Counting Connected Subgraphs with Maximum-Degree-Aware Sieving

Andreas Björklund
Department of Computer Science, Lund University, Sweden

Thore Husfeldt
BARC, IT University of Copenhagen, Denmark and Lund University, Sweden

Petteri Kaski
Department of Computer Science, Aalto University, Finland

Mikko Koivisto
Department of Computer Science, University of Helsinki, Finland

Abstract

We study the problem of counting the isomorphic occurrences of a k-vertex *pattern* graph P as a subgraph in an n-vertex *host* graph G. Our specific interest is on algorithms for subgraph counting that are sensitive to the maximum degree Δ of the host graph.

Assuming that the pattern graph P is connected and admits a vertex *balancer* of size b, we present an algorithm that counts the occurrences of P in G in $O\big((2\Delta-2)^{\frac{k+b}{2}}2^{-b}\frac{n}{\Delta}k^2\log n\big)$ time. We define a balancer as a vertex separator of P that can be represented as an intersection of two equal-size vertex subsets, the union of which is the vertex set of P, and both of which induce connected subgraphs of P.

A corollary of our main result is that we can count the number of k-vertex paths in an n-vertex graph in $O\big((2\Delta-2)^{\lfloor\frac{k}{2}\rfloor}nk^2\log n\big)$ time, which for all moderately dense graphs with $\Delta\leq n^{1/3}$ improves on the recent breakthrough work of Curticapean, Dell, and Marx [STOC 2017], who show how to count the isomorphic occurrences of a q-edge pattern graph as a subgraph in an n-vertex host graph in time $O(q^q n^{0.17q})$ for all large enough q. Another recent result of Brand, Dell, and Husfeldt [STOC 2018] shows that k-vertex paths in a bounded-degree graph can be approximately counted in $O(4^k n)$ time. Our result shows that the *exact* count can be recovered at least as fast for $\Delta<10$.

Our algorithm is based on the principle of inclusion and exclusion, and can be viewed as a sparsity-sensitive version of the "counting in halves"-approach explored by Björklund, Husfeldt, Kaski, and Koivisto [ESA 2009].

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases graph embedding, k-path, subgraph counting, maximum degree

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.17

Funding This work was supported in part by the Swedish Research Council grant VR-2016-03855, "Algebraic Graph Algorithms". BARC, Basic Algorithms Research Copenhagen, is funded by the Villum Foundation grant 16582. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement 338077 "Theory and Practice of Advanced Search and Enumeration".

1 Introduction

Subgraph statistics are among the most fundamental and extensively studied invariants of graphs. A canonical task in this domain is to count the number of isomorphic occurrences of a connected k-vertex *pattern* graph as a subgraph in an n-vertex *host* graph.

Assuming k is a constant, we can explicitly list all the occurrences of the pattern in the host graph in time $O(n^k)$. A substantial literature exists on counting algorithms that improve on the $O(n^k)$ bound. Currently the fastest algorithm design for the general case of an unconstrained k-vertex pattern remains the $O(n^{\omega k/3})$-time algorithm of Nešetřil and Poljak [27] (see also Eisenbrand and Grandoni [14]), where $2 \leq \omega < 2.3728639$ is the exponent of $n \times n$ matrix multiplication (cf. Le Gall [24] and Vassilevska Williams [30]). By parameterizing on the structure of the pattern graph, many further and faster algorithm designs become possible; we postpone a detailed discussion of earlier work after a statement of our present focus and main result.

Sensitivity to the maximum degree. In this paper, we are interested in subgraph-counting algorithms that are sensitive to the *maximum degree* Δ in addition to the number of vertices n in the host graph.[1] Our interest is in particular on algorithm designs that scale to massive graphs where Δ can be orders of magnitude smaller than n. Such study of algorithms that are sensitive to Δ can be found, for example, in the work of Komusiewicz and Soren [23] in the context of optimization over all k-vertex subgraphs.

In our case of connected subgraph counting, it is immediate that the host can contain at most $n(\Delta-1)^{k-1}$ subgraphs that are isomorphic to the pattern, and furthermore these subgraphs can be trivially listed in $O(n(\Delta-1)^{k-1})$ time.

Our goal in this paper is to improve the trivial running time of connected subgraph counting to the general form

$$O\big(n(\alpha\Delta)^{\beta k}\big) \tag{1}$$

for constants $\alpha \geq 1$ and $0 \leq \beta \leq 1$ that depend on the topology of the pattern but not on the parameters k, n, and Δ. In particular, the main conceptual contribution of this paper is to establish that nontrivial Δ-sensitive exponents $\beta < 1$ can be achieved for elementary connected topologies, such k-vertex paths and cycles, for which we establish $\beta = 1/2$ and $\alpha = 2$ independently of k (cf. Corollary 3 for a precise statement). Furthermore, our algorithms scale *linearly* in the number of vertices n, thus enabling a more fine-grained control of subgraph counting by isolating the complexity to the maximum degree Δ and the topology of the connected pattern.

Our results. Let us now proceed with a detailed statement of our results. (The standard graph-theoretic terminology and preliminaries can be found in Section 2.) We are interested in connected pattern graphs that admit a small *balancer* in the following sense.

