. **399**(3), 236–245 (2008)
19. Fraigniaud, P., Nisse, N.: Connected Treewidth and Connected Graph Searching. In: Proceedings of the 7Th Latin American Symposium on Theoretical Informatics (LATIN'06), LNCS, volume 3887, pp. 479–490. Valdivia, Chile (2006)
20. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman & Co., New York (1979)
21. Gibbs, N.E., Poole, W.G.Jr.: Tridiagonalization by permutations. Commun. ACM **17**(1), 20–24 (1974)
22. Gupta, A., Nishimura, N.: The complexity of subgraph isomorphism for classes of partial k-trees. Theor. Comput. Sci. **164**(1&2), 287–298 (1996)
23. Gupta, A., Nishimura, N., Proskurowski, A., Ragde, P.: Embeddings of k-connected graphs of pathwidth k. Discret. Appl. Math. **145**(2), 242–265 (2005)
24. Gustedt, J.: On the pathwidth of chordal graphs. Discrete Appl. Math. **45**(3), 233–248 (1993)
25. Kashiwabara, T., Fujisawa, T.: Np-Completeness of the Problem of Finding a Minimum-Clique-Number Interval Graph Containing a Given Graph as a Subgraph. In: Proceedings of the International Conference on Circuits and Systems, pp. 657–660 (1979)
26. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. Inf. Process. Lett. **42**(6), 345–350 (1992)
27. Kirousis, L.M., Papadimitriou, C.H.: Interval graphs and searching. Discrete Math. **55**, 181–184 (1985)
28. Kirousis, L.M., Papadimitriou, C.H.: Searching and pebbling. Theor. Comput. Sci. **47**(2), 205–218 (1986)
29. Kloks, T.: Treewidth, Computations and Approximations, volume 842 of Lecture Notes in Computer Science. Springer, Berlin (1994)
30. Kloks, T., Bodlaender, H.L.: Approximating Treewidth and Pathwidth of Some Classes of Perfect Graphs. In: Proceedings of the Third International Symposium on Algorithms and Computation, ISAAC '92, pp. 116–125. Springer, London (1992)
31. LaPaugh, A.S.: Recontamination does not help to search a graph. J. ACM **40**(2), 224–245 (1993)
32. Lengauer, T.: Black-white pebbles and graph separation. Acta Inf. **16**, 465–475 (1981)
33. Li, B., Moataz, F.Z., Nisse, N., Suchan, K.: Minimum size tree-decompositions. Discret. Appl. Math. **245**, 109–127 (2018)
34. Möhring, R.: Graph Problems Related to Gate Matrix Layout and PLA Folding. In: Mayr, E., Noltemeier, H., Syslo, M. (eds.) Computational Graph Theory, Computing Supplementum, vol. 7, pp. 17–51 (1990)
35. Robertson, N., Seymour, P.D.: Graph minors. II. algorithmic aspects of tree-width. J. Algorithm **7**(3), 309–322 (1986)

36. Sethi, R.: Complete register allocation problems. SIAM J. Comput. **4**(3), 226–248 (1975)
37. Seymour, P.D., Thomas, R.: Graph searching and a min-max theorem for tree-width. J. Comb. Theory Ser. B **58**(1), 22–33 (1993)
38. Yannakakis, M.: Computing the minimum fill-in is np-complete. SIAM J. Algebraic Discrete Method **2**(1), 77–79 (1981)
39. Yuan, J., Lin, Y., Liu, Y., Wang, S.: Np-completeness of the profile problem and the fill-in problem on cobipartite graphs. J. Math. Study **31**(3), 239–243 (1998)