A subexponential parameterized algorithm for Proper Interval Completion[⋆]

Ivan Bliznets[1], Fedor V. Fomin[1,2], Marcin Pilipczuk[2], and Michał Pilipczuk[2]

[1] St. Petersburg Department of Steklov Institute of Mathematics, Russia
[2] Department of Informatics, University of Bergen, Norway
(fomin|Marcin.Pilipczuk|Michal.Pilipczuk)@ii.uib.no

Abstract. In the Proper Interval Completion problem we are given a graph G and an integer k, and the task is to turn G using at most k edge additions into a proper interval graph, i.e., a graph admitting an intersection model of equal-length intervals on a line. The study of Proper Interval Completion from the viewpoint of parameterized complexity has been initiated by Kaplan, Shamir and Tarjan [FOCS 1994; SIAM J. Comput. 1999], who showed an algorithm for the problem working in $\mathcal{O}(16^k \cdot (n+m))$ time. In this paper we present an algorithm with running time $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$, which is the first subexponential parameterized algorithm for Proper Interval Completion.

1 Introduction

A graph G is an *interval graph* if it admits a model of the following form: each vertex is associated with an interval on the real line, and two vertices are adjacent if and only if the associated intervals overlap. If moreover the intervals can be assumed to be of equal length, then G is a *proper interval graph*; equivalently, one may require that no associated interval is contained in another. Interval and proper interval graphs appear naturally in molecular biology in the problem of *physical mapping*, where one is given a graph with vertices modeling contiguous intervals (called *clones*) in a DNA sequence, and the edges indicate which intervals overlap. Based on this information one would like to reconstruct the layout of the clones. We refer to [12] for further discussion on biological applications of (proper) interval graphs.

The biological motivation was the starting point of the work of Kaplan et al. [12], who initiated the study of (proper) interval graphs from the point of view of parameterized complexity. It is namely natural to expect that some information about overlaps will be lost, and hence the model will be missing a

⋆ The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959 and the Government of the Russian Federation (grant 14.Z50.31.0030). A full version of this paper is available at arXiv [4].

small number of edges. Thus we arrive at the problems of Interval Completion (IC) and Proper Interval Completion (PIC): given a graph G and an integer k, one is asked to add at most k edges to G to obtain a (proper) interval graph. Both problems are NP-hard [17], and hence it is natural to ask for an FPT algorithm parameterized by the number of additions. For PIC Kaplan et al. [12] presented an algorithm with running time $\mathcal{O}(16^k \cdot (n+m))$, while fixed-parameterized tractability of IC was proved much later by Villanger et al. [16]. Recently, Liu et al. [14] obtained an $\mathcal{O}(4^k + nm(n+m))$-time algorithm for PIC.

The approach of Kaplan et al. [12] is based on a characterization by *forbidden induced subgraphs*, also studied by Cai [5]: proper interval graphs are exactly graphs that are chordal (do not contain any induced cycle C_ℓ for $\ell \geq 4$) and additionally exclude three small graphs as induced subgraphs: a *claw*, a *tent*, and a *net* (see e.g. [2]). Thus, in PIC we may apply a basic branching strategy: Whenever a forbidden induced subgraph is encountered, we branch into several possibilities of how it is going to be destroyed in the optimal solution. A cycle C_ℓ can be destroyed only by adding $\ell - 3$ edges to triangulate it, and there are roughly $4^{\ell-3}$ different ways to do so. Since there is only a constant number of ways of destroying a small subgraph, the whole branching procedure runs in $c^k n^{\mathcal{O}(1)}$ time, for some constant c.

The approach via forbidden induced subgraphs has driven the research on the parameterized complexity of graph modification problems ever since the work of Cai [5]. Of particular importance was the work on polynomial kernelization; recall that a *polynomial kernel* for a parameterized problem is a polynomial-time preprocessing routine that reduces the size of the instance at hand to polynomial in the parameter. While many natural completion problems admit polynomial kernels, there are also examples where no such kernel exists under plausible complexity assumptions [13]. In particular, PIC admits a kernel with $\mathcal{O}(k^3)$ vertices which is computable in $\mathcal{O}(nm(kn+m))$ time [2], while the kernelization status of IC remains a notorious open problem.

The turning point came recently, when Fomin and Villanger [9] proposed an algorithm for Chordal Completion (aka Fill-in), that runs in *subexponential parameterized time*, more precisely $k^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$. As observed by Kaplan et al. [12], the approach via forbidden induced subgraphs leads to an FPT algorithm for Fill-in with running time $16^k n^{\mathcal{O}(1)}$. However, in order to achieve a subexponential running time one needs to completely abandon this route, as even branching on obstacles as small as, say, induced C_4-s, leads to running time $2^k n^{\mathcal{O}(1)}$. To circumvent this, Fomin and Villanger proposed the approach of gradually building the structure of a chordal graph in a dynamic programming manner. The crucial observation was that the number of 'building blocks' (in their case, *potential maximal cliques*) is subexponential in a YES-instance, and thus the dynamic program operates on a subexponential space of states.

