Tight Bounds for Some Classical Problems Parameterized by Cutwidth

Narek Bojikian
Humboldt-Universität zu Berlin, Germany

Vera Chekan
Humboldt-Universität zu Berlin, Germany

Stefan Kratsch
Humboldt-Universität zu Berlin, Germany

Abstract

Cutwidth is a widely studied parameter and it quantifies how well a graph can be decomposed along small edge-cuts. It complements pathwidth, which captures decomposition by small vertex separators, and it is well-known that cutwidth upper-bounds pathwidth. The SETH-tight parameterized complexity of problems on graphs of bounded pathwidth (and treewidth) has been actively studied over the past decade while for cutwidth the complexity of many classical problems remained open.

For Hamiltonian Cycle, it is known that a $(2+\sqrt{2})^{\mathrm{pw}} n^{\mathcal{O}(1)}$ algorithm is optimal for pathwidth under SETH [Cygan et al. JACM 2018]. Van Geffen et al. [J. Graph Algorithms Appl. 2020] and Bojikian et al. [STACS 2023] asked which running time is optimal for this problem parameterized by cutwidth. We answer this question with $(1+\sqrt{2})^{\mathrm{ctw}} n^{\mathcal{O}(1)}$ by providing matching upper and lower bounds. Second, as our main technical contribution, we close the gap left by van Heck [2018] for Partition Into Triangles (and Triangle Packing) by improving both upper and lower bound and getting a tight bound of $\sqrt[3]{3}^{\mathrm{ctw}} n^{\mathcal{O}(1)}$, which to our knowledge exhibits the only known tight non-integral basis apart from Hamiltonian Cycle [Cygan et al. JACM 2018] and C_4-Hitting Set [SODA 2025]. We show that the cuts inducing a disjoint union of paths of length three (unions of so-called Z-cuts) lie at the core of the complexity of the problem – usually lower-bound constructions use simpler cuts inducing either a matching or a disjoint union of bicliques. Finally, we determine the optimal running times for Max Cut ($2^{\mathrm{ctw}} n^{\mathcal{O}(1)}$) and Induced Matching ($3^{\mathrm{ctw}} n^{\mathcal{O}(1)}$) by providing matching lower bounds for the existing algorithms – the latter result also answers an open question for treewidth by Chaudhary and Zehavi [WG 2023].

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized complexity, cutwidth, Hamiltonian cycle, triangle packing, max cut, induced matching

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.13

Related Version *Full Version*: https://arxiv.org/abs/2502.15884 [8]

1 Introduction

In parameterized complexity the (worst-case) complexity of problems is expressed in terms of input size n and one or more parameters, often denoted k. The parameter can, for example, be the size of the sought solution or some measure of the structure of the input. The goal is to understand the influence of solution size or structure on the complexity. A problem is said to be fixed-parameter tractable with parameter k if it admits an algorithm with running time of $f(k) \cdot n^{\mathcal{O}(1)}$ (also denoted by $\mathcal{O}^*(f(k))$) for some computable function f. For NP-hard problems, this function f is usually exponential and it may be doubly exponential or worse.

Due to space constraints, we defer full proofs and technical details to the full version [8].

This motivates a line of research devoted to the study of the smallest possible functions f for various problem-parameter combinations. In this context, one is often interested in NP-hard problems and hence, conjectures stronger than $\mathsf{P} \neq \mathsf{NP}$ are assumed for conditional lower bounds. For example, it has been shown that unless the Exponential Time Hypothesis (ETH) fails, some problems do not admit algorithms with running time $\mathcal{O}^*(c^k)$ for any constant c (e.g., [38]) – algorithms with such a running time are called *single-exponential*. For problems admitting single-exponential algorithms, it is natural to search for the smallest value of c for which such an algorithm exists. An even stronger conjecture called the Strong Exponential Time Hypothesis (SETH) has been assumed to prove for many problems that existing algorithms with some running time $\mathcal{O}^*(c^k)$ are essentially optimal, i.e., for any $\varepsilon > 0$, there is no algorithm for this problem running in time $\mathcal{O}^*((c-\varepsilon)^k)$. This conjecture states, informally speaking, that the SAT problem cannot be solved much more efficiently than brute-forcing all truth-value assignments.

SETH-tight complexity of problems parameterized by treewidth has been actively studied over the last decade. This was initiated by Lokshtanov et al. [37] who showed that for many classical graph problems (e.g., Independent Set or Max Cut) the folklore dynamic-programming (DP) algorithms are essentially optimal under SETH. In parallel, there is also a line of research devoted to accelerating the existing DP algorithms by employing more careful analysis and advanced tools like fast subset convolution (e.g., [4, 48, 5]), Discrete Fourier Transform (e.g., [47]), rank-based approach (e.g., [6]), isolation lemma ([40]), and Cut&Count (e.g., [17]), we refer to the survey by Nederlof [41] for more details.

Such dynamic-programming algorithms on graphs of bounded treewidth employ the fact that those graphs can be decomposed along small vertex separators and therefore, when processing this decomposition in a bottom-up way, one only needs to remember how a partial solution interacts with the current small separator, also called a bag. Thus it is also natural to study parameters based on small edge-cuts (as edge-counterparts of vertex separators). A linear arrangement of a graph places its vertices on a horizontal line so that no two vertices have the same x-coordinate. Now suppose every edge is drawn as an x-monotone curve, then the cutwidth of this linear arrangement is the maximum number of edges crossing any vertical line – observe that the edges crossing such a line separate the vertices on the left side of the vertical line from the vertices on the right side so they form an edge-cut. The cutwidth of the graph, denoted ctw, is then the minimum over all of its linear arrangements. It is well-known that pathwidth, denoted pw, can be defined similarly but instead of the edges crossing the vertical line, one counts its end-vertices on, say, the right side of the cut. In particular, this implies that the cutwidth of a graph upper-bounds its pathwidth.

Due to this relation, every $\mathcal{O}^*(f(\mathrm{pw}))$-time algorithm is also a $\mathcal{O}^*(f(\mathrm{ctw}))$-time algorithm. However, it is possible that a problem admits a more efficient algorithm when parameterized by cutwidth than when parameterized by pathwidth. For example, Edge Disjoint Paths is paraNP-hard for pathwidth [20] but becomes FPT for cutwidth [24]. This difference sparked deeper interest in studying the cutwidth parameter, and in distinguishing problems whose complexity differ between these two parameterizations. On a finer level, for some of the problems for which the SETH-tight complexity parameterized by treewidth is known, the study continued for the parameterization by cutwidth. The SETH-tight bounds parameterized by cutwidth are known for Independent Set and Dominating Set [45], Odd Cycle Transversal [7], Chromatic Number [30], #q-Coloring [26], as well as a list of connectivity problems (e.g., Feedback Vertex Set and Steiner Tree) [7]. The Chromatic Number problem exposes again a particularly interesting behavior: for treewidth, it is known that for any $q \geq 3$, the folklore $\mathcal{O}^*(q^{\mathrm{tw}})$ algorithm is optimal under SETH [37], but for

cutwidth Jansen and Nederlof [30] proved that there is a (randomized) algorithm computing the chromatic number of the graph in time $\mathcal{O}^*(2^{\mathrm{ctw}})$, i.e., independent of the number of colors in question. Recently, a notable progress was also made for H-Homomorphism: Groenland et al. [25] provided a non-algorithmic proof of the existence of so-called representative sets of certain small size on which a dynamic-programming algorithm can rely, and they also provided a lower-bound construction matching the size of these representative sets. They leave it open, though, whether representative sets of small size can also be found efficiently.

The above-mentioned paper by Lokshtanov et al. [37] provided SETH-tight lower bounds for six classical graph problems, namely Independent Set, Dominating Set, Partition Into Triangles, Odd Cycle Transversal, q-Coloring (for any fixed $q \geq 3$), and Max Cut when parameterized by treewidth [37]. The SETH-tight complexity of these problems parameterized by cutwidth was only partially known till now.

