SIAM J. DISCRETE MATH.
Vol. 26, No. 2, pp. 695–717

COUNTING SUBGRAPHS VIA HOMOMORPHISMS[*]

OMID AMINI[†], FEDOR V. FOMIN[‡], AND SAKET SAURABH[§]

Abstract. We introduce a generic approach for counting subgraphs in a graph. The main idea is to relate counting subgraphs to counting graph homomorphisms. This approach provides new algorithms and unifies several well-known results in algorithms and combinatorics, including the recent algorithm of Björklund, Husfeldt, and Koivisto for computing the chromatic polynomial, the classical algorithm of Kohn et al. for counting Hamiltonian cycles, Ryser's formula for counting perfect matchings of a bipartite graph, and color-coding-based algorithms of Alon, Yuster, and Zwick. By combining our method with known combinatorial bounds, ideas from succinct data structures, partition functions, and the color coding technique, we obtain the following new results. The number of optimal bandwidth permutations of a graph on n vertices excluding a fixed graph as a minor can be computed in time $2^{n+o(n)}$, in particular, in time $\mathcal{O}(2^n n^3)$ for trees and in time $2^{n+\mathcal{O}(\sqrt{n})}$ for planar graphs. Counting all maximum planar subgraphs, subgraphs of bounded genus, or more generally subgraphs excluding a fixed graph M as a minor can be done in $2^{\mathcal{O}(n)}$ time. Counting all subtrees with a given maximum degree (a generalization of counting Hamiltonian paths) of a given graph can be done in time $2^{\mathcal{O}(n)}$. A generalization of Ryser's formula is, Let G be a graph with an independent set of size ℓ. Then the number of perfect matchings in G can be found in time $\mathcal{O}(2^{n-\ell}n^3)$. Let $\mathcal{H}$ be a graph class excluding a fixed graph M as a minor. Then the maximum number of vertex disjoint subgraphs from $\mathcal{H}$ in a graph G on n vertices can be found in time $2^{\mathcal{O}(n)}$. In order to show this, we prove that there exists a constant c_M depending only on M such that the number of nonisomorphic n-vertex graphs in $\mathcal{H}$ is at most c_M^n. Let F be a k-vertex graph of treewidth t and let G be an n-vertex graph. A subgraph of G isomorphic to F (if one exists) can be found in $\mathcal{O}(4.32^k \cdot k \cdot t \cdot n^{t+1})$ expected time using $\mathcal{O}(\log k \cdot n^{t+1})$ space.

Key words. exact exponential algorithms, subgraph isomorphism, homomorphism, coloring

AMS subject classifications. 05C85, 68R05

DOI. 10.1137/100789403

1. Introduction.

Given two undirected graphs F and G, a *homomorphism* from F to G is a mapping from the vertex set of F to that of G such that the image of every edge of F is an edge of G. Different important combinatorial properties of F, such as being k-colourable, may be viewed as graph homomorphisms to a particular graph G, see the book of Hell and Nešetřil [38] for a thorough introduction to the topic. Counting homomorphisms between graphs has applications in a variety of areas, including extremal graph theory, properties of graph products, partition functions in statistical physics and property testing of large graphs. We refer to the excellent survey of Borgs *et al.* [18] for references on counting homomorphisms.

There is an extensive literature on the computational complexity of graph homomorphism and counting homomorphisms. Hell and Nešetřil showed that for any fixed simple graph G, the problem whether there exists a homomorphism from F to G is solvable in polynomial time if G is bipartite, and NP-complete if G is not bipartite [37]. Dyer and Greenhill [26] completely characterized the dichotomy between P and #P-complete for counting homomorphisms from F to G. It appears that polynomial-time solvable cases arise only when G is an isolated vertex, a complete

[*]Received by the editors March 22, 2010; accepted for publication (in revised form) March 22, 2012; published electronically May 22, 2012. An extended abstract of this paper was presented at ICALP'09.
http://www.siam.org/journals/sidma/26-2/78940.html
[†]CNRS-DMA, École Normale Supérieure, Paris, France (omid.amini@m4x.org).
[‡]Department of Informatics, University of Bergen, Bergen, Norway (fomin@ii.uib.no).
[§]TCS, Institute of Mathematical Sciences, 600 113 Chennai, India (saket@imsc.res.in).

graph with all loops present, a complete bipartite graph without loops, or a disjoint union of these graphs. Extending a result of Grohe [35], Dalmau and Jonsson [21] proved that counting homomorphisms from a graph F in a given family $\mathcal{F}$ to an arbitrary graph G is in P if and only if all graphs in the family $\mathcal{F}$ have bounded treewidth (up to the assumption from parameterized complexity that FPT $\neq$ #W[1]).