This research direction was continued by Ghosh et al. [10] and by Drange et al. [7], who identified several more graph classes for which completion problems have subexponential parameterized complexity: threshold graphs, split graphs, pseudo-split graphs, and trivially perfect graphs. Let us remark here that problems admitting subexponential parameterized algorithms are very scarce, since for most natural parameterized problems existence of such algorithms can be refuted under the Exponential Time Hypothesis (ETH) [11]. Up to very recently, the only natural positive examples were problems on specifically constrained inputs, like H-minor free graphs [6] or tournaments [1]. Thus, completion problems admitting subexponential parameterized algorithms can be regarded as 'singular points on the complexity landscape'. Indeed, Drange et al. [7] complemented their work with a number of lower bounds excluding (under ETH) subexponential parameterized algorithms for completion problems to many related graphs classes, e.g. cographs.

Interestingly, threshold, trivially perfect and chordal graphs, which are currently our main examples, correspond to graph parameters *vertex cover*, *treedepth*, and *treewidth* in the following sense: the parameter is equal to the minimum possible maximum clique size in a completion to the graph class (± 1). It is therefore natural to ask if Interval Completion and Proper Interval Completion, which likewise correspond to *pathwidth* and *bandwidth*, also admit subexponential parameterized algorithms.

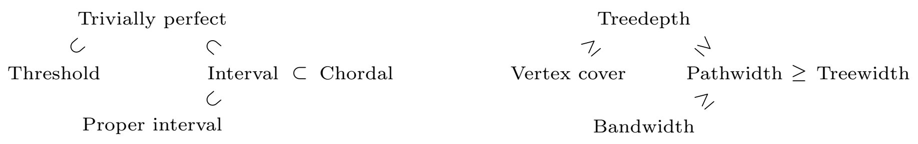

Fig. 1. Graph classes and corresponding graph parameters. Inequalities on the right side are with ± 1 slackness.

Our Results. In this paper we answer the question about Proper Interval Completion in affirmative by proving the following theorem:

Theorem 1. Proper Interval Completion *is solvable in time* $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$.

In a companion paper [3] we also present an algorithm for Interval Completion with running time $k^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$, which means that the completion problems for all the classes depicted on Fig. 1 in fact do admit subexponential parameterized algorithms. We now describe briefly our techniques employed to prove Theorem 1, and main differences with the work on interval graphs [3].

From a space-level perspective, both the approach of this paper and of [3] follows the route laid out by Fomin and Villanger in [9]. That is, we enumerate a subexponential family of potentially interesting building blocks, and then try to arrange them into a (proper) interval model with a small number of missing edges using dynamic programming (DP for short). In both cases, a natural candidate for this building block is the concept of a *cut*: given an interval model of a graph, imagine a vertical line placed at some position x that pins down intervals containing x. A *potential cut* is then a subset of vertices that becomes a cut in some minimal completion to a (proper) interval graph of cost at most k. The starting point of both this work and of [3] is enumeration of potential cuts.

Using different structural insights into the classes of interval and proper interval graphs, one can show that in both cases the number of potential cuts is at most $n^{\mathcal{O}(\sqrt{k})}$, and they can be enumerated efficiently. Since in the case of proper interval graphs we can start with a cubic kernel given by Bessy and Perez [2], this immediately gives $k^{\mathcal{O}(\sqrt{k})}$ potential cuts for the PIC problem. In the interval case the question of existence of a polynomial kernel is widely open, and the need of circumventing this obstacle causes severe complications in [3].

Afterwards the approaches diverge completely, as it turns out that in both cases the potential cuts are insufficient building blocks to perform dynamic programming, however for very different reasons. For Interval Completion the problem is that the cut itself does not define what lies on the left and on the right of it. Even worse, there can be an exponential number of possible left/right alignments when the graph contains many modules that neighbor the same clique. To cope with this problem, the approach taken in [3] remodels the dynamic programming routine so that, in some sense, the choice of left/right alignment is taken care of inside the dynamic program. However, this leads to extremely complicated definition of a DP state and its relations.

Curiously, in the proper interval setting the left/right choice can be easily guessed along with a potential cut at basically no extra cost. Hence, the issue causing the most severe problems in the interval case is non-existent. The problem, however, is in the *ordering* of intervals in the cut: while performing a natural left-to-right DP that builds the model, we need to ensure that intervals participating in a cut begin in the same order as they end. Therefore, apart from the cut itself and a partition of the other vertices into left and right, a state would also need to include the ordering of the vertices of the cut; as the cut may be large, we cannot afford constructing a state for every possible ordering.

Instead we remodel the dynamic program, this time by introducing two layers. We first observe that the troublesome ordering may be guessed expeditiously providing that the cut in question has only a sublinear in k number of incident edge additions. Hence, in the first layer of dynamic programming we aim at chopping the optimally completed model using such cheap cuts, and to conclude the algorithm we just need to be able to compute the best possible completed model between two border cuts that are cheap, assuming that all the intermediate cuts are expensive. This task is performed by the layer-two dynamic program. The main observation is that since all the intermediate cuts are expensive, there cannot be many disjoint such cuts and, consequently, the space between the border cuts is in some sense 'short'. As the border cuts can be large, it is natural to start partitioning the space in between 'horizontally' instead of 'vertically' — shortness of this space guarantees that the number of sensible 'horizontal' separations is subexponential. The horizontal partitioning method that we employ resembles the classic $\mathcal{O}^\star(10^n)$-time exact algorithm for bandwidth of Feige [8].