Our results. As a first contribution, we resolve the complexity of the two remaining problems from [37], namely Partition Into Triangles (and also Triangle Packing) as well as Max Cut when parameterized by cutwidth. The study of Partition into Triangles parameterized by cutwidth was initiated by van Heck [46]: it was shown that the problem admits a $\mathcal{O}^*(\sqrt[4]{8}^{\mathrm{ctw}})$ algorithm and no $\mathcal{O}^*((\sqrt{2}-\varepsilon)^{\mathrm{ctw}})$ algorithm exists for any $\varepsilon > 0$ unless SETH fails. In this work, we close this gap by providing an algorithm that solves Triangle Packing in time $\mathcal{O}^*(\sqrt[3]{3}^{\mathrm{ctw}})$, together with a matching SETH-based lower bound for the Partition Into Triangles. There is a trivial reduction from Partition Into Triangles to Triangle Packing and hence, $\mathcal{O}^*(\sqrt[3]{3}^{\mathrm{ctw}})$ is the SETH-optimal running time for both problems. Our algorithm is a straightforward dynamic programming over a path decomposition. While a simple analysis yields running time $\mathcal{O}^*(2^k)$ over a path decomposition of width k, we show that given a linear arrangement ℓ of cutwidth ctw, one can construct a specific path decomposition form ℓ, where a careful analysis shows that the number of states in the dynamic programming is bounded by $\mathcal{O}^*(\sqrt[3]{3}^{\mathrm{ctw}})$, resulting in the desired running time. A bottleneck for the running time are the so-called Z-cuts, i.e., bipartite graphs consisting of a disjoint paths of length three. We show that these cuts inherently determine the complexity of this problem, as the lower bound construction also relies on them. For Max Cut we provide a lower bound showing that no $\mathcal{O}^*((2-\varepsilon)^{\mathrm{ctw}})$ algorithm can solve this problem for any $\varepsilon > 0$ implying that the folklore algorithm for tree decompositions is optimal for cutwidth as well.

Apart from these two problems, we resolve two further open questions. We show that SETH-tight complexity of Hamiltonian Cycle is $\mathcal{O}^*((1+\sqrt{2})^{\mathrm{ctw}})$ when parameterized by cutwidth – this was asked by van Geffen et al. [45] and Bojikian et al. [7]. Let us remark, that although for pathwidth it is known that the $\mathcal{O}^*((2+\sqrt{2})^{\mathrm{pw}})$ algorithm is optimal for Hamiltonian Cycle, the complexity relative to treewidth remains a challenging open problem. Finally, Chaudhary and Zehavi [13] developed a $\mathcal{O}^*(3^{\mathrm{tw}})$ algorithm for Induced Matching problem and an SETH-based lower bound excluding $\mathcal{O}^*((\sqrt{6}-\varepsilon)^{\mathrm{ctw}})$ algorithms for any $\varepsilon > 0$. They conjectured that their algorithm is optimal and asked for a matching lower bound. We confirm their conjecture and provide a stronger result, namely that Induced Matching cannot be solved in $\mathcal{O}^*((3-\varepsilon)^{\mathrm{ctw}})$ for any $\varepsilon > 0$ when parameterized by cutwidth – this resolves the complexity of the problem for both treewidth and cutwidth. [1]

[1] Recently, the conjecture was also independently confirmed by Vasilakis and Lampis [36] by providing a matching lower bound for the parameterization by pathwidth. We remark that our lower bound for cutwidth is a stronger result.

Table 1 Tight bounds for parameterizations by treewidth / pathwidth, and cutwidth. For all problems but Hamiltonian Cycle the tight bounds are known to be equal for treewidth and pathwidth. And the tight complexity of Hamiltonian Cycle parameterized by treewidth remains open. The results of this paper are in the right column.

Triangle Packing (TP)	2^{tw}	$\sqrt[3]{3}^{\mathrm{ctw}}$
Partition into Trianlges (PT)	2^{tw}	$\sqrt[3]{3}^{\mathrm{ctw}}$
Hamiltonian Cycle (HC)	$(2+\sqrt{2})^{\mathrm{pw}}$	$(1+\sqrt{2})^{\mathrm{ctw}}$
Max Cut (MC)	2^{tw}	2^{ctw}
Induced Matching (IM)	3^{tw}	3^{ctw}

Related Work. The (S)ETH tight complexity of problems parameterized by structural parameters has been widely studied. Apart from the already mentioned results, there is a long list of papers related to such algorithms on graphs of bounded treewidth (e.g., [10, 14, 15, 18, 19, 21, 22, 32, 43, 44]), treedepth (e.g., [27, 31, 32, 42]), clique-width (e.g., [1, 2, 9, 23, 28, 31, 33]), rank-width (e.g., [3, 11]), and cutwidth (e.g., [7, 39]). There is also a line of work devoted to conjectures weaker than SETH and yielding the same lower bounds for structural parameterizations (e.g., [12, 35, 34]).

Organization. We start by providing a short summary of the used notation. Section 3 is devoted to Triangle Packing, there we provide our algorithm together with the main steps required to justify its running time. We also sketch the lower-bound construction. After that in Section 4 we present the main idea behind the lower bound state the lower bound for Hamiltonian Cycle. In Section 5 we state our lower bounds for Induced Matching and Max Cut. We conclude in Section 6 by providing some open questions.

2 Preliminaries

We use $\mathcal{O}^*$ notation to suppress factors polynomial in the input size. For an integer $i \in \mathbb{N}_0$ by $[i]$ we denote the set $\{1, \ldots, i\}$ (in particular, we have $[0] = \emptyset$) and by $[i]_0$ we denote the set $[i] \cup \{0\}$. For a vector $a = (a_1, \ldots, a_n)$ over a ground set U and a mapping $f : U \to V$ for some set V, we denote by $\big(f(v)\big)_{v \in a}$ the vector $(f(a_1), \ldots, f(a_n))$.

Given a graph $G = (V, E)$ and a vertex $v \in V$, the neighborhood of v in G is defined as $N_G(v) = \{w \in V(G) \colon \{v, w\} \in E(G)\}$. We omit the index G when clear from the context. For an edge set $F \subseteq E$, we define $G[F] = (V, F)$ and we define $V(F)$ as the set of end-points of F. For a vertex set $S \subseteq V$ and a vertex $v \in V$ we define $N_S(v) = N_G(v) \cap S$ and $\deg_S(v) = |N_S(v)|$. We define $\mathrm{cc}(G)$ as the set of all connected components of G, where a connected component of G is a maximal connected subgraph of G.

We base our lower bounds on the Strong Exponential Time Hypothesis (SETH) [29]:

▶ **Conjecture 1** (SETH). *For every $\varepsilon > 0$, there exists a constant $d > 0$ such that d-SAT cannot be solved in time $\mathcal{O}^*((2-\varepsilon)^n)$, where n is the number of variables.*

Cutwidth. A linear arrangement $\ell = v_1, \ldots, v_n$ of a graph $G = (V, E)$ is a linear ordering of V. We define $V_0 = \emptyset$, $V_i = \{v_1, \ldots, v_i\}$ and $\overline{V}_i = V \setminus V_i$ for $i \in [n]_0$. We define the *cut-graph* at $i \in [n]_0$ as the bipartite graph $H_i = G[V_i, \overline{V}_i]$. The set E_i denotes the edge set of H_i. The cutwidth of ℓ is defined as $\mathrm{ctw}(\ell) = \max_{i \in [n]} |E(H_i)|$. The cutwidth of G is defined as $\mathrm{ctw}(G) = \min_\ell \mathrm{ctw}(\ell)$, where the minimum is taken over all linear arrangements of G. We define L_i and R_i as the set of the left and right endpoints of edges in E_i, respectively. Finally, for $i \in [n]$, we define $Y_i = L_{i-1} \cup \{v_i\}$, i.e., Y_i contains all left end-points of the edges of E_{i-1} together with v_i.

