Algorithmica manuscript No.
(will be inserted by the editor)

On cutwidth parameterized by vertex cover

Marek Cygan · Daniel Lokshtanov · Marcin Pilipczuk · Michał Pilipczuk · Saket Saurabh

the date of receipt and acceptance should be inserted later

Abstract We study the Cutwidth problem, where the input is a graph G, and the objective is find a linear layout of the vertices that minimizes the maximum number of edges intersected by any vertical line inserted between two consecutive vertices. We give an algorithm for Cutwidth with running time $O(2^k n^{O(1)})$. Here k is the size of a minimum vertex cover of the input graph G, and n is the number of vertices in G. Our algorithm gives an $O(2^{n/2} n^{O(1)})$ time algorithm for Cutwidth on bipartite graphs as a corollary. This is the first non-trivial exact exponential time algorithm for Cutwidth on a graph class where the problem remains NP-complete. Additionally, we show that Cutwidth parameterized by the size of the minimum vertex cover of the input graph does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. Our kernelization lower bound contrasts with the recent results of Bodlaender et al. [ICALP 2011, SWAT 2012] that both Treewidth and Pathwidth parameterized by vertex cover do admit polynomial kernels.

Keywords cutwidth · vertex cover parameterization · parameterized complexity · composition algorithms · polynomial kernel

A preliminary version of this paper appeared at International Symposium on Parameterized and Exact Computation, IPEC 2011.

Marek Cygan · Marcin Pilipczuk
Institute of Informatics, University of Warsaw, Banacha 2, 02-097, Warsaw, Poland. E-mail: {cygan,malcin}@mimuw.edu.pl

Daniel Lokshtanov
University of California, San Diego, La Jolla, CA 92093-0404, USA. E-mail: dlokshtanov@cs.ucsd.edu

Michał Pilipczuk
University of Bergen, Bergen, Norway. E-mail: michal.pilipczuk@ii.uib.no

Saket Saurabh
The Institute of Mathematical Sciences, Chennai - 600113, India. E-mail: saket@imsc.res.in

1 Introduction

In the CUTWIDTH problem we are given an n-vertex graph G together with an integer w. The task is to determine whether there exists a linear layout of the vertices of G such that any vertical line inserted between two consecutive vertices of the layout intersects with at most w edges (see Section 2 for a formal definition). The *cutwidth* ($cw(G)$) of G is the smallest w for which such a layout exists. The problem has numerous applications [10,23,24,29], ranging from circuit design [1,27] to protein engineering [4]. Unfortunately CUTWIDTH is NP-complete [18], and remains so even when the input is restricted to subcubic planar bipartite graphs [28,13] or split graphs where all independent set vertices have degree 2 [20]. On the other hand, the problem has a factor $O(\log^2(n))$-approximation on general graphs [26] and is polynomial time solvable on trees [32,12], graphs of constant treewidth and constant degree [31], threshold graphs [20], proper interval graphs [34] and bipartite permutation graphs [19].

In this article we study the complexity of computing cutwidth exactly on general graphs, where the running time is measured in terms of the size of the smallest vertex cover of the input graph G. A *vertex cover* of G is a vertex set S such that every edge of G has at least one endpoint in S. We show that CUTWIDTH can be solved in time $2^k n^{O(1)}$ where k is the size of the smallest vertex cover of G. An immediate consequence of our algorithm is that CUTWIDTH can be solved in time $2^{n/2} n^{O(1)}$ on bipartite graphs. This is the first non-trivial exact exponential time algorithm for CUTWIDTH on a graph class where the problem is NP-complete. Furthermore, our algorithm improves considerably over the previous best algorithm for CUTWIDTH parameterized by vertex cover [15], whose running time is $O(2^{2^{O(k)}} n^{O(1)})$ (however, it was not the focus of [15] to optimize the running time dependence on k).

Additionally, we show that CUTWIDTH parameterized by vertex cover does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. A *polynomial kernel* for CUTWIDTH parameterized by vertex cover is a polynomial time algorithm that takes as input a CUTWIDTH instance (G, w), where G has a vertex cover of size at most k and outputs an equivalent instance (G', w') of CUTWIDTH such that G' has at most $k^{O(1)}$ vertices. We show that unless NP $\subseteq$ coNP/poly such a kernelization algorithm can not exist. This contrasts with the recent results of Bodlaender et al. [7,8] that both TREEWIDTH and PATHWIDTH parameterized by the vertex cover number of the input graph do admit polynomial size kernels.

Context of our work. The CUTWIDTH problem is one of many *graph layout* problems, where the task is to find a permutation of the vertices of the input graph that optimizes a problem specific objective function. Graph layout problems, such as TREEWIDTH, BANDWIDTH and HAMILTONIAN PATH are not amenable to "branching" techniques, and hence the design of faster exact exponential time algorithms for these problems has resulted in several new and useful tools. For example, Karp's *inclusion-exclusion* based algorithm [25] for

Hamiltonian Path was the first application of inclusion-exclusion in exact algorithms. Another example is the introduction of *potential maximal cliques* as a tool for the computation of treewidth. Most graph layout problems (with the exception of Bandwidth) admit an $O(2^n n^{O(1)})$ time dynamic programming algorithm [2,21]. For several of these problems, faster algorithms with running time below $O(2^n)$ have been found [3,16,30], a stellar example is the recent algorithm by Björklund [3] for Hamiltonian Path. The Cutwidth problem is perhaps the best known graph layout problem for which a $O(2^n n^{O(1)})$ time algorithm is known, yet no better algorithm has been found. Hence, whether such an improved algorithm exists is a tantalizing open problem. While we do not resolve this problem in this article, we make considerable progress; hard instances of Cutwidth cannot contain any independent set of size cn for any $c > 0$.