2 Preliminaries

In most cases, we follow standard graph notation.

For integers a, b, we denote $[a, b] = \{a, a+1, \dots, b\}$. An ordering σ of a subset $X \subseteq V(G)$ is an injective function $\sigma : X \to [1, |V(G)|]$, and an ordering of G is simply an ordering of $V(G)$. Note that an ordering of G is a bijection. We sometimes treat an ordering σ of $X \subseteq V(G)$ as an ordering of $G[X]$ as well, implicitly identifying $\sigma(X)$ with $[1, |X|]$ in the monotonous way.

We use n and m to denote the number of vertices and edges of the input graph, respectively. Moreover, for an input graph G we fix some arbitrary order $\preceq$ of $V(G)$ and with every ordering σ of $X = \{x_1 \prec x_2 \prec \dots \prec x_{|X|}\} \subseteq V(G)$ we associate a sequence $(\sigma(x_1), \sigma(x_2), \dots, \sigma(x_{|X|}))$. The *lexicographically minimum* ordering from some family of orderings of a fixed set X is the ordering with lexicographically minimum associated sequence.

A graph G is a *proper interval graph* if it admits an intersection model, where each vertex is assigned a closed interval on a line such that no interval is a proper subset of another one. In our work it is more convenient to use an equivalent combinatorial object, called an *umbrella ordering.*

Definition 2 (umbrella ordering). *Let G be a graph and $\sigma : V(G) \to [1, n]$ be an ordering. We say that σ satisfies the* umbrella property *for a triple $a, b, c \in V(G)$ if $ac \in E(G)$ and $\sigma(a) < \sigma(b) < \sigma(c)$ implies $ab, bc \in E(G)$. Furthermore, σ is an* umbrella ordering *if it fulfills the umbrella property for all $a, b, c \in V(G)$.*

It is known that a graph is a proper interval graph if and only if it admits an umbrella ordering [15]. Observe that we may equivalently define an umbrella ordering σ as such an ordering that for every $ab \in E(G)$ with $\sigma(a) < \sigma(b)$, the vertices in $[\sigma(a), \sigma(b)]$ in σ form a clique in G, or, alternatively, if and only if for every $a, a', b', b \in V(G)$ such that $\sigma(a) \le \sigma(a') < \sigma(b') \le \sigma(b)$, if $ab \in E(G)$, then also $a'b' \in E(G)$.

For a graph G, a *completion* of G is a set $F \subseteq \binom{V(G)}{2} \setminus E(G)$ such that $G + F := (V(G), E(G) \cup F)$ is a proper interval graph. The Proper Interval Completion problem asks for a completion of G of size not exceeding a given budget k. For a completion F of G and $v \in V(G)$, we denote by $F(v)$ the set of edges of F incident with v and for $X \subseteq V(G)$, we define $F(X) = \bigcup_{v \in X} F(v)$.

However, for our purposes it is more convenient to work with orderings rather than completions. Consider an ordering σ and define F^σ to be the set of these pairs $uv \notin E(G)$ such that there exists an edge $u'v' \in E(G)$ with $\sigma(u') \le \sigma(u) < \sigma(v) \le \sigma(v')$. It is straightforward to verify the following.

Lemma 3. *The graph $G^\sigma := G + F^\sigma$ is a proper interval graph, and σ is its umbrella ordering. Moreover, F^σ is the unique inclusion-wise minimal completion of G among completions F for which σ is an umbrella ordering of $G + F$.*

The *canonical ordering* of a graph G is the lexicographically minimum ordering among orderings σ with minimum possible $|F^\sigma|$. For a canonical ordering σ, the set F^σ is called the *canonical completion.* If additionally $|F^\sigma| \le k$, the canonical ordering σ is also called the *canonical solution.*

Our starting point for the proof of Theorem 1 is the polynomial kernel for Proper Interval Completion due to Bessy and Perez.

Theorem 4 ([2]). Proper Interval Completion *admits a kernel with* $\mathcal{O}(k^3)$ *vertices computable in time* $\mathcal{O}(nm(kn+m))$.

The algorithm of Theorem 1 starts with applying the kernelization algorithm of Theorem 4; all further computation will take $k^{\mathcal{O}(k^{2/3})}$ time, yielding the promised time bound. Hence, in the rest of the paper we assume that we are given a PIC instance (G,k) with $n = |V(G)| = \mathcal{O}(k^3)$, and we are looking for the canonical solution of G provided that (G,k) is a YES-instance. Moreover, by standard arguments we may assume that G is connected.