In this paper we design *exact* and *parameterized* algorithms for counting the number of (not necessarily induced) subgraphs isomorphic to a given graph F in a general graph. We refer to the book [32] for an introduction to exact algorithms and to the book of Downey and Fellows [24] for an introduction to parameterized complexity. The number of isomorphic subgraphs can be found in polynomial time when the size of the pattern graph F is constant [60] and in linear time when the host graph G is planar [27]; see also [23, 28]. For any graph G with n vertices and m edges, all subgraphs of G isomorphic to a given graph F can be counted by trying all possible edge subsets of G and for each subset checking if the obtained graph is isomorphic to F. This algorithm runs in time $2^{m+o(n)}$ by making use of an algorithm due to Babai [4] to check isomorphism in time subexponential in n. Another approach is to try all the permutations of the vertices of G and F, and for each of these permutations, to compare vertex neighborhoods. This will give us running time $\mathcal{O}(n!n^2) = 2^{\mathcal{O}(n \log n)}$. While it is an open question whether subgraph isomorphism can be solved in time $2^{\mathcal{O}(n)}$, there are many special cases, depending on the structure of the graph F, for which $2^{\mathcal{O}(n)}$ algorithms are known. Many natural problems such as Hamiltonian Cycle, Perfect Matching, Graph Coloring, Bandwidth Minimization, Triangle Packing, and many others, can be seen as a subgraph isomorphism problem, and for each of these problems, there are $2^{\mathcal{O}(n)}$ time algorithms known in the literature. However, all known algorithms for these problems are tailored to their specific properties.

The main idea behind our results is to reduce the problem of counting subgraphs of a nonlabeled graph G isomorphic to a given graph F to counting homomorphisms from F to G. Let $\text{sub}(F, G)$ denote the number of distinct (not necessarily induced) copies of a graph F contained in a graph G. Let also $\text{hom}(F, G)$ and $\text{inj}(F, G)$ be the number of homomorphisms and injective homomorphisms from F to G, respectively. The idea of relating $\text{hom}(F, G)$ and $\text{inj}(F, G)$ is not new in graph theory. Lovász [47, 18] gave the following identities relating $\text{hom}(F, G)$ and $\text{inj}(F, G)$. For an equivalence relation Θ on $V(F)$, or equivalently for a partition Θ of the vertex set $V(F)$, let F/Θ be the graph obtained from G by identifying vertices that belong to the same equivalence class of Θ. Thus two nodes x, y of F/Θ are adjacent if and only if there are $u \in x$ and $v \in y$, such that $uv \in E(F)$. Then

$$\text{inj}(F, G) = \sum_{\Theta} \mu(\Theta) \text{hom}(F/\Theta, G),$$

where

$$\mu(\Theta) = \prod_{A \in \Theta} \left((-1)^{|A|-1} (|A| - 1)! \right),$$

with the product running over all the equivalence classes of Θ and the sum running over all the equivalence relations (equivalently, over all partitions of $V(F)$). From an algorithmic point of view the above formula is not efficient because the number of equivalence relations is generally too large to be meaningful even for simple graphs such as the graph containing n isolated vertices. We give an alternative formula which is helpful in counting "simple structures" in time exponential in the size of the target graph G, i.e., $|V(G)|$. Let us denote by $\text{aut}(F)$ the number of automorphisms of F, that is bijective homomorphisms from F to itself. Also for a subset $W \subset V(G)$, we simply write $G \setminus W$ to denote the induced graph of G on $V(G) \setminus W$.

Our first result shows that if $|V(F)| = |V(G)|$, then

$$\text{sub}(F,G) = \frac{\text{inj}(F,G)}{\text{aut}(F)} = \frac{\sum_{W \subseteq V(G)} (-1)^{|W|} \text{ hom}(F, G \setminus W)}{\text{aut}(F)}. \tag{1}$$

This can be seen as a generalization of inclusion-exclusion-based formulas which were used for some counting problems, including counting the number of perfect matchings in a graph [11, 55], counting Hamiltonian cycles [6, 41, 43], and computing the chromatic polynomial of a graph [12, 44]. The main advantage of using graph homomorphisms is that despite their expressive power, graph homomorphisms from many structures can be counted efficiently.

1.1. Our results and related work. We start by proving (1) and Theorem 1, which is our main tool in the design of exact algorithms. We observe that a number of well-known classical and more recent results can be obtained as corollaries of Theorem 1. We demonstrate its power by reproving the following results. Let G be a graph on n vertices. Then the number of Hamiltonian cycles in G can be computed in time $2^n n^{\mathcal{O}(1)}$ and in polynomial space. (This result was rediscovered several times [6, 41, 43].) In the recent breakthrough paper of Björklund [10], a Monte Carlo algorithm detecting Hamiltonicity in time $1.657^n n^{\mathcal{O}(1)}$ is given.