Our choice of the vertex cover number as a relevant parameter for the Cutwidth problem originates in a recent interest in structural parameters (see e.g. [6–8,22]), in particular from the study of a very closely related problems of computing treewidth and pathwidth of the input graph. Note that both these problems can be easily seen to be AND-compositional when parameterized by the target treewidth or pathwidth of the graph (see [5] for discussion and relevant definitions) and an AND-composition, together with existence of a polynomial kernel, is now known to cause a collapse of the polynomial hierarchy [14]. This situation, together with the importance of the Treewidth and Pathwidth problems, motivated Bodlaender et al. [7,8] to investigate their other, stronger paramerizations. Among many other results, they have proven that both these problems admit a polynomial kernel with respect to the vertex cover of the graph, while such a kernel is unlikely if we parameterize by the deletion distance to a clique. We show that, although Cutwidth seems very similar to Pathwidth, these problems behave differently with respect to polynomial kernelization: Cutwidth does not admit a polynomial kernel when parameterized by the vertex cover number unless NP $\subseteq$ coNP/poly, which is known to imply a collapse of the polynomial hierarchy to its third level [33,11].

Organization of the paper. In Section 2 we present a dynamic programming algorithm which computes cutwidth in time $O(2^k n^{O(1)})$ for a given vertex cover of size k, whereas in Section 3 we show that Cutwidth parameterized by vertex cover does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. Section 4 is devoted to concluding remarks.

Notation. All graphs in this paper are undirected and simple. For a vertex $v \in V$ we define its neighbourhood $N_G(v) = \{u : uv \in E(G)\}$ and closed neighbourhood $N_G[v] = N_G(v) \cup \{v\}$. If G is clear from the context, we might omit the subscript. For $X \subseteq V$ we denote $N_G[X] = \bigcup_{v \in X} N_G(v) \setminus X$.

2 Faster Cutwidth Parameterized by Vertex Cover

In this section we show that given a graph $G = (C \cup I, E)$ such that C is a vertex cover of G of size k, we can compute the cutwidth of G in time $O(2^k n^{O(1)})$, using a dynamic programming approach. We start by showing that there always exists an optimal ordering of a specific form.

For an ordering $\sigma = v_1 \dots v_n$ of $V = C \cup I$ we define $V_i = \{v_j : j \leq i\}$. For vertices u and $v \in V$ we say that $u \leq_\sigma v$ if u occurs before v in σ. Denote by $\delta(V_i)$ the number of edges between V_i and $V \setminus V_i$. The cutwidth of the ordering, $cw_\sigma(G)$, is defined as the maximum of $\delta(V_i)$ for $i = 1, 2, \dots, |V| - 1$, and the cutwidth of the graph G is the minimum cutwidth over all possible orderings σ of V. The *rank* of a vertex v_i with respect to an ordering σ is denoted by $rank_\sigma(v_i)$ and it is equal to $|N(v_i) \setminus V_i| - |N(v_i) \cap V_i|$. Notice that $\delta(V_{i+1}) = \delta(V_i) + rank_\sigma(v_{i+1})$ and hence $\delta(V_i) = \sum_{j \leq i} rank_\sigma(v_j)$. Moving a vertex v_p *backward* to position q with $q < p$ results in the ordering

$$\sigma' = v_1 v_2 \dots v_{q-2} v_{q-1} \mathbf{v_p} v_q v_{q+1} \dots v_{p-2} v_{p-1} v_{p+1} v_{p+2} \dots v_n.$$

Moving v_p *forward* to a position q with $q > p$ results in the ordering

$$\sigma' = v_1 v_2 \dots v_{p-2} v_{p-1} v_{p+1} v_{p+2} \dots v_{q-2} v_{q-1} \mathbf{v_p} v_q v_{q+1} \dots v_n.$$

Notice that any vertex with odd degree must have (nonzero) odd rank. Moreover, moving a vertex backward cannot decrease its rank, whereas moving a vertex forward cannot increase its rank.

Lemma 1 *If moving v_p backward to position q results in an ordering σ' such that $rank_{\sigma'}(v_p) \leq 0$ then $cw_{\sigma'}(G) \leq cw_\sigma(G)$. If moving v_p forward to position q results in an ordering σ' such that $rank_{\sigma'}(v_p) \geq 0$ then $cw_{\sigma'}(G) \leq cw_\sigma(G)$.*

Proof Suppose moving v_p backward to position q results in an ordering σ' such that $rank_{\sigma'}(v_p) \leq 0$. For every non-negative integer i define V'_i to contain the first i vertices of σ'. Then, for every $i < q$ and $i \geq p$ we have $V'_i = V_i$ and hence $\delta(V'_i) = \delta(V_i)$. For every i such that $q \leq i < p$ we have that $V'_i = V_{i-1} \cup \{v_p\}$. Observe that for any other vertex v_j, $j \neq p$, $rank_{\sigma'}(v_j) \leq rank_\sigma(v_j)$, while $rank_{\sigma'}(v_p) \leq 0$. Thus $\delta(V'_i) = rank_{\sigma'}(v_p) + \sum_{j \leq i-1} rank_{\sigma'}(v_j) \leq \delta(V_{i-1})$ and $cw_{\sigma'}(G) \leq cw_\sigma(G)$. The proof that if moving v_p forward to position q results in an ordering σ' such that $rank_{\sigma'}(v_p) \geq 0$ then $cw_{\sigma'}(G) \leq cw_\sigma(G)$ is analogous. □

Lemma 1 allows us to rearrange optimal orderings. Let σ be an optimal cutwidth ordering of G, $c_1 c_2 \dots c_k$ be the ordering which σ imposes on C and $C_i = \{c_1, \dots, c_i\}$ for every i. Observe that if u and v are both in I then moving u does not affect the rank of v. In particular, if moving u yields the ordering σ', then $rank_{\sigma'}(v) = rank_\sigma(v)$. For every vertex $u \in I$ with odd degree and $rank_\sigma(u) < 0$ we move u backward to the leftmost position where u has rank -1. For every vertex $u \in I$ with odd degree and $rank_\sigma(u) > 0$ we move u forward to the rightmost position where u has rank 1. For every vertex of the set I with even degree we move it (forward or backward) to the rightmost position where u has rank 0. This results in an optimal cutwidth ordering σ' with the following properties.