Path decompositions. A *path decomposition* of a graph G is a pair $(P, \mathcal{B} : V(P) \to 2^V)$, where P is a simple path and the following properties hold:

1. For every vertex $v \in V(G)$, the set $\{x \in V(P) \colon v \in \mathcal{B}(x)\}$ induces a non-empty connected subgraph of P.
2. For every edge $\{u, v\} \in E(G)$, there exists a node $x \in V(P)$ with $\{u, v\} \subseteq \mathcal{B}(x)$.

Let $x_1, \ldots, x_r$ be the *nodes* of P in the order they occur on P. The sets $\mathcal{B}(x_1), \ldots, \mathcal{B}(x_r)$ are called *bags*. For every $x_i \in V(P)$, we define $B_{x_i} = \mathcal{B}(x_i)$, $V_{x_i} = \cup_{j \leq i} \mathcal{B}(x_i)$, and $G_{x_i} = G[V_{x_i}]$. A path decomposition $(P, \mathcal{B})$ is *nice*, if we have $\mathcal{B}(x_1) = \mathcal{B}(x_r) = \emptyset$ and for any two consecutive nodes x, x' on P, we have $|\mathcal{B}(x) \Delta \mathcal{B}(x')| \leq 1$. Hence, one can define a nice path decomposition by a sequence of introduce- and forget-vertex operations. It is sometimes useful to have designated *introduce-edge* operations as well, in this case, we call the path decomposition *very nice*. For a node x of a very nice path decomposition, by $G_x = (V_x, E_x)$ we denote the (not necessarily induced) subgraph of G whose vertex resp. edge set consists of the vertices resp. edges of G introduced in this decomposition up to the node x.

3 Triangle Packing

A *triangle packing* of a graph $G = (V, E)$ is a subgraph T of G such that each connected component of T is a cycle of length three, the *size* of T is the number of its connected components. In the Triangle Packing problem, given a graph G and a positive integer b^*, we are asked whether there exists a triangle packing of size b^* in G. In the Partition into Triangles problem, we are asked whether a triangle packing of size $|V|/3$ exists in G. We obtain the following result:

▶ **Theorem 2.** *There exists an algorithm that given an instance (G, b^*) of* Triangle Packing *together with a linear arrangement of G of width at most* ctw*, runs in time $\mathcal{O}^*(\sqrt[3]{3}^{\text{ctw}})$ and outputs whether G admits a triangle packing of size b^*.*

We emphasize that in what follows we only provide a sketch of the proof of this theorem and we refer to the full version for all details.

Let $(G = (V, E), b^*)$ be an instance of Triangle Packing and let $\ell = v_1, \ldots, v_n$ be a linear arrangement of G of cutwidth at most ctw. First, we describe a dynamic-programming algorithm over a nice path decomposition $(P, \mathcal{B})$ of G. After that, we construct a nice path decomposition of G from ℓ and show that it has certain useful properties, namely, the number of possible states of our dynamic-programming algorithm is bounded for each bag.

Algorithm over a path decomposition

Let $(P, \mathcal{B})$ be a nice path decomposition of G. For every node x of P, every set $S \subseteq B_x$, and every integer b, we define the family $\mathcal{H}_x^b[S]$ of all triangle packings of G_x of size b whose intersection with B_x is precisely S and furthermore, each triangle in this packing contains at least one vertex of $V_x \setminus B_x$ (i.e., forgotten already). We define the set $\mathcal{S}_x^b$ consisting of all subsets $S \subseteq B_x$ such that $\mathcal{H}_x^b[S]$ is non-empty. We call $\mathcal{S}_x = \bigcup_{b \in \mathbb{N}} \mathcal{S}_x^b$ the set of *realizable states* at x. For a set $Y \subseteq B_x$, let $\mathcal{S}_x[Y] = \{S \cap Y \mid S \in \mathcal{S}_x\}$ be the set of states "induced" by the set Y.

A straightforward algorithm computing all sets $\mathcal{S}_x^b$ can be informally summarized as follows. Let x' be the node preceding x. At every node x forgetting a vertex, say v, we iterate over all triangles containing the vertex v whose other end-points are still in the bag, iterate over all sets in $\mathcal{S}_{x'}^{b-1}$, and "combine" the two if they are vertex-disjoint. For every $S \in \mathcal{S}_{x'}^b$, we also add $S \setminus \{v\}$ to $\mathcal{S}_x^b$ as the corresponding triangle packing of $G_{x'}$ remains "valid" in G_x.

At every introduce-node x we just keep the family $\mathcal{S}^b_{x'}$. Recall that the root-node, say r, of the nice path decomposition $(P, \mathcal{B})$ is an empty bag. Then the graph G admits a triangle packing of size b if and only if $\emptyset \in \mathcal{S}^b_r$ holds. We can show that if α denotes the maximum size of a family $\mathcal{S}^b_x$ over all nodes x and all integers $b \in [\lfloor \frac{n}{3} \rfloor]_0$, then all families $\mathcal{S}^b_x$ can be computed in time $\mathcal{O}^*(\alpha)$. In order to achieve this, we describe a data structure that allows the insertion of a single "state" in time polynomial in n as well as the iteration over all states present in the data structure in time $\mathcal{O}^*(\alpha)$. In the remainder of the proof, we show how to get a nice path decomposition of G from its linear arrangement ℓ so that the value of α is sufficiently small.

From linear arrangement to path decomposition

To obtain the desired path decomposition $(P, \mathcal{B})$ we will proceed as follows. We will first define a set of the so-called *checkpoint* bags $X_0, \ldots, X_n$ corresponding to the cut graphs $H_0, \ldots, H_n$ of ℓ. Later we will show how to add so-called *transition* bags to turn this sequence into a nice path decomposition, while still keeping the number of possible states bounded.

First, for a bipartite graph H (think of $H = H_0, \ldots, H_n$), we define the sets F_H and I_H by an iterative process formally defined next. The set F_H is intended to represent the set of vertices from the left side of the cut that are forgotten already, while the set I_H corresponds to the vertices on the right side of the cut that are introduced already. To ensure that the corresponding "ordering" of the forget- and introduce-operations even yields a valid path decomposition, i.e., no vertex is forgotten before one of its neighbors is introduced, we will ensure that I_H contains all neighbors of F_H.

Clearly, one can safely forget a vertex if all of its neighbors have already been introduced. We additionally apply the following non-trivial rule. We forget a vertex v if (1) it has a single neighbor w in H that is not introduced yet, and (2) the vertex v has degree at most two in H. In this case we introduce the unique missing neighbor of v before forgetting v. We define Q_H as the vertices on the right side that have not been introduced yet, i.e., do not belong to I_H. We provide formal definitions of these sets (see Figure 1 for an illustration):

▶ **Definition 3.** *Let $H = (L, R, E)$ be a bipartite graph. First, we define $F^{(0)} = \emptyset$. For $j \geq 0$, let $I^{(j)} = N_H(F^{(j)})$, and let $Q^{(j)} = R \setminus I^{(j)}$. We define the sets $F^{(j+1)}$ recursively as follows:*

$$F^{(j+1)} = \left\{ v \in L \colon \deg_{Q^{(j)}}(v) = 0 \vee \left(\deg_{Q^{(j)}}(v) = 1 \wedge \deg(v) \leq 2 \right) \right\}.$$

In the full version we show, that there exists an integer k such that $F^{(k')} = F^{(k)}$ for all $k' \geq k$. Let k be the smallest such value. We define $F_H = F^{(k)}$, $I_H = I^{(k)}$, and $Q_H = Q^{(k)}$. We also define the sets L^1_H and L^2_H as the vertices of $L_H \setminus F_H$ that have exactly one or at least two neighbors in Q_H, respectively. Finally, we define the *bag* at H as $X_H = L^1_H \cup L^2_H \cup I_H$.