The chromatic polynomial of G can be computed in time $2^{n+\mathcal{O}(\sqrt{n})}$ (this almost matches the running time of the celebrated result of Björklund, Husfeldt, and Koivisto [12, 14, 44]). The number of perfect matchings in a *bipartite* graph can be counted in time $2^{n/2} n^{\mathcal{O}(1)}$ (the classical result of Ryser [55]; see also Björklund and Husfeldt [12]).

We then use Theorem 1 and its variants to obtain improvements on the following.

Number of optimal permutations for bandwidth. The Bandwidth problem is a famous combinatorial problem where given an undirected graph G on n vertices, we wish to embed its vertices onto an integer line such that the maximum stretch of any edge, $bw(G)$, of G is minimized. The parameter $bw(G)$ is called bandwidth of G. The best known approximation algorithm for this problem is an $\mathcal{O}(\log^3 n \sqrt[4]{\log\log n})$-approximation algorithm due to Lee [46], which is a slight improvement of the earlier algorithm of Dungan and Vempala [25]. Saxe has shown that the bandwidth of a graph G, $bw(G)$, can be computed in time $\mathcal{O}(n^{bw(G)+1})$ [56]. When parameterized by bandwidth bw, the Bandwidth problem is also known to be W[t]-hard for all $t \geq 1$ [16].

Feige and Kilian [29] provided an exact algorithm computing the optimal bandwidth in time $10^n n^{\mathcal{O}(1)}$. Recently, several improvements were obtained by Cygan and Pilipczuk, resulting in an algorithm of running time $4.383^n n^{\mathcal{O}(1)}$ [20]. Feige and Talwar [31] showed that the bandwidth of a graph of treewidth at most t can be $(1+\varepsilon)$-approximated in time $2^{\mathcal{O}(\log n (t+\sqrt{\frac{n}{\varepsilon}}))}$. Vassilevska, Williams, and Woo [58] gave a hybrid algorithm which after a polynomial time test either computes the bandwidth of a graph in time $4^{n+o(n)}$ or provides a $\gamma(n)\log^2 n \log\log n$-approximation in polynomial time for any unbounded function γ.

The Bandwidth problem can be seen as a subgraph isomorphism problem, and by combining Theorem 1 with the techniques of counting homomorphisms on graphs of bounded treewidth, we obtain the following. The number of optimal bandwidth permutations of a graph of treewidth at most t on n vertices can be counted in time $2^n n^{t+\mathcal{O}(1)}$ and space $n^{t+\mathcal{O}(1)}$. When t is a constant, the algorithm uses polynomial space and runs in time $2^n n^{\mathcal{O}(1)}$. Independently, Cygan and Pilipczuk [20] announced a $2^n n^{\mathcal{O}(1)}$ algorithm that computes an optimal bandwidth layout in graphs of constant treewidth. However, their algorithm uses exponential space. Our result also yields a hybrid algorithm which after a polynomial time test either computes the minimum bandwidth of the graph in time $4^n n^{\mathcal{O}(1)}$ or provides an $\mathcal{O}(\log^{3/2} n)$-approximation in polynomial time, improving the algorithm presented in [58].

Counting perfect matchings. While a perfect matching in a graph can be found in polynomial time, the problem of counting the number of perfect matchings is #P-complete [57]. For bipartite graphs on n vertices, the best known exact algorithm for counting perfect matchings is to apply Ryser's formula for the permanent [55], which runs in time $\mathcal{O}(1.414^n)$. Björklund and Husfeldt [11] showed how to compute the number of perfect matchings of a graph in time $2^n n^{\mathcal{O}(1)}$ and polynomial space. Koivisto [45] showed how to count perfect matchings in time $\mathcal{O}(1.6181^n)$ and exponential space.

We generalize the classical result of Ryser by showing that if G contains an independent set of size k, then the number of perfect matchings in G can be found in time $\mathcal{O}(2^{n-k}n^3)$. Let us remark that the case of bipartite graphs is a special case as $k \geq \lfloor n/2 \rfloor$. In their recent work, Vassilevska and Williams [59] gave algorithms to count subgraphs with running time depending on the size of an independent set in pattern graph F.