▶ **Definition 1** (Balancer). A vertex subset $B \subseteq V(P)$ of a connected graph P is a *balancer* if there exist subsets $C_1, C_2 \subseteq V(P)$ such that

1. $|C_1| = |C_2|$,
2. $C_1 \cup C_2 = V(P)$,
3. the induced subgraphs $P[C_1]$ and $P[C_2]$ are connected, and
4. $B = C_1 \cap C_2$ is a $(C_1 \setminus C_2, C_2 \setminus C_1)$-separator in P.

[1] To avoid degenerate cases, let us assume $\Delta \geq 2$ in what follows.

For example, a k-vertex path has a balancer of size $2 - (k \bmod 2)$, a k-vertex cycle has a balancer of size $2 + (k \bmod 2)$, and a k-vertex tree for $k \geq 3$ has a balancer of size at most $\lceil k/3 \rceil$ (cf. Lemma 7). Trivially, every k-vertex connected graph has a balancer of size at most k. It is also immediate that a balancer and k must have the same parity.

▶ **Theorem 2** (Main; Counting connected subgraphs with a small balancer). *Let P be a connected k-vertex graph with a balancer of size b, and let G be an n-vertex graph with maximum degree $\Delta \geq 2$. There exists an algorithm that counts the number of isomorphic occurrences of P as a subgraph in G in time*

$$O\left((2\Delta - 2)^{\frac{k+b}{2}} 2^{-b} \frac{n}{\Delta} k^2 \log n\right). \tag{2}$$

Let us illustrate the use of Theorem 2 by stating a corollary for elementary connected patterns such as paths, cycles, and trees with arbitrary topology.

▶ **Corollary 3.** *There exist algorithms that output, given as input an n-vertex host graph G of maximum degree $\Delta \geq 2$,*

1. *the number of k-vertex paths in G in time $O\big((2\Delta - 2)^{\lfloor \frac{k}{2} \rfloor} nk^2 \log n\big)$,*
2. *the number of k-vertex cycles in G in time $O\big((2\Delta - 2)^{\lceil \frac{k}{2} \rceil} nk^2 \log n\big)$, and*
3. *the number of isomorphic occurrences of any fixed k-vertex tree T for $k \geq 3$ in G in time $O\big((2\Delta - 2)^{\lceil \frac{2k-3}{3} \rceil} nk^2 \log n\big)$.*

Discussion and related work. Recently, Patel and Regts [29] have shown that the number of subgraphs of G that *induce* an isomorphic occurrence of a given k-vertex pattern graph can be computed in time $\Delta^{O(k)} n$; their precise bound is $O\big((n(4\Delta)^{2k} + 2^{10k})\mathrm{poly}(k)\big)$. This result is sensitive to the sparsity of the host graph even when the pattern graph is disconnected.

Just as recently, Curticapean, Dell, and Marx [12] showed that the isomorphic occurrences of a q-edge pattern graph in an n-vertex host graph can be counted for all large enough q in $O(q^q n^{0.17q})$ time, building on the connection between the number of subgraphs and the number of homomorphisms established by Lovász 50 years ago [25]. As further motivation for our present study, we observe that the Curticapean–Dell–Marx algorithm cannot in a direct way utilise sparsity of the host graph, even when the pattern graph is connected. Indeed, the Curticapean–Dell–Marx algorithm is based on homomorphism-counting over low-width tree-decompositions of consolidations of the pattern graph, and there is no guarantee that the bags of such a tree-decomposition induce connected subgraphs, which means their algorithm has to track essentially arbitrary mappings of vertices to the host graph. In contrast, our present algorithm tracks embeddings of connected subgraphs of the pattern graph, which enables us to control the number of such embeddings with Δ. For k-vertex paths, or any connected pattern graph with a balancer much smaller than k, we obtain a faster subgraph counting algorithm for every $\Delta \leq n^{1/3}$.

In another very recent work, Brand, Dell, and Husfeldt [8] show that one can approximately count the number of k-vertex paths in a bounded-degree n-vertex host graph in $O(4^k n)$ time with a randomized approximation scheme. That is, for every $\epsilon > 0$ they present a Monte-Carlo algorithm that computes, with probability at least $\frac{99}{100}$, a factor-$(1+\epsilon)$ approximation of the exact count, with running time inversely proportional to ϵ^2. Our algorithm is deterministic and recovers the *exact* count in the same time, or faster, for all graphs with $\Delta < 10$.

A third improvement concerns deterministic k-path detection. Zehavi [33] shows that there is a deterministic algorithm for k-path detection in general graphs running in $O(2.6^k \,\mathrm{poly}(n))$ time. No better algorithm is known for $\Delta = 4$. For $\Delta = 3$, one can enumerate the non-backtracking walks in $O(2^k n)$ time. For $\Delta = 4$, we directly obtain a $O(2.44^k n)$ time algorithm as a special case of Corollary 3.

Methodology. Our algorithmic insight here is an old one: that one can use a meet-in-the-middle approach dividing the pattern graph (or more precisely in our present case, an embedding of the pattern graph) into two equal halves. To count half-pairs that together define an embedding of the pattern into the host, we can use an inclusion–exclusion sieve that cancels every pair with overlaps outside a controlled root. Björklund *et al.* [6] showed one can count the occurrences of a subgraph in $n^{k/2}$ time using fast zeta transforms and an inclusion–exclusion sieve on the subset lattice. However, as far as we can tell, one cannot exploit sparsity in this sieve directly. Our new algorithm here is based on observing that many of the computation points on the sieve will be zero. Rather than applying a fast zeta transform, we are better off by explicitly computing the points where the result is non-zero.