For every $i \in [n]_0$, we define $L^2_i = L^2_{H_i}$, $L^1_i = L^1_{H_i}$, $F_i = F_{H_i}$, $I_i = I_{H_i}, Q_i = Q_{H_i}$, and $X_i = X_{H_i}$. We call F_i the set of *forgotten vertices* at i, I_i the set of *introduced vertices*, and Q_i the set of *unintroduced vertices*.

It can be shown that X_i is a vertex separator for every $i \in [n]$: this follows from the fact that I_i is the neighborhood of F_i. Furthermore, we show in the full version that if in Definition 3 instead of $F^{(0)} = \emptyset$, we start with the set $F^{(0)} = F_{i-1}$ of the vertices already forgotten at the previous cut, then we obtain the same set F_i of vertices forgotten at the current cut. From this we can then conclude that a nice path decomposition of G can be constructed by using $X_1, \ldots, X_n$ as the main building blocks. We emphasize that the sequence of vertex separators $X_1, \ldots, X_n$ itself does not even necessarily contain every vertex of G. To resolve this, we will turn it into a nice path decomposition by adding the so-called *transition bags*. The bags $X_1, \ldots, X_n$ are called the *checkpoint bags*, and $x_1, \ldots, x_n$ denote the corresponding nodes in the arising path decomposition.

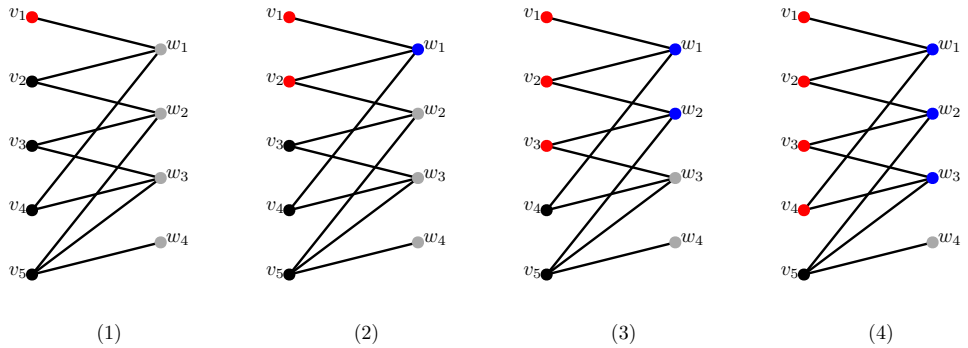

Figure 1 For $i = 1, 2, 3, 4$, the family $F^{(i)}$ is red on the left-hand side of the cut while the families $I^{(i-1)}$ and $Q^{(i-1)}$ are blue and gray, respectively, on the right-hand side. The sets F_H, I_H, Q_H are the red, blue, and gray vertices, respectively, in (4). The black vertex in (4) belongs to L_H^1 as it has a single gray neighbor. The bag X_H is the set of all black and blue vertices in (4).

The reason why we distinguish the sets L_i^1 and L_i^2 is two-fold. First, to bound the number of states in terms of cutwidth, we will "assign" the edges of the cut H_i to certain sets of vertices in the bag, these sets are then called *components*. Since every vertex in L_i^2 has at least two neighbors in Q_i, and since no vertex of Q_i belongs to the bag X_i by definition, we can assign at least two cut-edges to every vertex of L_i^2. Based on this, we show that for some components of a specific structure, enough edges of the cut can be assigned to these components to "allow" all possible state combinations of the vertices of these components. Let us elaborate a bit more on what we mean by this. Our goal is to bound the number of states of the bag X_i in terms of the number of edges in the cut H_i. To prove the desired bound, it suffices to partition the bag into the components, partition the edges of the cut to be assigned to these components, and then show that the bound holds for each component with respect to the number of assigned edges. We will show that if a component contains a vertex from L_i^2, then for the number s of vertices of this component and the number q of edges assigned to it, we have $2^s \leq \sqrt[3]{3}^q$, i.e., the desired bound holds even without a further careful analysis. We will provide more details later. Second, we also need to ensure that along the way between the checkpoint bags, i.e., in transition bags, we do not have too many possible states. This will be achieved by a careful choice of the ordering in which the vertices are forgotten and introduced. The sets L_i^1 and L_i^2 will be used to determine this ordering.

Bounding the number of realizable states

We define the graph $\hat{H}_i$ as $\hat{H}_i = H_i[V_i, I_i]$ to represent the part of the cut restricted to the vertices introduced so far (i.e., we discard the not yet introduced vertices of Q_i from H_i). For a set $S_0 \subseteq R_i$ of vertices on the right side of the cut-graph H_i, we also define the graph $H_i^{S_0} = H_i[V_i, I_i \setminus S_0]$ that additionally removes the vertices S_0 from the right side of the cut. This graph will be crucial to bound the number of the possible states for the transition bags: for example, the graph $H_{i-1}^{\{v_i\}}$ will be useful to analyze the first step of the transition from the cut H_{i-1} to the cut H_i, i.e., removing v_i from the right side of the cut.

▶ **Definition 4.** *For every $i \in [n]$ and $S \subseteq V(H_i)$, we define the set E_i^S of all edges of H_i incident with S and we define $m_i(S) = |E_i^S|$.*

By definition of the bipartite graph $\hat{H}_i$, each edge of H_i either has both end-points in some connected component of $\hat{H}_i$, or it has its left end-point in some connected component of $\hat{H}_i$ and its right end-point in Q_i. First, this implies that we have $E_i^{V(C)} = E(C) \dot{\cup} E_{H_i}(C, Q_i)$ for

every connected component C of $\hat{H}_i$. And second, it shows that every edge of H_i is incident with the vertices of exactly one connected component of $\hat{H}_i$. Therefore, the sets $E_i^{V(C)}$ for $C \in \mathrm{cc}(\hat{H}_i)$ partition $E(H_i)$. This will be crucial to bound the number of states in the checkpoint bags. An analogous argument shows that for any choice of $S_0 \subseteq R_i$, the sets $E_i^{V(C)}$ for $C \in \mathrm{cc}(H_i^{S_0})$ partition the set $E(H_i)$ as well.

Now we aim at showing that for each connected component $C \in \mathrm{cc}(\hat{H}_i)$, the number of possible states of $\mathcal{S}_{x_i}[V(C)]$ is upper-bounded by $\sqrt[3]{3}^{m_i(V(C))}$. The bound on $\mathcal{S}_{x_i}$ then follows by the fact that the sets $E_i^{V(C)}$ are pairwise disjoint. This will actually be the part where it becomes evident, as we shall see in the following proof, that the so-called Z-cuts form the bottleneck of the algorithm: We will distinguish different types of components C of $\hat{H}_i$ and the tight upper bound on the number of possible states is achieved by the Z-cuts.

We actually prove a more general statement. First, we prove that the claimed bound holds not only for $C \in \mathrm{cc}(\hat{H}_i)$, but for any connected component C of $H_i^{S_0}$ where $S_0 \subseteq R_i$ is arbitrary. Here it is important to remember that S_0 only contains vertices from the right side of the cut. Second, we will show that the bound holds even if we allow to add any subset of $L_i \setminus F_i$ to every realizable state. This motivates the next definition of the set $\mathcal{T}_i^b[S]$ which can be considered as a robust generalization of the set of possible states: intuitively, this permits us to say that even if we allow, instead of $G[V_i]$, an arbitrary graph on the left-hand side of the cut (i.e., a triangle packing is allowed to use an arbitrary subset of vertices on the left-hand side), the number of states is still bounded. This will later allow us to prove the bound for the transition bags as well, given that the transitions are carried out in the correct order. For every $i \in [n]$ and every $b \in [\lfloor \frac{n}{3} \rfloor]_0$, we will use $\mathcal{S}_i^b$ as a shorthand for $\mathcal{S}_{x_i}^b$.