Counting maximum subgraphs with a given property. Combining algorithms for counting homomorphisms with ideas from data structures, we give algorithms running in time $2^{\mathcal{O}(n)}$ for various problems asking to count the number of subgraphs with specific properties in an n-vertex graph. For example, it is possible to count maximum planar subgraphs, subgraphs of bounded genus, or, even more generally, subgraphs excluding a fixed graph M as a minor in time $2^{\mathcal{O}(n)}$. The last result requires a new combinatorial bound on the number of nonisomorphic unlabeled M-minor-free graphs which implies as a corollary the main theorem of Norine et al. [52] on minor-closed families. This answers affirmatively an open problem of Bernardi, Noy, and Welsh in [9], where they ask whether the number of unlabeled graphs of size n in a given minor-closed class of graphs $\mathcal{F}$ can be bounded above by d^n for some constant d (Problem 5 of section 4 in [9]). We also obtain a number of algorithms for counting spanning trees with different degree conditions. These structures can be seen as generalizations of Hamiltonian paths. Prior to our work, the only known algorithm for many of the problems above was to try all possible permutations of vertex sets, which takes time $n!\, n^{\mathcal{O}(1)}$.

Packing problems. Given a graph class $\mathcal{H}$, the packing problem asks for the maximum number of vertex disjoint subgraphs of G, all of which belong to $\mathcal{H}$. (See section 5.6 for a more precise definition.) We also show how to answer in time $2^{\mathcal{O}(n)}$ the packing problem from a certain class $\mathcal{H}$, where $\mathcal{H}$ is a subclass of the class of all graphs excluding a fixed graph M as a minor. In particular the Maximum Vertex Disjoint Cycles problem can be solved in time $2^{n+\mathcal{O}(\sqrt{n})}$. For these problems, no $2^{\mathcal{O}(n)}$ time algorithms were previously known.

Parameterized algorithms. By applying the inclusion-exclusion idea, it is possible to refine the celebrated color coding technique of Alon, Yuster, and Zwick [2]. Their probabilistic algorithm determines whether a given graph G contains a fixed graph F as a subgraph and works in two stages. First, one colors the vertices of G at random, and then one performs dynamic programming on the colored graph in order to find an isomorphic subgraph of F whose vertices have distinct colors. In [2], the authors provide an algorithm for the case when F is a forest, and then they mention that this algorithm can be generalized to an algorithm that finds a k-vertex graph F of treewidth t in an n-vertex graph G (if such a copy exists) in expected time $2^{\mathcal{O}(k)}n^{t+1}$. One of the significant disadvantages of using dynamic programming with color coding is that it requires exponential space. By combining ideas based on inclusion-exclusion and graph homomorphisms with color coding, we provide a polynomial space algorithm that in expected time $\mathcal{O}((2e)^k \cdot k \cdot t \cdot n^{t+1})$ finds a k-vertex graph F of treewidth t in an n-vertex graph G (if such a copy exists). This algorithm can be derandomized resulting in a deterministic algorithm which solves the problem in time $\mathcal{O}((2e)^{k+o(k)} \cdot k \cdot t \cdot n^{t+1})$ and space $\mathcal{O}(\log k \cdot n^{t+1})$. Finally, by extending the approach of Hüffner, Wernicke, and Zichner [40] that was used to speed up the algorithm of [2] for paths, we prove that a k-vertex graph F of treewidth t in an n-vertex graph G can be found in $\mathcal{O}(4.32^k \cdot k \cdot t \cdot n^{t+1})$ expected time using $\mathcal{O}(\log k \cdot n^{t+1})$ space.

The rest of the paper is organized as follows. In sections 2 and 3, we provide necessary definitions and some preliminary results. In section 4, we show that several classical results form the area of exact algorithms can be obtained by making use of graph homomorphisms. Section 5 provides new applications of our approach mentioned above. In section 6, we revisit the color coding approach of Alon, Yuster, and Zwick [2].

2. Preliminaries.

Let G be a simple undirected graph without self loops and multiple edges. The set of vertices and the set of edges of G are denoted by $V(G)$ and $E(G)$, respectively. For a subset $W \subseteq V(G)$, the subgraph of G induced by W is denoted by $G[W]$. To simplify the notation, for a subset $W \subset V$, we write $G \setminus W$ to denote $G[V \setminus W]$. For a given vertex $v \in V(G)$ and a subset $W \subseteq V(G)$, we denote by $deg_W(v)$ the number of vertices in W which are adjacent to v.

A *tree decomposition* of a graph G is a pair (X, U), where U is a tree whose vertices are called *nodes*, and $X = (\{X_i \mid i \in V(U)\})$ is a collection of subsets of $V(G)$ such that

1. $\bigcup_{i\in V(U)} X_i = V(G)$;
2. for each edge $vw \in E(G)$, there is an $i \in V(U)$ such that $v, w \in X_i$; and
3. for each $v \in V(G)$, the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of U.

The *width* of a tree decomposition $(\{X_i \mid i \in V(U)\}, U)$ equals $\max_{i\in V(U)}\{|X_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all the tree decompositions of G and is denoted by $\mathbf{tw}(G)$.