1. For every vertex $v \in I$ of even degree $rank_{\sigma'}(v) = 0$ and every vertex $v \in I$ of odd degree $rank_{\sigma'}(v) \in \{-1, 1\}$.
2. For every vertex $v \in I$ such that $rank_{\sigma'}(v) \geq 0$ and $c_i \in C$ we have $c_i \leq_{\sigma'} v$ if and only if $|N(v) \cap C_i| \leq |N(v) \setminus C_i|$.
3. For every vertex $v \in I$ such that $rank_{\sigma'}(v) < 0$ and $c_i \in C$ we have $c_i \leq_{\sigma'} v$ if and only if $|N(v) \cap C_{i-1}| < |N(v) \setminus C_{i-1}|$.

Define I'_0 and I'_k to be the set of vertices in I appearing before c_1 and after c_k in σ', respectively. For i between 1 and $k-1$ we denote I'_i the set of vertices in I appearing between c_i and c_{i+1} in σ'. For any i, if I'_i contains any vertices of rank -1, we move them backward to the position right after c_i. This results in an ordering σ'' where for every i, all the vertices of I'_i with negative rank appear before all the vertices of I'_i with non-negative rank. By Lemma 1 and the fact that moving a vertex from independent set does not affect the rank of another vertex from the independent set we have that σ'' is still an optimal cutwidth ordering. Also, σ'' satisfies the properties $1-3$. We say that an ordering σ is *C-good* if it satisfies properties $1-3$ and orders the vertices between vertices of C in such a way that all vertices of negative rank appear before all vertices of non-negative rank. The construction of σ'' from an optimal ordering σ proves the following lemma.

Lemma 2 *Let $G = (C \cup I, E)$ be a graph and C be a vertex cover of G. There exists an optimal cutwidth ordering σ of G which is C-good.*

In a C-good ordering σ, consider a position i such that $c_i \in C$. Because of the properties of a C-good ordering we can essentially deduce $V_i \cap I$ from $V_i \cap C$ and the vertex c_i. We will now formalize this idea. For a set $S \subseteq C$ and vertex $v \in S$ we define the set $X(S, v) \subseteq I$ as follows. A vertex $u \in I$ of even degree is in $X(S, v)$ if $|N(u) \cap S| > |N(u) \setminus S|$. A vertex $u \in I$ of odd degree is in $X(S, v)$ if $|N(u) \cap (S \setminus \{v\})| > |N(u) \setminus (S \setminus \{v\})|$. Now we define the set $Y(S, v)$. A vertex $u \in I$ is in $Y(S, v)$ if $uv \in E$ and $|(N(u) \setminus \{v\}) \cap S| = |(N(u) \setminus \{v\}) \setminus S|$. Note that the vertices in $Y(S, v)$ have odd degrees and $Y(S, v)$ is disjoint with $X(S, v)$. We refer to Figure 1 for an illustration. The following observation follows directly from the properties of a C-good ordering.

Observation 3 *In a C-good ordering σ let i be an integer such that $c_i \in C$ and let $S = V_i \cap C$. Then $X(S, c_i) \subseteq V_i \cap I \subseteq X(S, c_i) \cup Y(S, c_i)$.*

A *prefix ordering* ϕ is a set $V_\phi \subseteq C \cup I$ together with an ordering of V_ϕ. The *size* of the prefix ordering ϕ is just $|V_\phi|$. Similarly to normal orderings we define $V_i^\phi = \{v_1 \dots v_i\}$. Let $c_1 c_2 \dots c_{|V_\phi \cap C|}$ be the ordering imposed on $V_\phi \cap C$ by ϕ, and for every $i \leq |V_\phi \cap C|$ we set $C_i^\phi = \{c_1, \dots, c_i\}$. The *rank* of a vertex $v \in V_\phi$ with respect to ϕ is defined as $rank_\phi(v_i) = |N(v_i) \setminus V_i^\phi| - |N(v_i) \cap V_i^\phi|$. We now extend the notion of being C-good from orderings of G to prefix

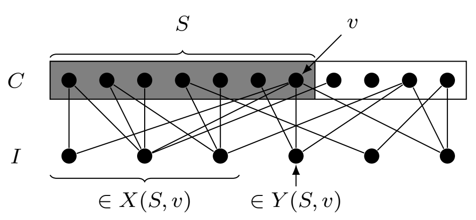

Fig. 1 Illustration of the definition of the sets $X(S,v)$ and $Y(S,v)$.

orderings of G in such a way that that the restriction of any C-good ordering σ of G to the first t vertices, where $v_t \in C$, must be C-good. We say that a prefix ordering $\phi = v_1 \dots v_t$ of size t with $v_t \in C$ is *C-good* if the following conditions are satisfied.

1. For every vertex $v \in V_\phi \cap I$ of even degree, $rank_\phi(v) = 0$ and for every vertex $v \in V_\phi \cap I$ of odd degree, $rank_\phi(v) \in \{-1, 1\}$.
2. $X(V_\phi \cap C, v_i) \subseteq V_\phi \cap I \subseteq X(V_\phi \cap C, v_i) \cup Y(V_\phi \cap C, v_i)$
3. For every vertex $v \in X(V_\phi \cap C, c_i)$ such that $rank_\phi(v) \geq 0$ and $c_i \in V_\phi \cap C$ we have $c_i \leq_\phi v$ if and only if $|N(v) \cap C_i^\phi| \leq |N(v) \setminus C_i^\phi|$.
4. For every vertex $v \in X(V_\phi \cap C, c_i)$ such that $rank_\phi(v) < 0$ and $c_i \in V_\phi \cap C$ we have $c_i \leq_\phi v$ if and only if $|N(v) \cap C_{i-1}^\phi| < |N(v) \setminus C_{i-1}^\phi|$.
5. Between two vertices $c_i, c_{i+1} \in C \cap V_\phi$, all vertices with $rank_\phi(v) < 0$ come before all vertices with $rank_\phi(v) \geq 0$.