▶ **Definition 5.** *For $b \in [\lfloor \frac{n}{3} \rfloor]_0$, $i \in [n]$, and $S \subseteq V(H_i)$, we define*

$$\mathcal{T}_i^b[S] = \{S' \cup T \colon S' \in \mathcal{S}_i^b[S], T \subseteq S \cap (L_i^1 \cup L_i^2)\} \text{ and } \mathcal{T}_i[S] = \cup_{b \in [\lfloor \frac{n}{3} \rfloor]_0} \mathcal{T}_i^b[S].$$

For a connected component C of a subgraph of H_i, we use $\mathcal{T}_i[C]$ and $m_i(C)$ as shorthands for $\mathcal{T}_i[V(C)]$ and $m_i(V(C))$, respectively. Now we prove the main technical lemma.

▶ **Lemma 6.** *For all $i \in [n]$, all $S_0 \subseteq R_i$, and all $C \in \mathrm{cc}(H_i^{S_0})$ it holds that $|\mathcal{T}_i[C]| \leq \sqrt[3]{3}^{m_i(C)}$.*

Sketch. Let $F = F_i \cap V(C)$, $I = I_i \cap V(C)$, $L^1 = L_i^1 \cap V(C)$, $L^2 = L_i^2 \cap V(C)$, and $X = X_i \cap V(C)$. By definition of these sets we thus get $|X| = |L^1| + |L^2| + |I|$. First of all, it can be argued that the claim is true whenever $m_i(C) \leq 2$ so we assume $m_i(C) \geq 3$ in the remainder. The proof is based on two main inequalities. The first follows directly from the definition of the $\mathcal{T}_i^b$: as every element of $\mathcal{T}_i$ is a subset of $V(C) \cap X_i = X$, we have:

$$|\mathcal{T}_i[C]| \leq 2^{|X|}. \tag{1}$$

For the second inequality, recall that $E_i^C = E(C) \dot{\cup} E_{H_i}(C, Q_i)$ holds, i.e., we have $m_i(C) = |E(C)| + |E_{H_i}(C, Q_i)|$. Since C is a connected component of $H_i^{S_0}$, its edge set $E(C)$ contains at least $|V(C)| - 1 = |L^2| + |L^1| + |F| + |I| - 1$ edges. Moreover, we claim that $|F| \geq |I|$ holds. Recall that by definition of the sets $F^{(j)}$ and $I^{(j)}$ for $j \geq 0$ (see Definition 3), we have that $|F^{(j)}| \geq |I^{(j)}|$: this is because a vertex can only be added to I_i (i.e., introduced) if at least one of its neighbors is added to F_i (i.e., forgotten). Furthermore, for any vertex v in $I^{(j)} \cap V(C)$, the unique vertex in $F^{(j)}$ due to which v was introduced belongs to the same connected component C. Thus we have $|E(C)| \geq |L^2| + |L^1| + 2|I| - 1$. Finally, recall that by definition, each vertex of L^1 has exactly one neighbor in Q_i while each vertex of L^2 has at least two neighbors in Q_i. Hence, it holds that $|E_{H_i}(C, Q_i)| \geq |L^1| + 2|L^2|$. It follows that $m_i(C) \geq (|L^2| + |L^1| + 2|I| - 1) + (|L^1| + 2|L^2|) = 2|X| + |L^2| - 1$, and hence we get:

$$|X| \leq \frac{m_i(C) + 1}{2}. \tag{2}$$

First, assume that at least one of the following hold: C contains a cycle, or L^2 is not empty, or $|F| > |I|$, or $V(C) \cap L_i$ has a neighbor in $S_0 \cap I_i$. It is not hard to verify that in this case $m_i(C) \geq 2|X|$ holds and therefore, $|\mathcal{T}_i[C]| \leq 2^{|X|} \leq \sqrt{2}^{m_i(C)} \leq \sqrt[3]{3}^{m_i(C)}$.

In the remainder of the proof we may thus assume that C is a tree, L^2 is empty, $|F| = |I|$, and each neighbor of $V(C) \cap L_i$ in S_0 belongs to Q_i. We can show that in this case, each vertex of F has degree at most 2 in H_i. After that we carry out an extensive case distinction and show that in each case, there exists a constant fraction of all subsets of X that are certainly not elements of $\mathcal{T}[C]$. The main idea behind each of the cases is to find a vertex, say v, in I with certain nice properties, namely there exists a constant-sized subset, say U, of X such that any triangle packing using v has to use at least one of the vertices in X. Thus, every subset of X that contains v and has an empty intersection with U is not a possible state – let us remark that this intuition reflects the proof in an overly simplified manner for space reasons.

The simplest of these cases is when C contains a leaf v in I. Then let w be the unique neighbor of v in L_i. As v was introduced due to forgetting one of its neighbors, this neighbor has to be w, i.e., we have $w \in F$. As $m_i(C) \geq 3$, there exists a vertex $v' \neq v \in I$ adjacent to w. Since every vertex in F has degree at most 2 in H_i, the vertex w does not have neighbors other than v and v'. Recall that the dynamic-programming algorithm only considers triangles where at least one vertex is forgotten already. Since w is the unique forgotten neighbor of v, every such triangle packing containing the vertex v, uses the triangle v, w, v'. Recall that we have $v, v' \in I \subseteq X$. Hence, for every $S \in \mathcal{T}_i[C]$, the property $v \in S$ implies $v' \in S$. This way we exclude one fourth of all subsets of X from $\mathcal{T}_i[C]$ and get:

$$|\mathcal{T}_i[C]| \leq \frac{3}{4}2^{|X|} \leq \frac{3}{4}2^{\frac{m_i(C)+1}{2}} \leq \sqrt[3]{3}^{m_i(C)},$$

where the last inequality holds due to $m_i(C) \geq 3$.

This is the spot where a Z-cut occurs: If C contains exactly three edges, then it consists of the vertices v, v', w as well as a vertex, say $w' \in F$, adjacent to v' only, and C induces a Z-cut. Recall that by definition we have $X = \{v, v'\}$ (i.e., the bag contains v and v') since the vertices w and w' are forgotten, i.e., belong to F. One can verify that in this case all states other than $\{v\}$ are possible and therefore, the above inequality is tight and the equality is achieved by a Z-cut.

If C contains no leaves in I, we can show that $m_i(C) \geq 5$ holds. Analyzing the structure of the cut, we then show the inequality $|\mathcal{T}_i[C]| \leq \frac{49}{64}2^{|X|}$ implying the desired bound. ◀

By applying the above lemma with $S_0 = \emptyset$, we can show that the number of states at each checkpoint bag is upper-bounded by $\sqrt[3]{3}^{\mathrm{ctw}}$:

▶ **Corollary 7.** *For all $i \in [n]$ and $b \in [\lfloor \frac{n}{3} \rfloor]_0$, it holds that $|\mathcal{S}_i^b| \leq \sqrt[3]{3}^{|E_i|}$.*

Due to space constraints, we omit the proof of the following bound for transition bags. It relies on a careful choice of the ordering of the forget- and introduce-operations together with an involved analysis of the relation between the connected components of the graphs $\hat{H}_{i-1}$ and $\hat{H}_i$ as well as the states of these components:

▶ **Lemma 8.** *For every transition node x_i^j and every $b \in [\lfloor \frac{n}{3} \rfloor]_0$, it holds that $|\mathcal{S}_{x_i^j}^b| \leq 4 \cdot \sqrt[3]{3}^{\mathrm{ctw}}$.*

In Figure 2 we sketch the idea behind our lower-bound construction matching the running time of our algorithm from the previous subsection and refer to the full version for a formal description. We prove the following result for Partition into Triangles:

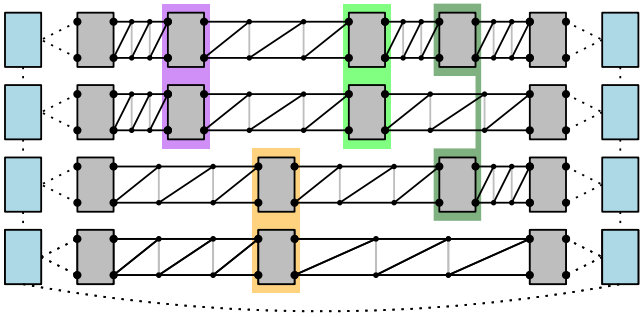

Figure 2 Sketch of the lower-bound construction for $n = 4$ variables and $m = 4$ clauses in an instance of d-CSP-3. Each of the gray boxes is a path gadget which may have one of the three states. Blue boxes are small cliques. Dotted lines reflect that there exist all possible edges between the sets. Every colored box is a constraint gadget. The cutwidth of the construction is essentially upper-bounded by $3n$ as each of the n Z-cuts contributes three edges to the size of a cut.