Given an edge $e = uv$ of a graph G, the graph G/e is obtained by contracting the edge uv, that is, we get G/e by identifying the vertices u and v and by removing all the loops and duplicate edges. A *minor* of a graph G is a graph M that can be obtained from a subgraph of G by contracting edges. A graph class $\mathscr{G}$ is *minor closed* if any minor of any graph in $\mathscr{G}$ is also an element of $\mathscr{G}$. A minor closed graph class $\mathscr{G}$ is *M-minor-free* or simply *M-free* if $M \notin \mathscr{G}$.

Given two graphs F and G, a graph *homomorphism* from F to G is a map f from $V(F)$ to $V(G)$, that is, $f: V(F) \to V(G)$, such that if $uv \in E(F)$, then $f(u)f(v) \in E(G)$. Furthermore, when the map f is injective, f is called an *injective homomorphism*. Given two graphs F and G, the problem of SUBGRAPH ISOMORPHISM asks whether there exists an injective homomorphism from F to G. We also recall that $\hom(F, G)$, $\mathrm{inj}(F, G)$, and $\mathrm{sub}(F, G)$ denote the number of homomorphisms from F to G, the number of injective homomorphisms from F to G, and the number of distinct (non necessarily induced) copies of F in G, respectively. We need the following result relating $\mathrm{sub}(F,G)$ and $\mathrm{inj}(F,G)$. This result is folklore and we omit its proof here.

PROPOSITION 1. $\mathrm{sub}(F,G) = \mathrm{inj}(F,G)/\mathrm{aut}(F)$.

Since one can compute $\mathrm{aut}(F)$ for a graph F on n_F vertices in time $2^{\mathcal{O}(\sqrt{n_F \log n_F})}$ [5],[1] which is subexponential in n_F, this proposition allows us to focus on computing the value of $\mathrm{inj}(F,G)$.

3. Relating counting subgraphs to counting homomorphisms. We first give a formula expressing the number of injective homomorphisms from F to G in terms of the number of homomorphisms from F to G, using the principle of inclusion-exclusion.

THEOREM 1. *Let F and G be two graphs with $|V(G)| = |V(F)|$. Then*

$$\mathrm{inj}(F,G) \;=\; \sum_{W \subseteq V(G)} (-1)^{|W|}\, \mathrm{hom}(F, G \setminus W),$$

and this is also equal to $\sum_{W \subseteq V(G)} (-1)^{|V|-|W|}\, \mathrm{hom}(F, G[W])$.

Proof. We only prove the first part; the last claim is easily obtained by a change of variable W to $W' = V \setminus W$. To prove the theorem, we first show that if there is an injective homomorphism f from F to G, then its contribution to the sum is exactly one. Notice that since $|V(G)| = |V(F)|$, an injective homomorphism only contributes when $W = \emptyset$. From this we conclude that injective homomorphisms are counted only once in the right-hand side. Since we are counting homomorphisms, we also count maps which are not injective in the right-hand-side sum. Next we show that if a map h is not an injective homomorphism, then its total contribution to the sum is zero, which will conclude the proof of the theorem. Observe that since h is not an injective homomorphism and $|V(F)| = |V(G)|$, it misses some vertices of $V(G)$. Let $\widetilde{V} = \mathrm{im}(h)$ be the image of h in $V(G)$ and $X = V(G) \setminus \widetilde{V} \neq \emptyset$. We now observe that h is counted only when we are counting homomorphisms from F to $G \setminus W$ such that $W \subseteq X$. The total contribution of h in the sum, taking into account the signs, is

$$\sum_{i=0}^{|X|} \binom{|X|}{i} (-1)^i = (1-1)^{|X|} = 0,$$

and the theorem follows. ∎

Let us assume that we can count the number of graph homomorphisms from F to all the graphs $G[W]$ in time $t(n)$, where $|F| \le |G| = n$ and $W \subseteq V(G)$. Then, as a consequence of Theorem 1, we can compute the value of $\mathrm{inj}(F,G)$ in time $\mathcal{O}(2^n \cdot t(n))$ when the size of $V(F)$ and $V(G)$ is n. A natural question arising here is to extend this to the case when the size of $V(F)$, say, n_F, is less than $n = |V(G)|$. The easiest solution will be to enumerate all subsets V' of size n_F of $V(G)$ and then to compute $\mathrm{inj}(F, G[V'])$. However, this will take time $\mathcal{O}(\binom{n}{n_F} 2^{n_F} t(n))$, which in the worst case could be equal to $\mathcal{O}(3^n \cdot t(n))$. In the rest of this section we show how to extend Theorem 1 to the case when $|V(F)| < |G|$.

THEOREM 2. *Let F and G be two graphs with $|V(F)| = n_F \le |V(G)| = n$. Then*

$$\mathrm{inj}(F,G) = \sum_{Y \subseteq V(G), |Y| \le n_F} (-1)^{n_F - |Y|} \binom{n - |Y|}{n_F - |Y|} \mathrm{hom}(F, G[Y]).$$

[1]In fact, for a given graph F, Babai, Kantor, and Luks [5] solve the harder problem of computing the automorphism group of F and a set of generators for *aut*(F). We refer to section 7 of [7] for further discussion.