3 Expensive vertices

We first deal with vertices that are incident with many edges of F^σ. Formally, we set a threshold $\tau := (2k)^{1/3}$ and say that a vertex v is *expensive* with respect to σ if it is incident with more than τ edges of F^σ, and *cheap* otherwise. As there are at most $(2k)^{2/3} = \tau^2$ expensive vertices, we may afford guessing a lot of information about expensive vertices within the promised time bound.

More formally, we branch into $k^{\mathcal{O}(k/\tau)} = k^{\mathcal{O}(k^{2/3})}$ subcases corresponding to the guesses about the expensive vertices in the canonical solution σ. We consider all possible

– sets $V_\$ \subseteq V(G)$ of size at most τ^2 of expensive vertices w.r.t. σ, and
– for each $V_\$$, all possible quadruples (v, p_v, p_v^L, p_v^R), where $v \in V_\$$, and p_v, p_v^L, p_v^R are integers such that $p_v = \sigma(v)$, $p_v^L = \min\{\sigma(w) : w \in N_{G^\sigma}[v]\}$ and $p_v^R = \max\{\sigma(w) : w \in N_{G^\sigma}[v]\}$.

In each branch, we look for the canonical solution to the instance (G,k), assuming that the aforementioned guess is the correct one. The *correct branch* is the one where this assumption is indeed true.

In each branch, some consistency checks are in place. For instance, the mapping $v \mapsto p_v$ should be injective, $p_{v_1} < p_{v_2}$ should imply $p_{v_1}^L \leq p_{v_2}^L$ and $p_{v_1}^R \leq p_{v_2}^R$, etc. We omit the full description of these checks in this extended abstract.

Observe that, in the correct branch, a vertex $v \in V_\$$ has degree exactly $p_v^R - p_v^L$ in the graph G^σ, with its closed neighborhood placed on positions $[p_v^L, p_v^R]$. This motivates us to define the following

$$F_\$ = \{v_1v_2 : v_1, v_2 \in V_\$ \wedge v_1 \neq v_2 \wedge v_1v_2 \notin E(G) \wedge v_1 \in [p_{v_2}^L, p_{v_2}^R]\},$$
$$c_\$ = -|F_\$| + \sum_{v \in V_\$} \left((p_v^R - p_v^L) - \deg_G(v)\right).$$

Let us observe that $F_\$$ is the set comprising edges of F^σ with both endpoints in $V_\$$, i.e. $F_\$ = F^\sigma \cap \binom{V_\$}{2}$, and that $c_\$$ is the number of edges of F^σ incident with $V_\$$, i.e. $c_\$ = |F^\sigma(V_\$)|$. Both notions are meaningful for every branch.

Lemma 5. *Let σ' be an ordering of $V(G)$ and F be a completion of G such that (i) σ' is an umbrella ordering of $G+F$, and (ii) for every $v \in V_\$$, we have $\sigma'(v) = p_v$ and $\sigma'(N_{G+F}[v]) = [p_v^L, p_v^R]$. Then $F \cap \binom{V_\$}{2} = F_\$$ and $|F(V_\$)| = c_\$$.*

We infer that the guesses made so far impose a fixed cost of $c_\$$ edges and it is tempting to consider the remaining instance $(G \setminus V_\$, k - c_\$)$. However, the guessed values impose some constraints on this remaining instance. First, if $uv \in E(G)$ for some expensive v and cheap u, we need to have $\sigma(u) \in [p_v^L, p_v^R]$. Second, due to the umbrella property, for any expensive v, all vertices placed on positions $[p_v^L, p_v]$ become a clique, whereas no edge of G connects a vertex placed before position p_v^L and a vertex placed on or after position p_v; similar constraints are imposed for positions p_v and p_v^R.

Luckily, all these constraints can be modelled as (i) prescribing for each cheap u a set $\Sigma_u \subseteq [1, n]$ of allowed positions, and (ii) prescribing some pairs of positions to be necessarily adjacent or necessarily nonadjacent in the ordering σ. It turns out that the aforementioned additional constraints only slightly increase the technical level of further reasonings, and none of them adds any significant difficulty. Hence, in this extended abstract we ignore them, and assume that in the canonical ordering σ there are *no* expensive vertices.

4 Sections

For any position p in the canonical ordering σ we define a *section* $A_p = \{v \in V(G) : \sigma(v) < p\}$, and additionally $A_\infty = V(G)$. We are now going to show the vital combinatorial result: in the absence of expensive vertices, there is only subexponential number of candidates for sections of σ.