▶ **Theorem 9.** *Assuming SETH, there exists no algorithm that solves* Partition into Triangles *on graphs given with linear arrangements of cutwidth k in time $\mathcal{O}^*\big((\sqrt[3]{3}-\varepsilon)^k\big)$ for any $\varepsilon > 0$.*

Since there is a trivial reduction from Partition into Triangles to Triangle Packing, this lower bound then also holds for Triangle Packing and so the running time of $\mathcal{O}^*(\sqrt[3]{3}^{\mathrm{ctw}})$ is optimal for both problems under SETH.

4 Hamiltonian Cycle

For a graph $G = (V, E)$, a set $C \subseteq E$ of edges is called a *Hamiltonian cycle* of G if C induces a single cycle visiting all vertices of G. In the Hamiltonian Cycle problem, we are given a graph G and asked if there is a Hamiltonian cycle in G.

Here we provide a randomized algorithm solving Hamiltonian Cycle in time $\mathcal{O}^*((1+\sqrt{2})^k)$ on graphs provided with a linear arrangement of cutwidth k. For this we adapt the $\mathcal{O}^*((2+\sqrt{2})^p)$ algorithm by Cygan et al. [16] working on graphs provided with a path decomposition of pathwidth p. We will transform a linear arrangement of cutwidth k into a path decomposition having a useful algorithmic property, namely, the dynamic-programming table as defined by Cygan et al. has only $\mathcal{O}^*((1+\sqrt{2})^k)$ non-zero entries and there is an efficient way to determine the "certainly zero" entries. Now we provide some details. The algorithm by Cygan et al. [16] strongly relies on their algebraic result about the $\mathbb{F}_2$-rank of a certain "compatibility" matrix reflecting, for each pair of perfect matchings, whether their union is a Hamiltonian cycle. We summarize the necessary parts of this result:

▶ **Theorem 10** ([16])**.** *Let $t \in \mathbb{N}$ be even. Then there exists a set $\mathcal{X}_t$ of perfect matchings of the complete graph K_t with the following properties: (1) $|\mathcal{X}_t| = 2^{t/2-1}$, (2) $\mathcal{X}_t$ can be computed in time $\sqrt{2}^t t^{\mathcal{O}(1)}$, and (3) for every matching $M \in \mathcal{X}_t$, there exists a unique matching $M' \in \mathcal{X}_t$ such that $M \cup M'$ forms a Hamiltonian cycle of K_t.*

Observe that the third property implies that one can partition the set $\mathcal{X}_t$ into pairs of matchings such that the union of two perfect matchings forms a Hamiltonian cycle, if and only if the two matchings are paired in this partition. In other words, the family $\mathcal{X}_t$ induces a permutation submatrix of the compatibility matrix mentioned above. Like Cygan et al., we will sometimes identify some ordered set, say S, of even cardinality t with the vertex set $[t]$ of K_t. Then by $\mathcal{X}(S)$ we denote the set obtained from $\mathcal{X}_t$ by identifying the elements of S with $[t]$. In particular, every element of $\mathcal{X}(S)$ is a perfect matching on the vertex set S.

In the following, we assume that the graph G is provided with a weight function $\omega\colon E(G) \to \mathbb{N}$. As many other algorithms for connectivity problems, the algorithm of Cygan et al. [16] makes use of the classic isolation lemma (see [40]) and samples ω in a certain probabilistic way. Their algorithm works on a very nice path decomposition of the input graph G. Essentially, it processes such a path decomposition and for every bag, counts, modulo 2, the number of subgraphs of the already processed graph such that every vertex in this subgraph has degree 2, except for vertices in the current bag which may have lower degree. The dynamic-programming table refines these counts depending on the weight of the subgraph, its degree sequence on the bag, and whether this subgraph forms a single cycle with a certain matching defined on the vertex set of the bag. A crucial implication of Theorem 10 (as proven in their paper) is that instead of taking all such matchings into account, it suffices to consider only the "base matchings" from $\mathcal{X}_t$ to solve the problem. Now we summarize this more formally:

▶ **Definition 11** ([16]). *A* partial cycle cover *of a graph is a set of edges such that every vertex has at most two incident edges in this set. For every node x of the provided very nice path decomposition, every $s\colon B_x \to \{0,1,2\}$ where $s^{-1}(1)$ has even cardinality, every $M \in \mathcal{X}(s^{-1}(1))$, and every $w \in \mathbb{N}$, the value $T_x[s,M,w]$ is defined as the number, modulo 2, of partial cycle covers C of the graph G_x with the following properties:*

1. *the set $C \cup M$ of edges induces a single cycle,*
2. *the total weight (with respect to ω) of the edges in C is w,*
3. *every vertex $v \in B_x$ has precisely $s(v)$ incident edges in C,*
4. *every vertex $v \in V(G_x) \setminus B_x$ has precisely two incident edges in C.*

We say that a partial cycle cover C of G_x has *footprint* s on x if it satisfies the last two items.

▶ **Theorem 12** ([16]). *Let x be a non-first node of a very nice path decomposition of a graph G and let y denote its predecessor. For any fixed $s\colon B_x \to \{0,1,2\}$ where $s^{-1}(1)$ has even cardinality, every $M \in \mathcal{X}(s^{-1}(1))$, and every $w \in \mathbb{N}$, given the table T_y, the value $T_x[s,M,w]$ can be computed in time $\mathcal{O}^*(1)$ by querying $\mathcal{O}(1)$ entries of T_y.*

Let G be a graph and let $\ell = v_1, \ldots, v_n$ be a linear arrangement of G of cutwidth at most k. The aim now is to compute from ℓ a path decomposition of G with certain nice properties to which we will later apply the algorithm from Theorem 12. We recall that by definition for every $i \in [n]$, the set Y_i consists of all left end-points of the edges in the $(i-1)$st cut E_{i-1} of ℓ together with the vertex v_i. Thus, we have $|E_{i-1}| \leq k$ and $|Y_i| \leq k+1$ for all $i \in [n]$. It is well-known that the sequence $Y_1, \ldots, Y_n$ of bags is a path decomposition of G (see e.g., [26] for the idea and [7] for a formal proof). To reverse the ordering in which these bags are traversed, for every $i \in [n]$, we define the set $X_i = Y_{n+1-i}$.

▶ **Definition 13.** *For a node x of a path decomposition we define the sets*

$$B_1(x) = \{u \in B_x \mid \deg_{G_x}(u) = 1\} \text{ and } B_2(x) = \{u \in B_x \mid \deg_{G_x}(u) \geq 2\}.$$

We call a mapping $s\colon B_x \to \{0,1,2\}$ relevant *if all of the following hold: (1) $|s^{-1}(1)|$ is even, (2) $s^{-1}(2) \subseteq B_2(x)$, and (3) $s^{-1}(1) \subseteq B_1(x) \cup B_2(x)$.*

▶ **Lemma 14.** *Let x be a node of a very nice path decomposition of G and P be a partial cycle cover of G_x. Let $s\colon B_x \to \{0,1,2\}$ be such that P has footprint s. Then s is relevant.*

Furthermore, if $|B_1(x)| + 2 \cdot |B_2(x)| \leq k + \mathcal{O}(1)$ holds, then the number of pairs (s,M) such that $s\colon B_x \to \{0,1,2\}$ is relevant and $M \in \mathcal{X}(s^{-1}(1))$ is upper-bounded by $\mathcal{O}((1+\sqrt{2})^k)$.