Proof. We have that

$$
\begin{aligned}
\operatorname{inj}(F,G) &= \sum_{W\subseteq V(G),|W|=n_F} \operatorname{inj}(F,G[W])\\
&\overset{\text{by Theorem 1}}{=} \sum_{W\subseteq V(G),|W|=n_F} \left(\sum_{Y\subseteq W} (-1)^{|W|-|Y|}\hom(F,G[Y])\right)\\
&= \sum_{W\subseteq V(G),|W|=n_F} \left(\sum_{Y\subseteq W} (-1)^{n_F-|Y|}\hom(F,G[Y])\right)\\
&= \sum_{Y\subseteq V(G),|Y|\le n_F} (-1)^{n_F-|Y|}\binom{n-|Y|}{n_F-|Y|}\hom(F,G[Y]).
\end{aligned}
$$

The last equality follows from the fact that for any subset Y with $|Y| \le n_F$, the value of $\hom(F, G[Y])$ is counted precisely for all those subsets W for which $Y \subseteq W$ and $|W| = n_F$. On the other hand, for every fixed Y, $\hom(F, G[Y])$ is counted once in the above sum for every superset W of Y of size n_F. The number of such sets W is precisely $\binom{n-|Y|}{n_F-|Y|}$. Furthermore, for all such sets, we have the same sign corresponding to Y, that is, $(-1)^{n_F-|Y|}$. This completes the proof. ☐

4. Classical results. In this section we give alternative algorithms for a few classical algorithms through the method of counting homomorphisms.

4.1. Counting Hamiltonian cycles: Kohn–Gottlieb–Kohn–Karp algorithm. Let #Ham(G) denote the number of Hamiltonian cycles in a graph G and let C_n be the cycle of length n; then $\operatorname{sub}(C_n, G) = \#\text{Ham}(G)$. It is easy to see that $\operatorname{aut}(C_n)= 2n$, and thus for any graph H, $\hom(C_n, H) = tr(A_H^n) = \sum_{i=1}^{n} \lambda_i^n$, where A_H is the adjacency matrix of H and $\lambda_1, \dots, \lambda_n$ are its eigenvalues (see, for example, [18]). Using these results and Theorem 1, we compute #Ham(G). We run through all vertex subsets of G, and for each subset W we compute the number of homomorphisms from C_n to $G[W]$ in polynomial time. Up to polynomial factor, the running time of this algorithm is proportional to the amount of vertex subsets of G, and we conclude that the algorithm runs in time $2^n n^{\mathcal{O}(1)}$ and uses polynomial space.

4.2. Chromatic polynomial: Björklund–Husfeldt–Koivisto algorithm. A *proper k-coloring* of a graph G is a function $f : V(G) \to \{1, \dots, k\}$ such that for every edge $uv \in E(G)$, $f(u) \neq f(v)$. A well-known polynomial associated with a graph G is its Chromatic Polynomial. The *rank* of the graph G is $r(G) = |V(G)|-\eta(G)$, where $\eta(G)$ is the number of connected components of G. The Chromatic Polynomial of G is defined as $\chi(G;x) = \sum_{E'\subseteq E(G)}(-1)^{|E'|}x^{|V(G)|-r(E')}$, where $r(E')$ is equal to the rank of the subgraph of G with vertex set $V(G)$ and the edge set E'. The polynomial derives its name due to the fact that for every fixed integer $k \ge 1$, $\chi(G;k)$ is the number of proper k-colorings of G. Also, $\chi(G;k) = \hom(G,K_k)$, where K_k is a complete graph of size k. (Recall that the chromatic number of G is the smallest integer $k > 0$ for which $\chi(G;k) > 0$.) To compute the chromatic polynomial of a graph, we also use homomorphisms. However, instead of homomorphisms into cliques, we look at homomorphisms with domains of specific structure.

A k-coloring of a graph G can also be viewed as a partition of the vertex set of the given graph into k independent sets, that is, a partition $(V_1, \dots, V_k)$ of $V(G)$ such that for every $i \in \{1, \dots, k\}$, $G[V_i]$ has no edges. For our purpose, we reformulate the problem of coloring as a problem of partitioning into k cliques in the complement graph. The complement of a graph G, denoted by $\overline{G}$, is the graph with the same vertex set as G, i.e., $V(\overline{G}) = V(G)$, and with $uv \in E(\overline{G})$ if and only if $uv \notin E(G)$. Then G can be partitioned into k independent sets if and only if $\overline{G}$ can be partitioned into k cliques. We model this as a problem of subgraph isomorphism as follows. We guess the sizes $t_1, t_2, \dots, t_k$ of these cliques, where $\sum_i t_i = n$. Then $\overline{G}$ can be partitioned into cliques of sizes $t_1, t_2, \dots, t_k$, respectively, if and only if there is a subgraph isomorphic to $F = \sqcup_{i=1}^{k} K_{t_i}$ in $\overline{G}$. Thus as long as we know the correct sizes of cliques, the problem of coloring G is a subgraph isomorphism problem for the complement of G.