Further earlier work and complexity results. Subgraph counting has received a substantial amount of attention in the algorithms community. A non-exhaustive sample of earlier work includes Itai and Rodeh [19], Nešetřil and Poljak [27], Alon, Yuster, Zwick [3], Alon and Gutner [2], Eisenbrand and Grandoni [14], Björklund, Husfeldt, Kaski, and Koivisto [6], Björklund, Kaski, and Kowalik [7], Vassilevska Williams, Wang, Williams, and Yu [31], Vassilevska Williams and Williams [32], Amini, Fomin, and Saurabh [4], Fomin, Lokshtanov, Raman, Saurabh, and Raghavenda Rao [17], Floderus, Kowaluk, Lingas, and Lundell [15], Olariu [28], Kloks, Kratsch, Müller [22], Curticapean, Dell, and Marx [12], Brand, Dell, and Husfeldt [8], and Austrin, Kaski, and Kubjas [5].

From a parameterized complexity perspective, subgraph counting parameterized by the number of vertices k in the pattern graph P is a hard problem in the class #W[1] when P has unbounded vertex cover number. Cf. Flum and Grohe [16], Chen and Flum [9], Chen, Thurley, Weyer [10], Curticapean [11], Curticapean and Marx [13], Jerrum and Meeks [20,21], and Meeks [26]. The specific problem of finding and counting cliques is used as a source of fine-grained hardness reductions by Abboud, Backurs, and Vassilevska Williams [1].

Under the Exponential Time Hypothesis (ETH), Impagliazzo *et al.* [18] have shown that there can be no algorithm for detecting a Hamiltonian path in time $\exp o(n)$. By inspecting the textbook reduction from 3-Satisfiability to Hamiltonicity used in that argument, we observe that this result holds even if the input graph has constant degree. Thus, the constant β in (1) cannot be arbitrarily reduced, even if the dependency on Δ is much relaxed. In particular, the hypothesis forbids an algorithm for counting (or even detecting) k-paths with running time $(f(\Delta))^{o(k)}\mathrm{poly}(n)$ for any computable function f.

Organization. The rest of this paper is organized as follows. Section 2 reviews the standard definitions and notational conventions used in this paper. Section 3 presents our main sieving lemma for counting embeddings from two parts. Section 4 gives an algorithm for listing the embeddings of a connected pattern graph to a host graph. Section 5 develops our sieving algorithm for counting embeddings from two parts. Section 6 completes our main algorithm design and the proof of Theorem 2. Section 7 proves Corollary 3 and studies balancers in elementary families of connected graphs.

2 Preliminaries

This section reviews the standard definitions and notational conventions used in this paper.

Graphs and subgraphs. Unless mentioned otherwise, all graphs in this paper are undirected, loopless, and without parallel edges. For a graph G, we write $V = V(G)$ for the vertex set and $E = E(G)$ for the edge set of G, where each edge $e \in E(G)$ is a 2-element subset of

$V(G)$. Let us write $\Delta = \Delta(G)$ for the maximum degree of a vertex in G. A graph H is a *subgraph* of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We write $H \subseteq G$ to indicate that H is a subgraph of G. For a set $S \subseteq V(G)$, the subgraph $G[S]$ of G *induced* by S is defined by $V(G[S]) = S$ and $E(G[S]) = \{\{u, v\} \in E(G) : u, v \in S\}$. We tacitly assume in what follows that all algorithms accept their input graphs in adjacency list form.

Separators. Let G be a graph and let $A, B \subseteq V(G)$. We say that a set $S \subseteq V(G)$ is an (A, B)-*separator* in G if for all $a \in A$ and all $b \in B$ it holds that every path in G joining a and b contains at least one vertex in S.

Mappings. For a mapping $\varphi : X \to Y$ and a subset $S \subseteq X$, we write $\varphi|_S : S \to Y$ for the *restriction* of φ to S and $\varphi(S) = \{\varphi(x) : x \in S\} \subseteq Y$ for the *image* of S under φ. For two mappings $\varphi : X \to Y$ and $\psi : Y \to Z$, let us write $\psi \circ \varphi : X \to Z$ for their *composition* defined for all $x \in X$ by $\psi \circ \varphi(x) = \psi(\varphi(x))$.

Homomorphism, embedding, isomorphism, automorphism. Let P and G be graphs. A mapping $\varphi : V(P) \to V(G)$ is a *homomorphism* from P to G if for all $\{u, v\} \in E(P)$ it holds that $\{\varphi(u), \varphi(v)\} \in E(G)$. An injective homomorphism is called an *embedding* (or a *monomorphism*) of P into G. A bijective homomorphism whose inverse is also a homomorphism is an *isomorphism*. An isomorphism from a graph P to itself is an *automorphism* of P.

Let us write $\mathrm{Hom}(P, G)$, $\mathrm{Emb}(P, G)$, $\mathrm{Iso}(P, G)$ for the set of all homomorphisms, embeddings, and isomorphisms, respectively, from P to G. Similarly, let us write $\mathrm{Aut}(P)$ for the set of all automorphisms of P.