Theorem 6. *In $k^{\mathcal{O}(\tau)}$ time one can enumerate a family $\mathcal{S}$ of $k^{\mathcal{O}(\tau)}$ subsets of $V(G)$ such that every section of the canonical solution σ is in $\mathcal{S}$.*

The main idea in the proof of Theorem 6 is to investigate *twin classes* in graph G^σ. Recall that two vertices x and y are *true twins* if $N[x] = N[y]$; in particular, this implies that they are adjacent. The relation of being true twins is an equivalence relation, and equivalence classes of this relation are called *twin classes*. Observe that by the definition of the umbrella ordering, in σ the vertices of each twin class of G^σ occupy consecutive positions. We show the following bound on the number of candidates for twin classes.[3]

Theorem 7. *In $k^{\mathcal{O}(\tau)}$ time one can enumerate a family $\mathcal{T}$ of $k^{\mathcal{O}(\tau)}$ triples $(L, \Lambda, \sigma_\Lambda)$, where $L, \Lambda \subseteq V(G)$ and σ_Λ is an ordering of Λ, with the following property. For every twin class Λ of G^σ, if L is the set of vertices of G placed before Λ in the ordering σ, and $\sigma|_\Lambda$ is the ordering σ restricted to Λ, then $(L, \Lambda, \sigma|_\Lambda) \in \mathcal{T}$.*

We remark that it is easy to derive Theorem 6 from Theorem 7: We first output the section $V(G)$ and then, for each $(L, \Lambda, \sigma_\Lambda) \in \mathcal{T}$ and $p \in [1, n]$, we output $L \cup \{u \in \Lambda : \sigma_\Lambda(u) < p\}$. Observe that if a section $A_p \neq V(G)$ is consistent with

[3] We care about the order inside twin classes because inside a single twin class we may have different restrictions imposed by the guesses on expensive vertices made in the previous section.

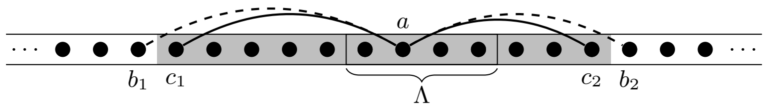

Fig. 2. The guessed vertices a, b_1, b_2, c_1 and c_2 with respect to a twin class Λ. The gray area denotes $N_{G^\sigma}[\Lambda]$.

σ, then A_p is output for the position p and the triple $(L, \Lambda, \sigma|_\Lambda) \in \mathcal{T}$ where Λ is the twin class of vertex $\sigma^{-1}(p)$.

To prove Theorem 7, we describe a branching algorithm that produces $k^{\mathcal{O}(\tau)}$ subcases and, in each subcase, produces one triple $(L, \Lambda, \sigma_\Lambda)$. We fix one twin class Λ of G^σ and argue that the algorithm in one of the branches produces $(L, \Lambda, \sigma|_\Lambda)$, where L is defined as in Theorem 7.

The algorithm first guesses the following five vertices, see also Fig. 2.

1. a is a vertex of Λ,
2. b_1 is the rightmost vertex outside $N_{G^\sigma}[\Lambda]$ in σ that lies before Λ, or $b_1 = \bot$ if no such vertex exists;
3. c_1 is the leftmost vertex of $N_{G^\sigma}[\Lambda]$ in σ;
4. c_2 is the rightmost vertex of $N_{G^\sigma}[\Lambda]$ in σ;
5. b_2 is the leftmost vertex outside $N_{G^\sigma}[\Lambda]$ in σ that lies after Λ, or $b_2 = \bot$ if no such vertex exists.

Moreover, for each $u \in \{a, b_1, b_2, c_1, c_2\} \setminus \{\bot\}$ the algorithm guesses $F^\sigma(u)$. This leads us to $k^{\mathcal{O}(\tau)}$ subcases, as all vertices of G are cheap. The crucial step in deducing the triple $(L, \Lambda, \sigma_\Lambda)$ is the following lemma (we take $N_{G^\sigma}[\bot] = \emptyset$).

Lemma 8. *In the branch where the guesses are correct, for every $u \in N_{G^\sigma}[a]$ the following holds*

1. *If $u \in N_{G^\sigma}[b_1]$ or $u \notin N_{G^\sigma}[c_2]$, then $u \notin \Lambda$ and u lies before Λ in σ;*
2. *If $u \in N_{G^\sigma}[b_2]$ or $u \notin N_{G^\sigma}[c_1]$, then $u \notin \Lambda$ and u lies after Λ in σ;*
3. *If none of the above happens, then $u \in \Lambda$.*

Proof. By the definition of b_1, b_2, c_1 and c_2, we have that every vertex $u \in \Lambda$ is in $N_{G^\sigma}[c_1]$ and $N_{G^\sigma}[c_2]$, but does not belong to $N_{G^\sigma}[b_1]$ and to $N_{G^\sigma}[b_2]$. Consequently, all vertices of Λ fall into the third category of the statement.

We now show that every vertex of $N_{G^\sigma}[a] \setminus \Lambda$ falls into one of the first two categories, depending on its position in the ordering σ. By symmetry, we may only consider a vertex $u \in N_{G^\sigma}[a] \setminus \Lambda$ that lies before Λ in σ. The umbrella property together with $a \notin N_{G^\sigma}[b_2]$ imply that $u \notin N_{G^\sigma}[b_2]$, and because $ac_1 \in E(G^\sigma)$, we have that $uc_1 \in E(G^\sigma)$. Consequently, u does not fall into the second category in the statement of the lemma.