To find the right sizes of the cliques, we can try all the possible combinations. Let $\mathcal{P}_k(n)$ be the set of all unordered partitions of an integer n into k parts. For every partition $\zeta = \{t_1, t_2, \dots, t_k\} \in \mathcal{P}_k(n)$, let $F(\zeta) = \sqcup_{i=1}^{k} K_{t_i}$. Then

$$\chi(G;k) = \sum_{\zeta \in \mathcal{P}_k(n)} k! \cdot \mathrm{sub}(F(\zeta), \overline{G}). \tag{2}$$

In order to estimate the size of $\mathcal{P}_k(n)$, we need a classical result from number theory giving an upper bound on the number of unordered partitions of n into k parts. Let $p(n)$ be the partition function which for every n is the number of partitions of n. The asymptotic behavior of $p(n)$ was given by Hardy and Ramanujan [36]:

$$p(n) \sim e^{\pi\sqrt{\frac{2n}{3}}}/4n\sqrt{3}, \text{ as } n \to \infty. \tag{3}$$

Furthermore, one can give an enumeration algorithm listing all partitions of n into k parts in time, up to polynomial factor, proportional to $p(n)$ [51]. This brings us to the following algorithm for computing $\chi(G;k)$. For every partition $\zeta = (t_1, t_2, \dots, t_k) \in \mathcal{P}_k(n)$, we want to compute the inner sum in (2). To compute (2), we have to know the value of $\mathrm{sub}(F(\zeta), \overline{G})$, and to compute this value we use Theorem 1. To implement Theorem 1, we have to compute the values of $\mathrm{aut}(F(\zeta))$, and $\mathrm{hom}(F(\zeta), \overline{G} \backslash W)$, where $W \subseteq V(G)$. The computation of $\mathrm{aut}(F(\zeta))$ is easy—the number of automorphisms of a complete graph on t vertices is $t!$. If $F(\zeta)$ consists of several connected components, then every automorphism maps a component (complete graph) either into itself or to a component of the same size. Let $n(x)$ be the number of components of size x in $F(\zeta)$ and let $x_1, x_2, \dots, x_p$, $p \le k$, be the sizes of the components in $F(\zeta)$. Note that x_i is not necessarily equal to t_i because it is possible in the partition ζ that for some $i \neq j$, $t_i = t_j$. Then $\mathrm{aut}(F(\zeta)) = \prod_{x \in \{x_1, x_2, \dots, x_p\}} n(x)!x!$, and this value is computable in polynomial time for each ζ.

To compute $\mathrm{hom}(F(\zeta), \overline{G} \setminus W)$, we observe that it is sufficient to count homomorphisms from every component of $F(\zeta)$. The following result for a graph F with several connected components is well known; see, e.g., [18].

Proposition 2. *If F has connected components $F_1, \dots, F_\ell$, then* $\mathrm{hom}(F, G) = \prod_{i=1}^{\ell} \mathrm{hom}(F_i, G)$.

However, every component of $F(\zeta)$ is a complete graph, and by Proposition 2, all we need are the values of $\mathrm{hom}(K_t, \overline{G} \setminus W)$. For every homomorphism f from K_t to $\overline{G} \setminus W$, the image of the complete graph K_t is a clique of size t in $\overline{G} \setminus W$. Therefore, $\mathrm{hom}(K_t, \overline{G} \setminus W) = \mathcal{T}[V(G) \setminus W][t]t!$, where $\mathcal{T}[V(G) \setminus W][t]$ is the number of cliques of size t in $\overline{G} \setminus W$.

Thus to finish all these computations, we have to find the number of cliques of size t in a graph. By making use of dynamic programming over vertex subsets

$W \subseteq V(G)$, we compute the numbers $\mathcal{T}[W][i]$, which is the number of cliques of size i in $\overline{G}[W]$. Dynamic programming is based on the observation that for $i > 0$, $\mathcal{T}[W][i] = \mathcal{T}[W \setminus \{v\}][i] + \mathcal{T}[N(v) \cap W][i-1]$ for some vertex v. By making use of this observation, one can compute the values $\mathcal{T}[W][i]$ for all $W \subseteq V(G)$ and $0 \leq i \leq n$ in time $\mathcal{O}(2^n n^2)$ and by making use of $2^n \times (n+1)$ space.