Subgraph occurrences and subgraph counting. Let P and G be graphs. Let us write $\mathrm{Sub}(P, G)$ for the set of all subgraphs $H \subseteq G$ such that P and H are isomorphic. We call each element of $\mathrm{Sub}(P, G)$ an *occurrence* of P in G. The number of embeddings of P to G and the number of occurrences of P in G are related by the identity

$$|\mathrm{Emb}(P, G)| = |\mathrm{Aut}(P)| \cdot |\mathrm{Sub}(P, G)| \,. \tag{3}$$

In particular, assuming $|\mathrm{Aut}(P)|$ is known, knowledge of one of $|\mathrm{Emb}(P, G)|$ or $|\mathrm{Sub}(P, G)|$ enables one to solve for the other via (3).

Iverson bracket notation. For a logical proposition P, it will be convenient to use Iverson's bracket notation

$$[\![P]\!] = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{if } P \text{ is false.} \end{cases}$$

Model of computation. We work in a word-RAM model where basic word operations on $O(\log n)$-bit words take time $O(1)$, where $n = |V(G)|$ is the number of vertices in the input host graph G.

3 A sieving lemma for the number of embeddings

This section starts our work towards the proof of Theorem 2. The goal of this section is a technical sieving lemma that enables us to count embeddings φ "in halves" (in analogy with Björklund *et al.* [6]) by sieving pairs (φ_1, φ_2) of partial embeddings for those pairs that both agree with a root map ρ and are otherwise disjoint in terms of their image sets.

In more precise terms, let P and G be graphs and let $C_1, C_2 \subseteq V(P)$ such that
1. $C_1 \cup C_2 = V(P)$ and
2. $C_1 \cap C_2$ is a $(C_1 \setminus C_2, C_2 \setminus C_1)$-separator in P.

Let us fix a *root map* $\rho : C_1 \cap C_2 \to V(G)$. We say that an embedding $\varphi \in \mathrm{Emb}(P, G)$ is *ρ-rooted* if $\varphi|_{C_1 \cap C_2} = \rho$. Let us write $\mathrm{Emb}_\rho(P, G)$ for the set of all ρ-rooted embeddings in $\mathrm{Emb}(P, G)$.

The following sets will form the core of the sieve. For $X \subseteq V(G)$ and $S \subseteq V(P)$ with $C_1 \cap C_2 \subseteq S$, let us define

$$I_{\rho,S}(X) = \{\varphi \in \mathrm{Emb}_\rho(P[S], G) : X \subseteq \varphi(S)\}\,. \tag{4}$$

We are now ready for our main sieving lemma.

▶ **Lemma 4** (Sieving ρ-rooted embeddings from two parts)**.** *We have*

$$|\mathrm{Emb}_\rho(P, G)| = \sum_{X \subseteq V(G) \setminus \rho(C_1 \cap C_2)} (-1)^{|X|} \cdot |I_{\rho,C_1}(X)| \cdot |I_{\rho,C_2}(X)|\,. \tag{5}$$

Proof. Recalling that every nonempty finite set has equally many even-sized and odd-sized subsets, for any (possibly empty) finite set Y we have

$$\sum_{X \subseteq Y} (-1)^{|X|} = [\![Y = \emptyset]\!]\,. \tag{6}$$

Let us use the notational shorthands $V_\rho = V(G) \setminus \rho(C_1 \cap C_2)$, $E_1 = \mathrm{Emb}_\rho(P[C_1], G)$, and $E_2 = \mathrm{Emb}_\rho(P[C_2], G)$. Expanding the right-hand side of (5), we obtain

$$\begin{aligned}
&\sum_{X \subseteq V_\rho} (-1)^{|X|} \cdot |I_{\rho,C_1}(X)| \cdot |I_{\rho,C_2}(X)| \\
&\quad= \sum_{X \subseteq V_\rho} (-1)^{|X|} \sum_{\varphi_1 \in E_1} [\![X \subseteq \varphi_1(C_1)]\!] \sum_{\varphi_2 \in E_2} [\![X \subseteq \varphi_2(C_2)]\!] \\
&\quad= \sum_{\varphi_1 \in E_1} \sum_{\varphi_2 \in E_2} \sum_{X \subseteq V_\rho} (-1)^{|X|} [\![X \subseteq \varphi_1(C_1)]\!] [\![X \subseteq \varphi_2(C_2)]\!] \\
&\quad= \sum_{\varphi_1 \in E_1} \sum_{\varphi_2 \in E_2} \sum_{X \subseteq V_\rho} (-1)^{|X|} [\![X \subseteq \varphi_1(C_1) \cap \varphi_2(C_2)]\!] \\
&\quad= \sum_{\varphi_1 \in E_1} \sum_{\varphi_2 \in E_2} \sum_{X \subseteq \varphi_1(C_1 \setminus C_2) \cap \varphi_2(C_2 \setminus C_1)} (-1)^{|X|} \\
&\quad= \sum_{\varphi_1 \in E_1} \sum_{\varphi_2 \in E_2} [\![\varphi_1(C_1 \setminus C_2) \cap \varphi_2(C_2 \setminus C_1) = \emptyset]\!]\,.
\end{aligned}$$