Comparing the properties of C-good orderings and C-good prefix orderings it is easy to see that the following lemma holds.

Lemma 4 *Let $\sigma = v_1 \dots v_n$ be a C-good ordering and let ϕ be the restriction of σ to the first t vertices, such that $v_t \in C$. Then ϕ is a C-good prefix ordering.*

For a prefix ordering ϕ define the cutwidth of G with respect to ϕ to be $cw_\phi(G) = \max_{i \leq |V_\phi|} \delta(V_i^\phi)$. For a subset S of C and vertex $v \in S$, define $T(S,v)$ to be the minimum value of $cw_\phi(G)$ where the minimum is taken over all C-good prefix orderings ϕ with $V_\phi \cap C = S$ and v being the last vertex of ϕ. Notice that property 5 of C-good prefix orderings implies that in a C-good prefix ordering ϕ there must be some i with $v_i \in C \cap V_\phi$ such that $cw_\phi(G) = \delta(V_i)$ or $cw_\phi(G) = \delta(V_{i-1})$. Also, notice that for any set $S \subseteq C$ and vertices $u, v \in S$ we have that $X(S \setminus \{v\}, u) \subseteq X(S,v)$ and $Y(S \setminus \{v\}, u) \subseteq X(S,v)$. Finally, observe that for any set $S \subseteq C$ and vertex $v \in S$, every vertex $u \in Y(S,v)$ is adjacent to v and satisfies $|(N(u) \setminus \{v\}) \cap S| = |(N(u) \setminus \{v\}) \setminus S|$. Thus, for any set $I' \subseteq I$ with $X(S,v) \subseteq I' \subseteq X(S,v) \cup Y(S,v)$ the value of $\delta(S \cup I')$ depends only on $|Y(S,v) \cap I'|$ and not on $Y(S,v) \cap I'$ in general. We let $Y_i(S,v)$ be an arbitrary subset of $Y(S,v)$ of size i. The discussion above yields that the following recurrence holds for $T(S,v)$, where $S \subseteq C$ and $v \in S$.

$$T(S,v) = \min_{u \in S} \min_{0 \leq i \leq |Y(S,v)|} \max \left\{ \begin{array}{r} \delta(S \cup X(S,v) \cup Y_i(S,v)) \\ \delta((S \setminus \{v\}) \cup X(S,v) \cup Y_i(S,v)) \\ T(S \setminus \{v\}, u) \end{array} \right\} .$$

Observe that $cw(G) = \min_{v \in C} T(S, v)$ because in any ordering σ all vertices of I appearing after the last vertex of C must have negative rank. Thus the recurrence above naturally leads to a dynamic programming algorithm for Cutwidth running in time $O(2^k n^{O(1)})$. This proves the main theorem of this section.

Theorem 5 *There is an algorithm that given a graph $G = (C \cup I, E)$ such that C is a vertex cover of G, computes the cutwidth of G in running time $O(2^{|C|}(|C| + |I|)^{O(1)})$. Thus,* Minimum Cutwidth *on bipartite graphs can be solved in time $O(2^{n/2} n^{O(1)})$, where n is the number of vertices of the input graph.*

3 Kernelization Lower Bound

In this section we show that Cutwidth parameterized by vertex cover does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.

3.1 The auxiliary problem

We begin by introducing an auxiliary problem, namely Hypergraph Minimum Bisection. Let $H = (V, E)$ be a multihypergraph with $|V| = n$, where n is even. A *bisection* of V is a colouring $\mathcal{B} : V \to \{0, 1\}$ such that $|\mathcal{B}^{-1}(0)| = |\mathcal{B}^{-1}(1)| = n/2$. For a hyperedge e let us define the cost of e with respect to a bisection $\mathcal{B}$ as $cost(e, \mathcal{B}) = \min\left(\left|e \cap \mathcal{B}^{-1}(0)\right|, \left|e \cap \mathcal{B}^{-1}(1)\right|\right)$. The cost of a bisection is defined as the sum of the contributions of the hyperedges, i.e., $cost(\mathcal{B}) = \sum_{e \in E} cost(e, \mathcal{B})$.

Hypergraph Minimum Bisection **Parameter:** n
Input: Multihypergraph H with n vertices, where n is even; an integer k
Question: Does there exist a bisection of H with cost at most k?

In the case when all the hyperedges are in fact edges (have cardinalities 2) and there are no multiedges, the problem is equivalent to the classical Minimum Bisection problem. As Minimum Bisection is NP-hard, Hypergraph Minimum Bisection is also NP-hard, so NP-complete as well.

The goal now is to prove that Cutwidth parameterized by the size of vertex cover does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly. We do it in two steps. First, using the OR-distillation technique of Bodlaender et al. [5] (with the backbone theorem proven by Fortnow and Santhanam [17]) we prove that Hypergraph Minimum Bisection does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly. Second, we present a parameterized reduction from Hypergraph Minimum Bisection to Cutwidth parameterized by vertex cover.

3.2 No polynomial kernel for Hypergraph Minimum Bisection

We use the OR-distillation technique (first introduced by Bodlaender et al. [5]) put into the framework called *cross-composition* [6]. Let us recall the crucial definitions.