Sketch. Let P be a partial cycle cover of G_x and let $s\colon B_x \to \{0,1,2\}$ be such that P has footprint s. By definition of a footprint, all end-vertices of P belong to B_x. Furthermore, every vertex of degree 1 resp. 2 in P has the degree of at least 1 resp. 2 in G_x so the first claim holds. Now let $\ell_1 = |B_1(x)|$ and $\ell_2 = |B_2(x)|$. The number of pairs (s, M) such that $s\colon B_x \to \{0,1,2\}$ is relevant and $M \in \mathcal{X}(s^{-1}(1))$ is at most

$$\sum_{0 \le i_2 \le \ell_2} \binom{\ell_2}{i_2} \sum_{0 \le i_1 \le (\ell_2 - i_2) + \ell_1} \binom{(\ell_2 - i_2) + \ell_1}{i_1} 2^{i_1/2-1}.$$

A careful analysis upper-bounds this by $\mathcal{O}((1+\sqrt{2})^k)$ if $\ell_1 + 2 \cdot \ell_2 \le k + \mathcal{O}(1)$ holds. ◀

▶ **Lemma 15.** *From the linear arrangement $v_1, \ldots, v_n$ of the graph G, in polynomial time we can construct a very nice path decomposition of G in which each node x satisfies $|B_1(x)| + 2 \cdot |B_2(x)| \le k + \mathcal{O}(1)$.*

Sketch. The desired path decomposition is obtained by starting with the nodes $x_1, \ldots, x_n$ corresponding to the bags $X_1, \ldots X_n$, respectively, and making the decomposition very nice by a careful choice of the ordering of introduce-vertex-, introduce-edge-, and forget-vertex-operations. This ordering, in particular, ensures that the graph G_{x_i} contains no edges with both end-points in $X_i \setminus \{v_{n+1-i}\}$. We recall that for every $i \in [n]$, all vertices in the bag X_i (apart from v_{n+1-i}) are left end-points of edges in the cut E_{n-i} (whose size is bounded by k). So every vertex in $B_1(x_i) \setminus \{v_{n+1-i}\}$ resp. $B_2(x_i) \setminus \{v_{n+1-i}\}$ contributes 1 resp. at least 2 to the size of E_{n-i}. And therefore $|B_1(x_i)| + 2 \cdot |B_2(x_i)|$ is upper-bounded by $k+2$. We also show that the intermediate bags added to make the decomposition very nice also have this property as they can be upper-bounded using x_i for some $i \in [n]$. ◀

We can then run the dynamic-programming algorithm from Theorem 12 restricted to relevant footprints only to compute the values $T_r[\emptyset, \emptyset, w]$ for every "reasonable" integer w where r denotes the root of a path decomposition satisfying the above lemma. Cygan et al. [16] show that this information suffices to find out, with high probability, if the graph G admits a Hamiltonian cycle. We refer to the full version for all details.

▶ **Theorem 16.** *There exists a one-sided error Monte-Carlo algorithm that takes a graph G together with a linear arrangement of G of cutwidth at most* ctw*, runs in time $\mathcal{O}^*((1+\sqrt{2})^{\mathrm{ctw}})$, and solves the* Hamiltonian Cycle *problem. The algorithm cannot give false positives and may give false negatives with probability at most $1/2$.*

Cygan et al. [16] showed that unless SETH fails, no algorithm working on path decompositions of pathwidth pw can solve the problem in time $\mathcal{O}^*((2+\sqrt{2}-\varepsilon)^{\mathrm{pw}})$ for any $\varepsilon > 0$. We show that their ideas are also useful to exclude the $\mathcal{O}^*((1+\sqrt{2}-\varepsilon)^{\mathrm{ctw}})$ algorithms for any $\varepsilon > 0$. To ensure that the construction has bounded cutwidth (and not only pathwidth), we modify the connections between path gadgets and employ new clause gadgets:

▶ **Theorem 17.** *Assuming SETH, there is no algorithm that solves* Hamiltonian Cycle *on graphs given with linear arrangements of cutwidth k in time $\mathcal{O}^*((1+\sqrt{2}-\varepsilon)^k)$ for any $\varepsilon > 0$.*

5 Max Cut and Induced Matching

In Induced Matching, given a graph G and an integer b, we are asked whether G contains a matching M of cardinality b such that $(V(M), M)$ is an induced subgraph of G, i.e., there are no edges other than M with both endpoints in $V(M)$. We prove the following result:

▶ **Theorem 18.** *Assuming SETH, there is no algorithm that solves* Induced Matching *on graphs given with linear arrangements of cutwidth k in time $\mathcal{O}^*\big((3-\varepsilon)^k\big)$ for any $\varepsilon > 0$.*

We recall that the treewidth of a graph is upper-bounded by its cutwidth. This shows that the $\mathcal{O}^*\big(3^k\big)$ algorithm by Chaudhary and Zehavi [13] working on graphs provided with tree decompositions of treewidth at most k is optimal for both treewidth and cutwidth and thus answers their open question. In Max Cut we are given a graph and asked to partition its vertex set into two sets to maximize the number of edges between these sets. We prove that the folklore $\mathcal{O}^*\big(2^k\big)$ algorithm working on graphs provided with tree decompositions of treewidth at most k is optimal for both treewidth and cutwidth:

▶ **Theorem 19.** *Assuming SETH, there is no algorithm that solves* Max Cut *on graphs given with linear arrangements of cutwidth k in time $\mathcal{O}^*\big((2-\varepsilon)^k\big)$ for any $\varepsilon > 0$.*

6 Conclusion and Future Work

Our results together with previous work [7, 30] show an interesting variety of behaviors for the complexity of problems relative to treewidth/pathwidth vs. relative to cutwidth: We may get the same tight bound, a small decrease in complexity, or a substantial decrease. We also discovered two more rare examples of tight exponential bounds with non-integral bases, especially Triangle Packing, which has an integral base relative to treewidth. In our opinion, this makes parameterization by cutwidth a very good test bed for deepening the understanding of dynamic programming and width parameters.

There are several avenues for future work: (i) Some edge-disjoint packing problems are intractable for treewidth but we may be able to dertermine tight bounds for cutwidth. (ii) Going beyond classical problems, it would be interesting to determine the tight complexity of (σ, ρ)-domination problems for cutwidth which is known for treewidth [21]. (iii) Closing the gap for H-Homomorphism left by Groenland et al. [25] is a challenging open question.

References

1 Benjamin Bergougnoux and Mamadou Moustapha Kanté. Fast exact algorithms for some connectivity problems parameterized by clique-width. *Theor. Comput. Sci.*, 782:30–53, 2019. doi:10.1016/j.tcs.2019.02.030.

2 Benjamin Bergougnoux and Mamadou Moustapha Kanté. More Applications of the d-Neighbor Equivalence: Acyclicity and Connectivity Constraints. *SIAM J. Discret. Math.*, 35(3):1881–1926, 2021. doi:10.1137/20M1350571.

3 Benjamin Bergougnoux, Tuukka Korhonen, and Jesper Nederlof. Tight Lower Bounds for Problems Parameterized by Rank-Width. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 11:1–11:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.11.

4 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.

5 Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto, Jesper Nederlof, and Pekka Parviainen. Fast Zeta Transforms for Lattices with Few Irreducibles. *ACM Trans. Algorithms*, 12(1):4:1–4:19, 2016. doi:10.1145/2629429.

6 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

7 Narek Bojikian, Vera Chekan, Falko Hegerfeld, and Stefan Kratsch. Tight Bounds for Connectivity Problems Parameterized by Cutwidth. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 14:1–14:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.14.

8 Narek Bojikian, Vera Chekan, and Stefan Kratsch. Tight Bounds for some Classical Problems Parameterized by Cutwidth. *CoRR*, abs/2502.15884, 2025. doi:10.48550/arXiv.2502.15884.