As $u \notin \Lambda$ and $u \in N_{G^\sigma}[a]$, either $N_{G^\sigma}(u) \setminus N_{G^\sigma}[a]$ is not empty or $N_{G^\sigma}(a) \setminus N_{G^\sigma}[u]$ is not empty. In the first case, let $uw \in E(G^\sigma)$ but $aw \notin E(G^\sigma)$. Since also $ua \in E(G^\sigma)$, by the umbrella property it easily follows that w lies before u

in the ordering σ, so in particular before Λ. By the definition of b_1, b_1 exists and $\sigma(b_1) \geq \sigma(w)$. By the umbrella property, $b_1 u \in E(G^\sigma)$ and hence $u \in N_{G^\sigma}[b_1]$.

In the second case, assume $uw \notin E(G^\sigma)$ but $aw \in E(G^\sigma)$. Again, since $ua \in E(G^\sigma)$, by the umbrella property it easily follows that w lies after Λ in the ordering σ, so in particular after u. By the definition of c_2 and the existence of w, $c_2 \notin \Lambda$ and $\sigma(c_2) \geq \sigma(w)$. By the umbrella property, $c_2 u \notin E(G^\sigma)$ and $u \notin N_{G^\sigma}[c_2]$. Hence, u falls into the first category and the lemma is proven. $\square$

The knowledge of a and $F^\sigma(a)$ allows us to compute $N_{G^\sigma}[\Lambda] = N_{G^\sigma}[a]$. Lemma 8 allows us further to partition $N_{G^\sigma}[\Lambda]$ into Λ, the vertices of $N_{G^\sigma}(\Lambda)$ that lie before Λ in the ordering σ, and the ones that lie after Λ.

We are left with the vertices outside $N_{G^\sigma}[\Lambda]$. Let C be a connected component of $G \setminus N_{G^\sigma}[\Lambda]$. As no vertex of C is incident with Λ in G^σ, by the properties of an umbrella ordering we infer that all vertices of $N_G[C]$ lie before Λ in the ordering σ or all vertices of $N_G[C]$ lie after Λ. As G is assumed to be connected, $N_G(C)$ contains a vertex of $N_{G^\sigma}(\Lambda)$, and we can deduce whether $C \subseteq L$ or $L \cap C = \emptyset$.

Finally, as Λ is a twin class in G^σ, the ordering σ sorts Λ according to $\preceq$. Thus, $\sigma|_\Lambda$ depends only on the position $p = \min \sigma(\Lambda)$, which we simply guess.

We remark here that there are some slight difficulties if we have some additional constraint imposed by the guesses of the previous section. First, for a connected component C of $G \setminus N_{G^\sigma}[\Lambda]$ it may happen that $N_G(C) \subseteq V_\$$. In this case, however, we may deduce whether $C \subseteq L$ or $C \cap L = \emptyset$ from the guessed positions of the expensive vertices and the position p of the first vertex of Λ. Second, the ordering $\sigma|_\Lambda$ needs to respect the prescribed allowed positions Σ_u for $u \in \Lambda$. Luckily, with this constraint the task of finding $\sigma|_\Lambda$ boils down to a task of finding a lexicographically minimum perfect matching in an auxiliary bipartite graph, which is solvable in polynomial time.

5 Dynamic programming

Layer one: jumps and jump sets. Armed with Theorem 6, we proceed to design a dynamic programming algorithm that constructs the canonical solution σ. We first develop a natural left-to-right DP that splits the graphs G and G^σ 'vertically'. For each position p, we define the *jump* and *jump set* X_p as

$$\begin{aligned} \mathtt{jump}(p) &= \min\{q : q > p \wedge \sigma^{-1}(p)\sigma^{-1}(q) \notin E(G^\sigma)\}, \\ X_p &= \sigma^{-1}([p, \mathtt{jump}(p) - 1]) = A_{\mathtt{jump}(p)} \setminus A_p. \end{aligned}$$

See also Fig. 3. The separation property of a jump is provided by the following direct consequence of the property of the umbrella ordering.

Lemma 9. *For each $p \in [1, n]$, X_p is a clique in G^σ and no edge of G^σ connects a vertex of A_p with a vertex of $V(G) \setminus A_{\mathtt{jump}(p)}$.*

It is tempting to define a DP state as a 'jump set with a history' $J := (A, X)$, where its value is an ordering $\sigma_J := A \cup X \to [1, |A \cup X|]$ that first places the

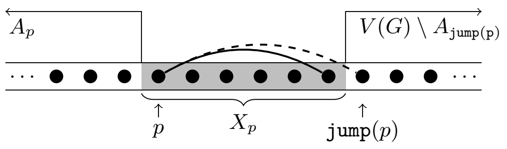

Fig. 3. A jump at position p and a corresponding jump set. The jump set X_p, denoted with gray, induces a clique in G^σ, and no edge of G^σ connects A_p with $V(G) \setminus A_{\texttt{jump}(p)}$.

vertices of A and then of X, with the intention that X is a jump set after the remaining vertices are placed (i.e., X is a clique in $G^{\sigma_J}[A \cup X]$, but no edge connects the first vertex of X in σ_J with $V(G) \setminus (A \cup X)$). However, this approach fails for the following reason: the internal ordering of the vertices of X affects both the ordering of A and the ordering of $V(G) \setminus (A \cup X)$, and hence needs to be stored in the DP state as well. Luckily, the ordering of X can be deduced from the knowledge of $F^\sigma(X)$, that is, the completion edges incident with X.