Definition 6 (Polynomial equivalence relation [6]) An equivalence relation $\mathcal{R}$ on Σ^* is called a *polynomial equivalence relation* if (1) there is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $\mathcal{R}(x, y)$ in $(|x|+|y|)^{O(1)}$ time; (2) for any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into at most $(\max_{x\in S} |x|)^{O(1)}$ classes.

Definition 7 (Cross-composition [6]) Let $L \subseteq \Sigma^*$ and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L *cross-composes* into Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given t strings $x_1, x_2, \dots x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that (1) $(x^*, k^*) \in Q$ iff $x_i \in L$ for some $1 \le i \le t$; (2) k^* is bounded polynomially in $\max_{i=1}^{t} |x_i| + \log t$.

Theorem 8 ([6], Theorem 9) *If $L \subseteq \Sigma^*$ is NP-hard under Karp reductions and L cross-composes into the parameterized problem Q that has a polynomial kernel, then NP $\subseteq$ coNP/poly.*

Lemma 9 Hypergraph Minimum Bisection*, parameterized by the size of the universe, does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly.*

Proof As Minimum Bisection is NP-hard under Karp reductions, it suffices to prove that it cross-composes into the Hypergraph Minimum Bisection problem, parameterized by n — the size of the universe. Let R be an equivalence relation on Σ^* defined as follows:

- all words that do not correspond to instances of Minimum Bisection form one equivalence class;
- all the well-formed instances are partitioned into equivalence classes having the same number of vertices, the same number of edges and the same demanded cost of the bisection.

It is straightforward that R is a polynomial equivalence relation. Therefore, we can assume that the composition algorithm is given a sequence of instances $(G_0, k), (G_1, k), \dots, (G_{t-1}, k)$ of Minimum Bisection with $|V(G_i)| = n$ and $|E(G_i)| = m$ for all $i = 0, 1, \dots, t-1$ (n is even). Moreover, by copying some instances if necessary we can assume without losing generality that $t = 2^l$ for some integer l. Note that in this manner we do not increase the order of $\log t$.

We now proceed to the construction of the composed Hypergraph Minimum Bisection instance (H, K). Let $N = 2m \cdot ((l+2)2^{l-1} - 1) + 2k + 1$ and $M = N(l^2 - l) + N$. We begin by creating two sets of vertices A_0 and A_1, each of size $2nl$. We introduce each set A_0, A_1 as a hyperedge of the constructed hypergraph M times.

Then, we introduce $2l$ vertices s_i^0, s_i^1 for $i = 0, 1, \ldots, l-1$ and denote the set of all these vertices by S. For every $i < j$ we put N times each hyperedge $\{s_i^0, s_j^0\}$, $\{s_i^0, s_j^1\}$, $\{s_i^1, s_j^0\}$, $\{s_i^1, s_j^1\}$. Thus, the hypergraph induced by S is a clique without a matching, repeated N times. Furthermore, for every $p = 0, 1$ and $i = 0, 1, \ldots, l-1$ we construct a set S_i^p of $n-1$ vertices and put $S_i^p \cup \{s_i^p\}$ as a hyperedge of the constructed hypergraph M times.

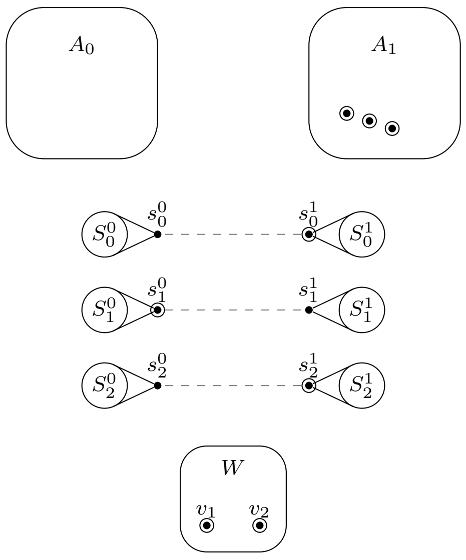

Fig. 2 The constructed hypergraph H for $l = 3$; encircled vertices indicate the hyperedge e^0 constructed for $e = v_1^5 v_2^5 \in E(G_5)$.

Now, we construct a set of n vertices $v_1, v_2, \ldots, v_n$ and denote it by W. For every instance G_a we arbitrarily choose an ordering of its vertices and denote it by $v_1^a, v_2^a, \ldots, v_n^a$. Let $b_{l-1}b_{l-2}\ldots b_1 b_0$ be the binary representation of a, with trailing zeroes added so that its length is equal to l. For every edge $e = v_g^a v_h^a \in E(G_a)$ we create two hyperedges:

- e^0, consisting of vertices v_g, v_h, $s_i^{b_i}$ for all $i = 0, 1, \ldots, l-1$ and l vertices from A_1, chosen arbitrarily;
- e^1, consisting of vertices v_g, v_h, $s_i^{1-b_i}$ for all $i = 0, 1, \ldots, l-1$ and l vertices from A_0, chosen arbitrarily.

Finally, we set the expected cost of the bisection to $K = M - 1 = N(l^2 - l) + 2m \cdot ((l+2)2^{l-1} - 1) + 2k$. We refer to Figure 2 for an illustration.

Assume that some graph G_a has a bisection $\mathcal{B}$ having cost at most k. Let $b_{l-1}b_{l-2}\ldots b_1 b_0$ be the binary representation of a, as in the previous paragraph. We now construct a bisection $\mathcal{B}'$ of H as follows:

- for each $u \in A_0$ we set $\mathcal{B}'(u) = 0$, for each $u \in A_1$ we set $\mathcal{B}'(u) = 1$;

- for each $u \in S_i^p \cup \{s_i^p\}$ for $p = 0, 1$, $i = 0, 1, \ldots, l-1$ we set $\mathcal{B}'(u) = p + b_i \pmod 2$;
- for each $v_j \in W$ we set $\mathcal{B}'(v_j) = \mathcal{B}(v_j^a)$.