To establish the lemma, it now suffices to show that the last double sum equals $|\mathrm{Emb}_\rho(P, G)|$. Toward this end, let us observe that a pair $(\varphi_1, \varphi_2) \in E_1 \times E_2$ of embeddings defines a unique embedding $\varphi \in \mathrm{Emb}_\rho(P, G)$ if and only if we have

$$\varphi_1(C_1 \setminus C_2) \cap \varphi_2(C_2 \setminus C_1) = \emptyset\,. \tag{7}$$

In the "if" direction, each pair $(\varphi_1, \varphi_2) \in E_1 \times E_2$ defines a map $\varphi : V(P) \to V(G)$ via the restrictions $\varphi|_{C_1} = \varphi_1$ and $\varphi|_{C_2} = \varphi_2$. Indeed, we observe that φ is a well-defined injective map because we have $\varphi_1|_{C_1 \cap C_2} = \varphi_2|_{C_1 \cap C_2} = \rho$ together with $C_1 \cup C_2 = V(P)$ and (7).

Furthermore, φ is a homomorphism from P to G because φ_1, φ_2 are homomorphisms and because $C_1 \cap C_2$ is a $(C_1 \setminus C_2, C_2 \setminus C_1)$-separator in P; that is, φ maps every edge of P to an edge of G since every edge of P has both of its end-vertices in C_1 or in C_2.

In the "only if" direction, each embedding $\varphi \in \mathrm{Emb}_\rho(P, G)$ restricts to $\varphi_1 = \varphi|_{C_1}$ and $\varphi_2 = \varphi|_{C_2}$. It is immediate that we have $\varphi_1 \in E_1$, $\varphi_2 \in E_2$, and (7) holds. This completes the lemma. ◀

▶ Remark. From (4) it is immediate that we have $I_{\rho,C_j}(X) = \emptyset$ unless $|X| \leq |C_j|$, so it suffices to restrict the sieve (5) to sets X with $|X| \leq \min(|C_1|, |C_2|)$.

4 Listing the embeddings of a connected pattern graph

To turn Lemma 4 into an algorithm that is sensitive to the maximum degree $\Delta = \Delta(G)$ of the host graph G, we will rely on a subroutine that we use to explicitly list the embeddings in $\mathrm{Emb}(P[C_1], G)$ and in $\mathrm{Emb}(P[C_2], G)$. This Δ-sensitive listing subroutine is the content of this section and the following lemma.

▶ **Lemma 5** (Listing embeddings of a connected pattern graph)**.** *Let Q be a connected graph with $q = |V(Q)|$. Let G a graph with $n = |V(G)|$ and $\Delta = \Delta(G) \geq 2$. There exists an algorithm that lists all the embeddings in* $\mathrm{Emb}(Q, G)$ *in time*

$$O\big(n(\Delta - 1)^{q-1} q^2 \log n\big)\,. \tag{8}$$

Proof. Since Q is connected, it has a spanning tree. Fix an arbitrary spanning tree T of Q and fix an arbitrary vertex $r \in V(T)$ as the root of T. Use a recursive procedure to construct all embeddings $\varphi : V(T) \to V(G)$ one image $\varphi(x) \in V(G)$ at a time for each $x \in V(T)$, starting from the root r, and proceeding so that whenever the image of $x \neq r$ is being fixed, the parent $p \in V(T)$ of x in T (towards the root r) has its image $\varphi(p)$ already fixed. Whenever an embedding $\varphi : V(T) \to V(G)$ is completed, we test whether φ is an embedding of Q to G and output φ if this is the case.

To analyze the running time, we observe that there are at most n choices for the image $\varphi(r) \in V(G)$ of the root r. Since the next image needs to be adjacent to $\varphi(r)$, there are at most Δ choices for the next image (if any). For all subsequent $q - 2$ images (if any), we have that there are at most $\Delta - 1$ choices for $\varphi(x)$ since $\varphi(x)$ and $\varphi(p)$ are adjacent and $\varphi(x)$ needs to be distinct from all the previously fixed images. Thus, in total there are at most $n\Delta(\Delta-1)^{q-2} = O(n(\Delta-1)^{q-1})$ embeddings $\varphi \in \mathrm{Emb}(T, G)$. The (unoptimized) component $q^2 \log n$ in the running time bound (8) comes from testing that the $|E(Q)| \leq q^2$ adjacencies $\{\varphi(z), \varphi(w)\} \in E(G)$ hold for each $\{z, w\} \in E(Q)$ by binary search to the adjacency lists of G given by φ. ◀

▶ Remark. We observe that the listing algorithm in Lemma 5 would not work without the assumption that Q is connected.

5 A sieving algorithm for the number of embeddings

This section continues our work towards Theorem 2 by combining Lemma 4 and Lemma 5 to a sieving algorithm for the number $|\mathrm{Emb}(P, G)|$ of embeddings of a connected k-vertex pattern graph P to an n-vertex host graph G.

The sieving algorithm will rely on a balancer for P. In more precise terms, let $C_1, C_2 \subseteq V(P)$ so that $B = C_1 \cap C_2$ is a balancer of size $b = |B|$. Recalling Definition 1, we have

1. $|C_1| = |C_2|$,
2. $C_1 \cup C_2 = V(P)$,
3. the induced subgraphs $P[C_1]$ and $P[C_2]$ are connected, and
4. $C_1 \cap C_2$ is a $(C_1 \setminus C_2, C_2 \setminus C_1)$-separator in P.