9 Narek Bojikian and Stefan Kratsch. A tight Monte-Carlo algorithm for Steiner Tree parameterized by clique-width. *CoRR*, abs/2307.14264, 2023. doi:10.48550/arXiv.2307.14264.

10 Glencora Borradaile and Hung Le. Optimal Dynamic Program for r-Domination Problems over Tree Decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.8.

11 Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. H-join decomposable graphs and algorithms with runtime single exponential in rankwidth. *Discret. Appl. Math.*, 158(7):809–819, 2010. doi:10.1016/j.dam.2009.09.009.

12 Barış Can Esmer, Jacob Focke, Dániel Marx, and Paweł Rzążewski. Fundamental Problems on Bounded-Treewidth Graphs: The Real Source of Hardness. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2024.34.

13 Juhi Chaudhary and Meirav Zehavi. P-Matchings Parameterized by Treewidth. In Daniël Paulusma and Bernard Ries, editors, *Graph-Theoretic Concepts in Computer Science – 49th International Workshop, WG 2023, Fribourg, Switzerland, June 28-30, 2023, Revised Selected Papers*, volume 14093 of *Lecture Notes in Computer Science*, pages 217–231. Springer, 2023. doi:10.1007/978-3-031-43380-1_16.

14 Radu Curticapean, Nathan Lindzey, and Jesper Nederlof. A Tight Lower Bound for Counting Hamiltonian Cycles via Matrix Rank. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1080–1099. SIAM, 2018. doi:10.1137/1.9781611975031.70.

15 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.ch113.

16 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity Checking Via Bases of Perfect Matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

17 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *CoRR*, abs/1103.0534, 2011. arXiv:1103.0534.

18 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:10.1145/3506707.

19 Baris Can Esmer, Jacob Focke, Dániel Marx, and Pawel Rzazewski. List homomorphisms by deleting edges and vertices: tight complexity bounds for bounded-treewidth graphs. *CoRR*, abs/2210.10677, 2022. doi:10.48550/arXiv.2210.10677.

20 Krzysztof Fleszar, Matthias Mnich, and Joachim Spoerhase. New algorithms for maximum disjoint paths based on tree-likeness. *Math. Program.*, 171(1-2):433–461, 2018. doi:10.1007/S10107-017-1199-3.

21 Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight Complexity Bounds for Counting Generalized Dominating Sets in Bounded-Treewidth Graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3664–3683. SIAM, 2023. doi:10.1137/1.9781611977554.ch140.

22 Jacob Focke, Dániel Marx, and Pawel Rzazewski. Counting list homomorphisms from graphs of bounded treewidth: tight complexity bounds. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 – 12, 2022*, pages 431–458. SIAM, 2022. doi:10.1137/1.9781611977073.22.

23 Robert Ganian, Thekla Hamm, Viktoriia Korchemna, Karolina Okrasa, and Kirill Simonov. The Fine-Grained Complexity of Graph Homomorphism Parameterized by Clique-Width. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 66:1–66:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.66.

24 Robert Ganian and Sebastian Ordyniak. The Power of Cut-Based Parameters for Computing Edge-Disjoint Paths. *Algorithmica*, 83(2):726–752, 2021. doi:10.1007/S00453-020-00772-W.

25 Carla Groenland, Isja Mannens, Jesper Nederlof, Marta Piecyk, and Pawel Rzazewski. Towards Tight Bounds for the Graph Homomorphism Problem Parameterized by Cutwidth via Asymptotic Matrix Parameters. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 77:1–77:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.77.

26 Carla Groenland, Isja Mannens, Jesper Nederlof, and Krisztina Szilágyi. Tight Bounds for Counting Colorings and Connected Edge Sets Parameterized by Cutwidth. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 36:1–36:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.36.

27 Falko Hegerfeld and Stefan Kratsch. Solving Connectivity Problems Parameterized by Treedepth in Single-Exponential Time and Polynomial Space. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.STACS.2020.29.

28 Falko Hegerfeld and Stefan Kratsch. Tight Algorithms for Connectivity Problems Parameterized by Clique-Width. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 59:1–59:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.59.

29 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

30 Bart M. P. Jansen and Jesper Nederlof. Computing the chromatic number using graph decompositions via matrix rank. *Theor. Comput. Sci.*, 795:520–539, 2019. doi:10.1016/j.tcs.2019.08.006.

31 Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k, r)-center. *Discret. Appl. Math.*, 264:90–117, 2019. doi:10.1016/j.dam.2018.11.002.

32 Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structurally parameterized d-scattered set. *Discret. Appl. Math.*, 308:168–186, 2022. doi:10.1016/j.dam.2020.03.052.

33 Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM J. Discret. Math.*, 34(3):1538–1558, 2020. doi:10.1137/19M1280326.

34 Michael Lampis. Circuits and Backdoors: Five Shades of the SETH, 2024. doi:10.48550/arXiv.2407.09683.

35 Michael Lampis. The Primal Pathwidth SETH. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 1494–1564. SIAM, 2025. doi:10.1137/1.9781611978322.47.

36 Michael Lampis and Manolis Vasilakis. Structural Parameterizations for Induced and Acyclic Matching. *CoRR*, abs/2502.14161, 2025. doi:10.48550/arXiv.2502.14161.

37 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

38 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly Superexponential Parameterized Problems. *SIAM Journal on Computing*, 47(3):675–702, 2018. doi:10.1137/16M1104834.

39 Dániel Marx, Govind S. Sankar, and Philipp Schepper. Degrees and Gaps: Tight Complexity Results of General Factor Problems Parameterized by Treewidth and Cutwidth. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 95:1–95:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.95.

40 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

41 Jesper Nederlof. Algorithms for NP-Hard Problems via Rank-Related Parameters of Matrices. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms – Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 145–164. Springer, 2020. doi:10.1007/978-3-030-42071-0_11.

42 Jesper Nederlof, Michał Pilipczuk, Céline M. F. Swennenhuis, and Karol Węgrzycki. Hamiltonian Cycle Parameterized by Treedepth in Single Exponential Time and Polynomial Space. In *Proc. WG 2020*, volume 12301 of *Lecture Notes Comput. Sci.*, pages 27–39, 2020. doi:10.1007/978-3-030-60440-0_3.

43 Karolina Okrasa, Marta Piecyk, and Pawel Rzazewski. Full Complexity Classification of the List Homomorphism Problem for Bounded-Treewidth Graphs. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 74:1–74:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.74.

44 Karolina Okrasa and Pawel Rzazewski. Fine-Grained Complexity of the Graph Homomorphism Problem for Bounded-Treewidth Graphs. *SIAM J. Comput.*, 50(2):487–508, 2021. doi:10.1137/20M1320146.

45 Bas A. M. van Geffen, Bart M. P. Jansen, Arnoud A. W. M. de Kroon, and Rolf Morel. Lower Bounds for Dynamic Programming on Planar Graphs of Bounded Cutwidth. *J. Graph Algorithms Appl.*, 24(3):461–482, 2020. doi:10.7155/jgaa.00542.

46 Ivo van Heck. Triangle partition on graphs of bounded cutwidth upper- and lower bounds on algorithmic and communication complexity. Master's thesis, Eindhoven University of Technology, Eindhoven, June 2018. Available at `https://pure.tue.nl/ws/portalfiles/portal/109480417/Thesis0775551_Ivo_van_Heck.pdf`.

47 Johan M. M. van Rooij. Fast Algorithms for Join Operations on Tree Decompositions. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms – Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 262–297. Springer, 2020. `doi:10.1007/978-3-030-42071-0_18`.

48 Johan M. M. van Rooij. A Generic Convolution Algorithm for Join Operations on Tree Decompositions. In Rahul Santhanam and Daniil Musatov, editors, *Computer Science – Theory and Applications – 16th International Computer Science Symposium in Russia, CSR 2021, Sochi, Russia, June 28 – July 2, 2021, Proceedings*, volume 12730 of *Lecture Notes in Computer Science*, pages 435–459. Springer, 2021. `doi:10.1007/978-3-030-79416-3_27`.