Observe that $\mathcal{B}'$ bisects each of the sets $A_0 \cup A_1$, S and W, so it is a bisection. We now prove that its cost is at most K. Let us count the contribution to the cost from every hyperedge of H.

Each copy of the hyperedges A_0, A_1 and $S_i^p \cup \{s_i^p\}$ for $p = 0, 1$, $i = 0, 1, \ldots, l-1$ has zero contribution, as it is monochromatic. The edges of $H[S]$ have contribution 0 or 1, depending on whether the endpoints are coloured in the same or in a different way in $\mathcal{B}'$. There are l vertices s_i^p that map to 0 in $\mathcal{B}'$ and l that map to 1, so there are l^2 pairs of vertices coloured in a different way. Between every pair of vertices there are N edges, apart from the pairs (s_i^0, s_i^1). Note that all these pairs are coloured differently; therefore, there are exactly $N(l^2 - l)$ edges in $H[S]$ contributing 1 to the cost.

Take $c \in \{0, 1, \ldots, t-1\}$ such that $c \neq a$. Let $d_{l-1}d_{l-2}\ldots d_0$ be the binary representation of c. For $e \in E(G_c)$ let us count the contribution to $cost(\mathcal{B}')$ of the hyperedges e^0 and e^1. Suppose that $q = |\{i : b_i \neq d_i\}| > 0$. Among vertices of e^0, l from A_1 are coloured 1, q from S are coloured 1 as well and $l - q$ from S are coloured 0. In total, we have $l + q$ vertices coloured 1 and $l - q$ coloured 0, so regardless of the colouring of the remaining two vertices from W, the contribution is equal to the number of vertices coloured 0 in e^0, namely $l - q + |e^0 \cap W \cap \mathcal{B}'^{-1}(0)|$. Analogously, the contribution of the hyperedge e^1 is equal to the number of vertices of e^1 coloured 1, namely $l - q + |e^1 \cap W \cap \mathcal{B}'^{-1}(1)|$. As there are exactly two vertices in $e^0 \cap W = e^1 \cap W$, $cost(e^0, \mathcal{B}') + cost(e^1, \mathcal{B}') = 2(l - q) + 2$. Thus, the total contribution of hyperedges e^0, e^1 for $e \in E(G_c)$ is equal to $2m(l - q) + 2m$.

Now we count the contribution of the edges e^0 and e^1 for $e \in E(G_a)$. Analogously as in the previous paragraph, both edges e^0, e^1 contain l vertices coloured 0, l vertices coloured 1 plus two vertices from W. If both these vertices are coloured in the same way, the sum of the contributions of e^0 and e^1 is equal to $2l$; however, if the vertices are coloured differently, the sum is equal to $2l+2$. As the cost of bisection $\mathcal{B}$ was at most k, the total contribution of edges e^0, e^1 for $e \in E(G_a)$ is at most $2ml + 2k$.

Finally, we sum up the contributions:

$$\begin{aligned}
cost(\mathcal{B}') &\leq N(l^2 - l) + 2m \sum_{q=1}^{l} (l - q + 1) \binom{l}{q} + 2ml + 2k \\
&= N(l^2 - l) + 2m \cdot (2^l - 1) + 2m \sum_{q=0}^{l} (l - q) \binom{l}{q} + 2k \\
&= N(l^2 - l) + 2m \cdot (2^l - 1) + 2ml2^{l-1} + 2k \\
&= N(l^2 - l) + N - 1 = K.
\end{aligned}$$

We proceed to the second direction. Assume that we have a bisection $\mathcal{B}'$ of H such that $cost(\mathcal{B}') \leq K$. Observe that as $M > K$, both the sets A_0, A_1

are monochromatic with respect to $\mathcal{B}'$. Moreover, they have to be coloured differently, as they contain more than half of the vertices of the graph in total. Without losing generality we can assume that A_0 is coloured in colour 0, while A_1 is coloured in colour 1, by flipping the colours if necessary.

Now consider the set $S_p^i \cup \{s_p^i\}$ for $p = 0, 1$, $i = 0, 1, \ldots, l-1$. Analogously as in the previous paragraph, $S_p^i \cup \{s_p^i\}$ has to be monochromatic. Furthermore, observe that exactly l such sets have to be coloured 0 in $\mathcal{B}'$ and the same number have to be coloured 1, as every set $S_p^i \cup \{s_i^p\}$ contains the same number of vertices as the set W and $\mathcal{B}'$ is a bisection. Therefore, $\mathcal{B}'$ has to bisect each of the sets $A_0 \cup A_1$, S and W.

Exactly l vertices s_i^p are coloured 0 in $\mathcal{B}'$ and exactly l are coloured 1. Let r be the number of indices i, such that s_i^0 and s_i^1 are coloured differently. Observe that analogously to our previous arguments, the contribution of the edges of $H[S]$ to $cost(\mathcal{B}')$ is equal to $N(l^2 - r) = N(l^2 - l) + N(l - r)$. If $r < l$, then $cost(\mathcal{B}') \geq N(l^2 - l) + N > K$, a contradiction. Therefore, all the pairs (s_i^0, s_i^1) are coloured differently.

Let a be a number with binary representation $\mathcal{B}'(s_{l-1}^0)\mathcal{B}'(s_{l-2}^1)\ldots\mathcal{B}'(s_0^1)$. Consider a bisection $\mathcal{B}$ of G_a defined as follows: $\mathcal{B}(v_i^a) = \mathcal{B}'(v_i)$. We claim that the cost of $\mathcal{B}$ is at most k. Indeed, the same computations as in the previous part of the proof show that

$$cost(\mathcal{B}') = N(l^2 - l) + 2m((l+2)2^{l-1} - 1) + 2cost(\mathcal{B})$$

Therefore, as $cost(\mathcal{B}') \leq K$, then $cost(\mathcal{B}) \leq k$. □

3.3 From Hypergraph Minimum Bisection to Cutwidth

Let us briefly recall the notion of polynomial parameter transformations.