Furthermore, since $k = |V(P)|$ and $b = |C_1 \cap C_2|$, we thus have $|C_1| = |C_2| = \frac{k+b}{2}$.

▶ **Lemma 6** (Sieving algorithm for the number of embeddings). *Let P be a connected k-vertex graph with a balancer of size b. Let G be a graph with $n = |V(G)|$ and $\Delta = \Delta(G) \geq 2$. There exists an algorithm that computes $|\mathrm{Emb}(P, G)|$ in time*

$$O\left((2\Delta - 2)^{\frac{k+b}{2}} 2^{-b} \frac{n}{\Delta} k^2 \log n\right). \tag{9}$$

Proof. Let C_1, C_2 be the sets in that define the balancer of size b. Recall the sets (4) that form the core of the sieve in Lemma 4. The algorithm works with a dictionary data structure that records and builds the *nonempty* sets $I_{\rho,C_j}(X)$ indexed by three-tuples (ρ, C_j, X) with $\rho : C_1 \cap C_2 \to V(G)$, $j = 1, 2$, and $X \subseteq V(G) \setminus \rho(C_1 \cap C_2)$. We build the nonempty sets $I_{\rho,C_j}(X)$ using the listing algorithm in Lemma 5.

First, we use the algorithm in Lemma 5 with $Q = P[C_1]$ and $|V(Q)| = \frac{k+b}{2}$ to list all the embeddings $\varphi_1 \in \mathrm{Emb}(P[C_1], G)$. By the analysis in Lemma 5, there are at most $n(\Delta - 1)^{\frac{k+b}{2}-1}$ such embeddings. For each listed φ_1, we insert φ_1 into the set $I_{\rho,C_1}(X)$ for $\rho = \varphi_1|_{C_1 \cap C_2}$ and for each $X \subseteq \varphi_1(C_1 \setminus C_2)$. Since $|\varphi_1(C_1 \setminus C_2)| = |C_1 \setminus C_2| = \frac{k-b}{2}$, the number of nonempty sets $I_{\rho,C_1}(X)$ will be at most

$$n(\Delta - 1)^{\frac{k+b}{2}-1} 2^{\frac{k-b}{2}} = O\left(\frac{n}{\Delta}(2\Delta - 2)^{\frac{k+b}{2}} 2^{-b}\right). \tag{10}$$

Second, we use the algorithm in Lemma 5 with $Q = P[C_2]$ and $|V(Q)| = \frac{k+b}{2}$ to list all the embeddings $\varphi_2 \in \mathrm{Emb}(P[C_2], G)$. For each listed φ_2, we insert φ_2 into the set $I_{\rho,C_2}(X)$ for $\rho = \varphi_2|_{C_1 \cap C_2}$ and for each $X \subseteq \varphi_2(C_2 \setminus C_1)$. The number of nonempty sets $I_{\rho,C_2}(X)$ will similarly be at most (10).

Third, let us observe that we have used total time (10) and have available all nonempty sets $I_{\rho,C_1}(X)$ and $I_{\rho,C_2}(X)$. From Lemma 4 we observe that

$$\begin{aligned} |\mathrm{Emb}(P, G)| &= \sum_{\rho : C_1 \cap C_2 \to V(G)} |\mathrm{Emb}_\rho(P, G)| \\ &= \sum_{\rho : C_1 \cap C_2 \to V(G)} \sum_{X \subseteq V(G) \setminus \rho(C_1 \cap C_2)} (-1)^{|X|} \cdot |I_{\rho,C_1}(X)| \cdot |I_{\rho,C_2}(X)|. \end{aligned} \tag{11}$$

Thus, we can compute $|\mathrm{Emb}(P, G)|$ using the nonempty sets $I_{\rho,C_1}(X)$ and $I_{\rho,C_2}(X)$ by sorting the index tuples based first on ρ and then based on X. We then evaluate $|\mathrm{Emb}(P, G)|$ using the double sum in (11). The total time is bounded by (9) since the embeddings φ_j and indices ρ, C_j, X can both be represented using $O(k)$ words of $O(\log n)$ bits. ◀

6 The main algorithm

This section proves Theorem 2. Let P be a connected k-vertex graph with a vertex balancer of size b and let G be an n-vertex graph.

First, let us observe that we have trivially $|\mathrm{Sub}(P, G)| = 0$ unless $\Delta(P) \leq \Delta(G)$. Furthermore, we observe that $\mathrm{Aut}(P) = \mathrm{Emb}(P, P)$.

The main algorithm starts by verifying that both $k \leq n$ and $\Delta(P) \leq \Delta(G)$; if this is not the case, the algorithm gives the output 0 and stops.

Next, the algorithm computes $|\mathrm{Aut}(P)| = |\mathrm{Emb}(P,P)|$ using the algorithm in Lemma 6 with G set to equal P. Since $k \leq n$ and $\Delta(P) \leq \Delta(G)$, it is immediate from (9) that this computation of $|\mathrm{Aut}(P)|$ runs within the main time bound (2).