Lemma 10. *For each $p \in [1, n]$, if $u_1, u_2 \in X_p$ and $\sigma(u_1) \le \sigma(u_2)$, then*

$$N_{G^\sigma}(u_1) \cap A_p \supseteq N_{G^\sigma}(u_2) \cap A_p \text{ and } N_{G^\sigma}(u_1) \setminus A_{\texttt{jump}(p)} \subseteq N_{G^\sigma}(u_2) \setminus A_{\texttt{jump}(p)}.$$

It is easy to observe that, after satisfying the conditions of Lemma 10, we may proceed greedily. That is, the ordering σ sorts the vertices of X according to Lemma 10, breaking ties using order $\preceq$ to preserve lexicographical minimality.

It is not clear (if possible at all) how to provide a subexponential number of candidate orderings of X but we can do it in the case when $F^\sigma(X)$ is small. More precisely, we say that a jump set X is *cheap* if it is incident to at most $2k/\tau$ edges of F^σ, and *expensive* otherwise. Consequently, we may enumerate $k^{\mathcal{O}(2k/\tau)} = k^{\mathcal{O}(k^{2/3})}$ candidate triples (A, X, σ_X) for $(A_p, X_p, \sigma|_{X_p})$ for each *cheap* jump sets X_p. The layer one DP treats such triples as states, and finds for each such triple (A, X, σ_X) the optimal ordering $(A \cup X) \mapsto [1, |A \cup X|]$ that is consistent with σ_X. However, now layer one DP needs to perform a big task in a single step: namely, it needs to find the optimal way to arrange vertices between two consecutive cheap jumps. We delegate this task to the layer two DP, described in what follows.

Layer two: chains. Here we assume that we are given two of the layer one states (A^1, X^1, σ_X^1), (A^2, X^2, σ_X^2), with no cheap jump sets between X^1 and X^2, and we are to place the vertices of $(A^2 \cup X^2) \setminus A^1$ on positions $[|A^1|+1, |A^2 \cup X^2|]$ respecting σ_X^1 and σ_X^2.

We now derive a different 'horizontal' way of partitioning G and G^σ, based on the following definition. For any $q \in [1, n]$, consider the following sequence: $z_q(0) = q$ and $z_q(i+1) = \texttt{jump}(z_q(i))$ (taking $\texttt{jump}(\infty) = \infty$). Observe that:

Lemma 11. *For any $q > |A^1|$, it holds that $z_q(\tau) \ge |A^2|$.*

Proof. Observe that for each $i > 0$ with $z_q(i) < |A^2|$, the jump set $X_{z_q(i)}$ is expensive and, moreover, these jump sets are pairwise disjoint for different choices of i. Hence, there are less than τ such jump sets. $\square$

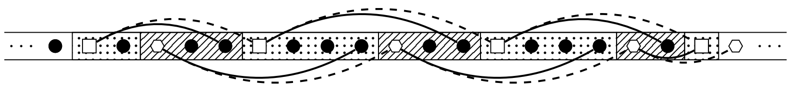

Fig. 4. The separation property provided by Lemma 12. The sequences $z_c(i)$ and $z_d(i)$ are denoted with rectangular and hexagonal shapes, respectively. The sets C_i and D_i are denoted boxes with dots and lines, respectively.

Moreover, observe that if we pick two positions c, d with $c \leq d \leq \mathtt{jump}(c)$ we have $z_c(i) \leq z_d(i) \leq z_c(i+1)$ for any $i \geq 0$.

The next immediate corollary of the definition of the umbrella property and the jump gives us the crucial separation property for the layer two DP (see Fig. 4).

Lemma 12. *For any positions c, d with $c \leq d \leq \mathtt{jump}(c)$, let $C_i = \sigma^{-1}([z_c(i), z_d(i)-1])$ and $D_i = \sigma^{-1}([z_d(i), z_c(i+1)-1])$. Then*

1. *sets C_i, D_i form a partition of $V(G) \setminus A_c$;*
2. *for every $i \geq 0$, both $C_i \cup D_i$ and $D_i \cup C_{i+1}$ are cliques in G^σ;*
3. *for every $j > i \geq 0$, there is no edge in G^σ between C_i and D_j;*
4. *for every $i > j+1 > 0$, there is no edge in G^σ between C_i and D_j.*

Lemma 12 allows us to define a layer two DP state consisting of sequences $z_c(i)$ and $z_d(i)$, up to minimum index i that satisfies $z_c(i) > |A^2|$, together with sections $A_{z_c(i)}$ and $A_{z_d(i)}$, for some choice of starting positions $|A^1| < c \leq d \leq \min(\mathtt{jump}(c), |A^2 \cup X^2|)$. In such a state, we ask for an optimal ordering of the sets $X^1 \cup X^2 \cup \bigcup_i C_i$ that respects the orderings σ_X^1 and σ_X^2, places the vertices of each C_i into positions $[z_c(i), z_d(i)-1]$ and turns each C_i into a clique. Note that Theorem 6, together with Lemma 11, gives us a bound $k^{\mathcal{O}(\tau^2)} = k^{\mathcal{O}(k^{2/3})}$ on the number of such states.