Definition 10 ([9]) Let P and Q be parameterized problems. We say that P is polynomial parameter reducible to Q, written $P \leq_p Q$, if there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ and a polynomial p, such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ the following holds: $(x, k) \in P$ iff $(x', k') = f(x, k) \in Q$ and $k' \leq p(k)$. The function f is called a *polynomial parameter transformation.*

Theorem 11 ([9]) *Let P and Q be parameterized problems and $\tilde{P}$ and $\tilde{Q}$ be the unparameterized versions of P and Q respectively. Suppose that $\tilde{P}$ is NP-hard and $\tilde{Q}$ is in NP. Assume there is a polynomial parameter transformation from P to Q. Then if Q admits a polynomial kernel, so does P.*

We apply this notion to our case.

Lemma 12 *There exists a polynomial-time algorithm that, given an instance of the* Hypergraph Minimum Bisection *problem with n vertices, outputs an equivalent instance of the* Cutwidth *problem along with its vertex cover of size n.*

Proof Let $(H = (V, E), k)$ be an instance of Hypergraph Minimum Bisection given in the input, where $|V| = n$ (n is even) and $|E| = m$. We construct a graph G as follows.

Let us denote $N = mn + 1$. We begin by taking the whole set V to be the set of vertices of G. For every distinct $u, v \in V$ we introduce N new vertices $x^i_{u,v}$ for $i = 1, 2, \ldots, N$, each connected only to u and v. Then, for every $e \in E$ we introduce a new vertex y_e connected to all $v \in e$. Denote the set of all vertices $x^i_{u,v}$ by X and the set of all vertices y_e by Y. This concludes the construction. Observe that V is a vertex cover of G of size n. We now prove that H has a bisection with cost at most k if and only if G has cutwidth at most $n^2N/4 + k$.

Assume that H has a bisection $\mathcal{B}$ with cost at most k. Let us order the vertices of the graph G as follows. First, we order the vertices from V: we place $\mathcal{B}^{-1}(0)$ first, in any order, and then $\mathcal{B}^{-1}(1)$, in any order. Then, we place every $x^i_{u,v}$ anywhere between u and v. At the end, for every $e \in E$ we place y_e at the beginning if at least half of the vertices of e are in $\mathcal{B}^{-1}(0)$, and in the end otherwise. Vertices y_e at the beginning and at the end are arranged in any order.

Now, we prove that the cutwidth of the constructed ordering is at most $n^2N/4 + k$. Consider any cut C, dividing the order on $V(G)$ into a first part V_1 and a second part V_2. Suppose that $|V_1 \cap V| = n/2 - l$ for some $-n/2 \leq l \leq n/2$, thus $|V_2 \cap V| = n/2 + l$. Observe that C cuts exactly $N(n/2 - l)(n/2 + l) = n^2N/4 - l^2N$ edges between V and X. Note that there are not more than $nm < N$ edges between V and Y. Therefore, if $l \neq 0$, then C can cut at most $n^2N/4 - N + nm < n^2N/4 + k$ edges.

We are left with the case when $l = 0$. Observe that $V_1 \cap V = \mathcal{B}^{-1}(0)$ and $V_2 \cap V = \mathcal{B}^{-1}(1)$. Moreover, the cut C cuts exactly $n^2N/4$ edges between sets V and X. As far as edges between V and Y are concerned, for every hyperedge $e \in E$, the cut C cuts exactly $cost(e, \mathcal{B})$ edges incident on y_e. As $cost(\mathcal{B}) \leq k$, the cut C cuts at most $n^2N/4 + k$ edges.

Now assume that there is an ordering of vertices of G that has cutwidth at most $n^2N/4 + k$. We construct a bisection $\mathcal{B}$ of H as follows. Let $\mathcal{B}(v) = 0$ for every v among the first $n/2$ vertices from V with respect to the ordering, and $\mathcal{B}(v) = 1$ for v among the second $n/2$ vertices. We now prove that the cost of this bisection is at most k.

Let C be any cut dividing the order into the first part V_1 and the second part V_2, such that $V_1 \cap V = \mathcal{B}^{-1}(0)$ and $V_2 \cap V = \mathcal{B}^{-1}(1)$. As the cutwidth of the ordering is at most $n^2N/4 + k$, C cuts at most $n^2N/4 + k$ edges. Observe that C needs to cut at least $n^2N/4$ edges between sets V and X, therefore it cuts at most k edges between sets V and Y. For every hyperedge $e \in E$, C cuts at least $cost(e, \mathcal{B})$ edges incident to y_e, thus $cost(\mathcal{B}) \leq k$. □

From Lemmata 9, 12 and Theorem 11 we conclude the following.

Theorem 13 Cutwidth *parameterized by the size of vertex cover does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly.*

4 Conclusions

In this paper we studied the complexity of computing the cutwidth of a graph parameterized by the size of a given vertex cover. We have shown an algorithm with running time $O(2^k n^{O(1)})$, where k is the cardinality of the vertex cover and n is the number of vertices of the graph. Moreover, we have proven that polynomial kernelization of the problem is unlikely, thus counterpoising the recent result of Bodlaender et al. [7].

The thrilling and natural question is whether the insight we have given into the problem can be a starting point to breaking the 2^n barrier for an exact algorithm computing cutwidth. Our result implies that one can assume that in any hard instance all the independent sets are small, i.e., of size not larger than cn for an arbitrarily small constant $c > 0$.