Finally, the algorithm computes $|\mathrm{Emb}(P,G)|$ using the algorithm in Lemma 6 and, using (3), gives the output

$$|\mathrm{Sub}(G,P)| = \frac{|\mathrm{Emb}(P,G)|}{|\mathrm{Aut}(P)|}\,.$$

Since (9) is bounded by (2), the total running time is bounded by (2). This completes the proof of Theorem 2.

7 Corollaries for elementary connected graphs

This section establishes Corollary 3. We start with a straightforward lemma on balancers.

▶ **Lemma 7.**

1. *A k-vertex path admits a balancer of size $2 - (k \bmod 2)$.*
2. *A k-vertex cycle admits a balancer of size $2 + (k \bmod 2)$.*
3. *A k-vertex tree for $k \geq 3$ admits a balancer of size at most $\lceil k/3 \rceil$.*

Proof. A k-vertex path $v_1, \ldots, v_k$ contains the balancer $\{v_{\lceil k/2 \rceil}\}$ for odd k and the balancer $\{v_{\lceil k/2 \rceil}, v_{\lceil k/2 \rceil + 1}\}$ for even k. A k-vertex cycle $v_1, \ldots, v_k$ for $k \geq 2$ contains the balancer $\{v_1, v_{\lceil k/2 \rceil}\}$ for even k and the balancer $\{v_1, v_{\lceil k/2 \rceil}, v_k\}$ for odd k.

We turn to the third item. Every k-vertex tree contains a *centroid* vertex c, which cuts it into subtrees $T_1, \ldots, T_r$ of size $k_1, \ldots, k_r$ with $k_i \leq k/2$, $\sum_i k_i = k - 1$, and $r \geq 2$. Put the largest two subtrees, say T_1 and T_2, into C_1 and C_2, respectively, and add c to each. If $r = 2$ then $k_1 = k_2 = (k-1)/2$ and c itself is a singleton balancer. Otherwise, add each of the remaining $r - 2$ trees, smallest first, to C_1 or C_2 such that $|C_1|$ and $|C_2|$ both remain at most $(k-1)/2$. This process continues until the last tree, say T_3, which is instead added to *both* C_1 and C_2. If C_1 and C_2 now have unequal size, say $|C_1| > |C_2|$ then repeatedly remove leaf nodes of T_3 from C_1 until $|C_1| = |C_2|$. The resulting balancer $C_1 \cap C_2$ consists of c together with a subtree of T_3, so we have

$$|C_1 \cap C_2| \leq k_3 + 1\,,$$

and since

$$k_3 \leq k_2 \leq k_1 \leq \tfrac{1}{3}(k-1)\,, \tag{12}$$

so we see that the balancer is roughly $\frac{1}{3}k$. For the precise bound in the lemma, the proceed by cases. If $(k \bmod 3) = 1$ then $\frac{1}{3}(k-1)$ is an integer, so we can just write

$$k_3 + 1 \leq \tfrac{1}{3}(k-1) + 1 = \lfloor (k-1)/3 \rfloor + 1 = \lceil k/3 \rceil\,.$$

If $(k \bmod 3) \in \{0, 2\}$ then the bound (12) cannot hold with equality, since otherwise the total number of vertices would be $k_1 + k_2 + k_3 + 1 = 1 \pmod 3$. Thus,

$$k_3 + 1 < \tfrac{1}{3}(k-1) + 1 = \tfrac{1}{3}k + \tfrac{2}{3} \leq \lceil k/3 \rceil + 1\,.$$

Since both sides of this strict inequality are integers, the bound in the lemma holds. ◀

Let us now proceed with a proof of Corollary 3. Recalling (2), for a connected k-vertex pattern P with balancer size b, the main algorithm runs in time

$$O\left((2\Delta-2)^{\frac{k+b}{2}}2^{-b}\frac{n}{\Delta}k^2\log n\right).$$

For a k-vertex path, we have $b = 2-(k \bmod 2) \leq 2$ by Lemma 7 and thus

$$\frac{k+b}{2}-1=\left\lfloor\frac{k}{2}\right\rfloor$$

implies the claimed running time

$$O\big((2\Delta-2)^{\lfloor\frac{k}{2}\rfloor}nk^2\log n\big).$$

For a k-vertex cycle, we have $b = 2+(k \bmod 2) \leq 3$ by Lemma 7 and thus

$$\frac{k+b}{2}-1=\left\lceil\frac{k}{2}\right\rceil$$

implies the claimed running time

$$O\big((2\Delta-2)^{\lceil\frac{k}{2}\rceil}nk^2\log n\big).$$

For a k-vertex tree wtih $k \geq 3$, we have $b \leq \lceil k/3\rceil$ by Lemma 7 and thus

$$\frac{k+b}{2}-1\leq\frac{k+\lceil k/3\rceil}{2}-1\leq\left\lceil\frac{2k-3}{3}\right\rceil$$

implies the claimed running time

$$O\big((2\Delta-2)^{\lceil\frac{2k-3}{3}\rceil}nk^2\log n\big).$$

This completes the proof of Corollary 3.

Treewidth. The notion of balancer is reminiscent of, but different from, the "balanced separators" that appear in the study of the graph parameter treewidth. However the latter is both more permissive and more strict, and no general corollaries for graphs of bounded treewidth follow from our results.