To compute the value of a layer two DP state, we guess the sequence $z_q(i)$ 'sandwiched' between $z_c(i)$ and $z_d(i)$ for some $c < q < d$, with the corresponding sections $A_{z_q(i)}$, and we glue the optimal values for states (c, q) and (q, d). If no such q exists, there are two special cases. If $c = d$, then the DP state in fact asks for $\sigma_X^1 \cup \sigma_X^2$. Finally, if $c + 1 = d$, then observe that all vertices of C_1 are adjacent to $\sigma^{-1}(d)$ and nonadjacent to $\sigma^{-1}(c)$. Hence, the vertices at positions c and d do not impose any constraints on the ordering of C_1, and, as a value for the state (c, d), we may use the value of the state $(\mathtt{jump}(c), \mathtt{jump}(d)) = (z_c(1), z_d(1))$, extended with the placement of the unique vertex of $A_d \setminus A_c$ at position c.

Overall, the described layer two DP allows us to perform a single step of the layer one DP in time $k^{\mathcal{O}(\tau^2)} = k^{\mathcal{O}(k^{2/3})}$. This concludes the proof of Theorem 1.

6 Conclusions and open problems

We have presented the first subexponential algorithm for Proper Interval Completion, running in time $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$. As many algorithms for completion problems in similar graph classes [3, 7, 9, 10] run in time $\mathcal{O}^\star(k^{\mathcal{O}(\sqrt{k})})$, it is tempting to ask for such a running time also in our case. The bottleneck in our approach is the trade-offs between the two layers of dynamic programming.

Also, observe that every $\mathcal{O}^\star(2^{o(\sqrt{k})})$-time algorithm for PIC would be in fact also a $2^{o(n)}$-time algorithm. Since existence of such an algorithm seems unlikely, we would like to ask for a $2^{\Omega(\sqrt{k})}$ lower bound, under the assumption of the Exponential Time Hypothesis. Note that no such lower bound is known for any other completion problem for related graph classes.

References

1. Alon, N., Lokshtanov, D., Saurabh, S.: Fast FAST. In: ICALP'13. LNCS, vol. 5555, pp. 49–58. Springer (2009)
2. Bessy, S., Perez, A.: Polynomial kernels for Proper Interval Completion and related problems. Information and Computation 231(0), 89 – 108 (2013)
3. Bliznets, I., Fomin, F.V., Pilipczuk, M., Pilipczuk, M.: A subexponential parameterized algorithm for interval completion. CoRR abs/1402.3473 (2014)
4. Bliznets, I., Fomin, F.V., Pilipczuk, M., Pilipczuk, M.: A subexponential parameterized algorithm for proper interval completion. CoRR abs/1402.3472 (2014)
5. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inf. Process. Lett. 58(4), 171–176 (1996)
6. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. J. ACM 52(6), 866–893 (2005)
7. Drange, P.G., Fomin, F.V., Pilipczuk, M., Villanger, Y.: Exploring subexponential parameterized complexity of completion problems. In: STACS'14 (2014)
8. Feige, U.: Coping with the NP-hardness of the graph bandwidth problem. In: SWAT 2000. pp. 10–19 (2000)
9. Fomin, F.V., Villanger, Y.: Subexponential parameterized algorithm for minimum fill-in. SIAM J. Comput. 42(6), 2197–2216 (2013)
10. Ghosh, E., Kolay, S., Kumar, M., Misra, P., Panolan, F., Rai, A., Ramanujan, M.: Faster parameterized algorithms for deletion to split graphs. In: SWAT'12. LNCS, vol. 7357, pp. 107–118. Springer (2012)
11. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. 63(4), 512–530 (2001)
12. Kaplan, H., Shamir, R., Tarjan, R.E.: Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. SIAM J. Comput. 28(5), 1906–1922 (1999)
13. Kratsch, S., Wahlström, M.: Two edge modification problems without polynomial kernels. Discrete Optimization 10(3), 193–199 (2013)
14. Liu, Y., Wang, J., Xu, C., Guo, J., Chen, J.: An effective branching strategy for some parameterized edge modification problems with multiple forbidden induced subgraphs. In: COCOON'13. LNCS, vol. 7936, pp. 555–566. Springer (2013)
15. Looges, P.J., Olariu, S.: Optimal greedy algorithms for indifference graphs. Computers and Mathematics with Applications 25(7), 15 – 25 (1993)
16. Villanger, Y., Heggernes, P., Paul, C., Telle, J.A.: Interval completion is fixed parameter tractable. SIAM J. Comput. 38(5), 2007–2020 (2009)
17. Yannakakis, M.: Computing the minimum fill-in is NP-complete. SIAM J. Alg. Disc. Meth. 2, 77–79 (1981)