References

1. Adolphson, D., Hu, T.C.: Optimal linear ordering. SIAM Journal of Applied Mathematics **25**, 403–423 (1973)
2. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. Journal of the ACM **9**(1), 61–63 (1962)
3. Björklund, A.: Determinant sums for undirected hamiltonicity. In: FOCS, pp. 173–182. IEEE Computer Society (2010)
4. Blin, G., Fertin, G., Hermelin, D., Vialette, S.: Fixed-parameter algorithms for protein similarity search under mRNA structure constraints. Journal of Discrete Algorithms **6**, 618–626 (2008)
5. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. Journal of Computer and System Sciences **75**(8), 423–434 (2009)
6. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Cross-composition: A new technique for kernelization lower bounds. In: T. Schwentick, C. Dürr (eds.) STACS, *LIPIcs*, vol. 9, pp. 165–176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
7. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Preprocessing for treewidth: A combinatorial analysis through kernelization. In: L. Aceto, M. Henzinger, J. Sgall (eds.) ICALP (1), *Lecture Notes in Computer Science*, vol. 6755, pp. 437–448. Springer (2011)
8. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernel bounds for structural parameterizations of pathwidth. In: F.V. Fomin, P. Kaski (eds.) SWAT, *Lecture Notes in Computer Science*, vol. 7357, pp. 352–363. Springer (2012)
9. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. Theoretical Computer Science **412**(35), 4570–4578 (2011)
10. Botafogo, R.A.: Cluster analysis for hypertext systems. In: R. Korfhage, E.M. Rasmussen, P. Willett (eds.) SIGIR, pp. 116–125. ACM (1993)
11. Cai, J., Chakaravarthy, V.T., Hemaspaandra, L.A., Ogihara, M.: Competing provers yield improved Karp-Lipton collapse results. Information and Computation **198**(1), 1–23 (2005)
12. Chung, M., Makedon, F., Sudborough, I., Turner, J.: Polynomial time algorithms for the min cut problem on degree restricted trees. SIAM Journal on Computing **14**, 158–177 (1985)
13. Diaz, J., Penrose, M., Petit, J., Serna, M.: Approximating layout problems on random geometric graphs. Journal of Algorithms **39**, 78–117 (2001)
14. Drucker, A.: New limits to classical and quantum instance compression. In: FOCS (to appear). IEEE Computer Society (2012)
15. Fellows, M.R., Lokshtanov, D., Misra, N., Rosamond, F.A., Saurabh, S.: Graph layout problems parameterized by vertex cover. In: S.H. Hong, H. Nagamochi, T. Fukunaga (eds.) ISAAC, *Lecture Notes in Computer Science*, vol. 5369, pp. 294–305. Springer (2008)

16. Fomin, F.V., Kratsch, D., Todinca, I., Villanger, Y.: Exact algorithms for treewidth and minimum fill-in. SIAM Journal of Computing **38**(3), 1058–1079 (2008)
17. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. Journal of Computer and System Sciences **77**(1), 91–106 (2011)
18. Gavril, F.: Some NP-complete problems on graphs. In: 11th Conference on Information Sciences and Systems, pp. 91–95 (1977)
19. Heggernes, P., van 't Hof, P., Lokshtanov, D., Nederlof, J.: Computing the cutwidth of bipartite permutation graphs in linear time. In: D.M. Thilikos (ed.) WG, *Lecture Notes in Computer Science*, vol. 6410, pp. 75–87 (2010)
20. Heggernes, P., Lokshtanov, D., Mihai, R., Papadopoulos, C.: Cutwidth of split graphs and threshold graphs. SIAM Journal of Discrete Mathematics **25**(3), 1418–1437 (2011)
21. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics **10**(1), 196–210 (1962)
22. Jansen, B.M.P., Kratsch, S.: Data reduction for graph coloring problems. In: O. Owe, M. Steffen, J.A. Telle (eds.) FCT, *Lecture Notes in Computer Science*, vol. 6914, pp. 90–101. Springer (2011)
23. Junguer, M., Reinelt, G., Rinaldi, G.: The travelling salesman problem. Handbook on Operations Research and Management Sciences **7**, 225–330 (1995)
24. Karger, D.R.: A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. SIAM Journal of Computing **29**(2), 492–514 (1999)
25. Karp, R.M.: Dynamic programming meets the principle of inclusion and exclusion. Operations Research Letters **1**, 49–51 (1982)
26. Leighton, F., Rao, S.: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. Journal of the ACM **46**, 787–832 (1999)
27. Makedon, F., Sudborough, I.H.: On minimizing width in linear layouts. Discrete Applied Mathematics **23**, 243–265 (1989)
28. Monien, B., Sudborough, I.H.: Min cut is NP-complete for edge weighted trees. Theoretical Computer Science **58**, 209–229 (1988)
29. Mutzel, P.: A polyhedral approach to planar augmentation and related problems. In: P.G. Spirakis (ed.) ESA, *Lecture Notes in Computer Science*, vol. 979, pp. 494–507. Springer (1995)
30. Suchan, K., Villanger, Y.: Computing pathwidth faster than 2^n. In: J. Chen, F.V. Fomin (eds.) IWPEC, *Lecture Notes in Computer Science*, vol. 5917, pp. 324–335. Springer (2009)
31. Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Cutwidth II: Algorithms for partial w-trees of bounded degree. Journal of Algorithms **56**, 24–49 (2005)
32. Yannakakis, M.: A polynomial algorithm for the min cut linear arrangement of trees. Journal of the ACM **32**, 950–988 (1985)
33. Yap, C.K.: Some consequences of non-uniform conditions on uniform classes. Theoretical Computer Science **26**, 287–300 (1983)
34. Yuan, J., Zhou, S.: Optimal labelling of unit interval graphs. Applied Mathematics: A Journal of Chinese Universities Series B (English edition) **10**, 337–344 (1995)