To see this, we exhibit infinite families of graphs where the two notions differ.

One one hand, low treewidth is a global property that extends to all subgraphs. For instance, consider the k-vertex graph formed by connecting two r-cliques, where $r=\frac{1}{2}k-1$, by identifying one vertex in each clique with the endpoints of a 3-vertex path. (This is the r-barbell graph with a subdivided bridge.) This graph has linear treewidth $\frac{1}{2}k-2$, but admits a one-vertex balancer.

On the other hand, Definition 1 requires C_1 and C_2 to be connected, which the parts arising from a tree-decomposition need not be. For instance, consider the k-vertex graph T formed by from three r-vertex paths, where $r=\frac{1}{3}(k-1)$, by identifying one endpoint in each path with the leaves of the 4-vertex "Claw graph" $K_{1,3}$. This graph is a tree and thus has treewidth 1. Any partition of $V(T)$ into C_1 and C_2 must put at least two leaves into the same part, say C_1. Since $T[C_1]$ is connected, the unique path between these leaves must belong to C_1, so $|C_1| \geq 2r+1$. By the balancing condition, $|C_2| = |C_1| \geq 2r+1$. But then the balancer has size at least $|C_1 \cap C_2| = |C_1| + |C_2| - |C_1 \cup C_2| \geq 4r+2-(3r+1) = r+1 = \frac{1}{3}k+\frac{2}{3}$. (Note that this derivation matches the tree bound from Lemma 7, showing that neither construction can be improved.) We conclude that there is an infinite family of graphs of treewidth 1 whose balancers have size at least $\frac{1}{3}k$.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the Current Clique Algorithms are Optimal, So is Valiant's Parser. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*, pages 98–117, 2015.

2 Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and Counting Given Length Cycles. *Algorithmica*, 17(3):209–223, 1997.

4 Omid Amini, Fedor V. Fomin, and Saket Saurabh. Counting Subgraphs via Homomorphisms. *SIAM J. Discrete Math.*, 26(2):695–717, 2012.

5 Per Austrin, Petteri Kaski, and Kaie Kubjas. Tensor network complexity of multilinear maps. *CoRR*, abs/1712.09630, 2017.

6 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting Paths and Packings in Halves. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 578–586, 2009.

7 Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. Counting Thin Subgraphs via Packings Faster Than Meet-in-the-Middle Time. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 594–603, 2014.

8 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-Coding. In *STOC '18: Symposium on Theory of Computing, June 23–27, 2018, Los Angeles, CA, USA*, page 14. ACM, New York, NY, USA, 2018.

9 Yijia Chen and Jörg Flum. On Parameterized Path and Chordless Path Problems. In *22nd Annual IEEE Conference on Computational Complexity (CCC)*, pages 250–263, 2007.

10 Yijia Chen, Marc Thurley, and Mark Weyer. Understanding the Complexity of Induced Subgraph Isomorphisms. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Track A*, pages 587–596, 2008.

11 Radu Curticapean. Counting Matchings of Size k Is #W[1]-Hard. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 352–363, 2013.

12 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 210–223, 2017.

13 Radu Curticapean and Dániel Marx. Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 130–139, 2014.

14 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoret. Comput. Sci.*, 326(1-3):57–67, 2004.

15 Peter Floderus, Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Detecting and Counting Small Pattern Graphs. *SIAM J. Discrete Math.*, 29(3):1322–1339, 2015.

16 Jörg Flum and Martin Grohe. The Parameterized Complexity of Counting Problems. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 538, 2002.

17 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B. V. Raghavendra Rao. Faster algorithms for finding and counting subgraphs. *J. Comput. Syst. Sci.*, 78(3):698–706, 2012.

18 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

19 Alon Itai and Michael Rodeh. Finding a Minimum Circuit in a Graph. *SIAM J. Comput.*, 7(4):413–423, 1978.

20 Mark Jerrum and Kitty Meeks. Some Hard Families of Parameterized Counting Problems. *TOCT*, 7(3):11:1–11:18, 2015.

21 Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015.

22 Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Inf. Process. Lett.*, 74(3-4):115–121, 2000.

23 Christian Komusiewicz and Manuel Sorge. An algorithmic framework for fixed-cardinality optimization in sparse graphs applied to dense subgraph problems. *Discrete Applied Mathematics*, 193:145–161, 2015.

24 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 296–303, 2014.

25 L. Lovász. Operations with structures. *Acta Math. Acad. Sci. Hungar.*, 18:321–328, 1967.

26 Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Applied Mathematics*, 198:170–194, 2016.

27 J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Comment. Math. Univ. Carolin.*, 26(2):415–419, 1985.

28 Stephan Olariu. Paw-Free Graphs. *Inf. Process. Lett.*, 28(1):53–54, 1988.

29 Viresh Patel and Guus Regts. Computing the number of induced copies of a fixed graph in a bounded degree graph. *CoRR*, abs/1707.05186, 2017.

30 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC*, pages 887–898, 2012.

31 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding Four-Node Subgraphs in Triangle Time. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1671–1680, 2015.

32 Virginia Vassilevska Williams and Ryan Williams. Finding, Minimizing, and Counting Weighted Subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

33 Meirav Zehavi. Mixing Color Coding-Related Techniques. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1037–1049, 2015.