Putting all pieces together, we conclude with the following algorithm. We compute all the values $\mathcal{T}[W][i]$, $W \subseteq V(G)$, $0 \leq i \leq n$, and keep them in a table $\mathcal{T}$ of size $2^n \times (n+1)$. As mentioned, this table is computable in time $2^n \cdot n^{\mathcal{O}(1)}$ and it uses space $2^n \cdot (n+1)$. Then we loop through every partition $\zeta = (t_1, t_2, \ldots, t_k) \in \mathcal{P}_k(n)$ and compute the inner sum in (2). Once the table $\mathcal{T}$ is computed, the computations of $\hom(F(\zeta), \overline{G} \setminus W)$ in (2) for every $W \subseteq V$ take polynomial time. Thus for every partition ζ, it takes time $2^n \cdot n^{\mathcal{O}(1)}$ to compute $\mathrm{sub}(F(\zeta), \overline{G})$. The number of partitions we need to loop is at most $2^{\mathcal{O}(\sqrt{n})}$ and thus the running time of the algorithm computing chromatic polynomial is $2^{n+\mathcal{O}\sqrt{n}}$. The space used by the algorithm is $2^n \cdot (n+1)$.

4.3. Number of perfect matchings in bipartite graphs: Ryser's formula. Let G be a bipartite graph on an even number of vertices, say, n, with $V(G)$ being partitioned into L and R of the same size. Then Ryser's formula [55] says that

$$\#\,\mathrm{PM}(G) = \sum_{X \subseteq R} (-1)^{|X|} \prod_{u \in L} \left(\sum_{v \notin X} 1_{[uv \in E(G)]} \right),$$

where $\#\mathrm{PM}(G)$ is the number of perfect matchings in G. The sum $\sum_{v \notin X} 1_{[uv \in E(G)]}$ counts the number of neighbors of u not in X. Thus, we can count the number of perfect matchings in a bipartite graph in time $\mathcal{O}(2^{n/2} n^2)$. If we take F as $n/2$ disjoint copies of an edge, then $\#\,\mathrm{PM}(G)=\mathrm{sub}(F,G)$. By using Theorem 1, it is easy to obtain an algorithm to compute the value of $\#\,\mathrm{PM}(G)$ in time $2^n n^{\mathcal{O}(1)}$. We will use the notion of saturating homomorphism in section 5.1 to faster compute $\#\mathrm{PM}(G)$; this in particular means in time $\mathcal{O}(2^{n/2} n^2)$ for bipartite graphs.

5. New applications. In this section we give new applications of Theorems 1 and 2 and show their wider applicability.

5.1. Set saturating homomorphisms and Ryser's formula. In this subsection we give a faster poly-space algorithm for counting perfect matchings in graphs with large independent sets. To do so, we first generalize the notion of graph homomorphism and prove a generalization of Theorem 1. Let S be a given subset of $V(G)$; then a homomorphism f from F to G is called *S-saturating* if

(a) $S \subseteq f(V(F))$, and

(b) for all $v \in S$, $|f^{-1}(v)| = 1$.

By S-$\hom(F,G)$ we denote the number of S-saturating homomorphisms. Observe that for $S = \emptyset$ an S-saturating homomorphism is simply a homomorphism. The following theorem is obtained similarly as in the proof of Theorem 1.

THEOREM 3. *Let F and G be two graphs with $|V(G)| = |V(F)|$ and $S \subseteq V(G)$. Then*

$$\mathrm{inj}(F,G) = \sum_{W \subseteq V(G) \setminus S} (-1)^{|W|} S\text{-}\hom(F, G \setminus W).$$

Proof. To prove the theorem, it will be enough to show that if there is an injective homomorphism f from F to G, then its total contribution to the sum on the right-hand side of the above equation is exactly one, while the total contribution to this sum of any noninjective S-saturating homomorphism from F to G is zero.

To show this, first we note that since $|V(G)| = |V(F)|$, all injective homomorphisms are S-saturating and they contribute only when $W = \emptyset$. From this we conclude that any injective homomorphism is counted exactly once in the above sum. This proves half the above claim. To show the other half, let h be a noninjective homomorphism from F to G. We show that its total contribution to the sum is zero, which will conclude the proof of the theorem. All the terms appearing in the sum concern only S-saturating homomorphisms, so we can assume that h is S-saturating. Since h is not an injective homomorphism and $|V(F)| - |V(G)|$, h has to miss some vertices of $V(G)$ in its image. In addition, h being S-saturating, it contains S in its image. Let $\widetilde{V} = \mathrm{im}(h)$ be the image of h in $V(G)$. We thus infer that $\widetilde{V} = V(G) \setminus X$ for some nonempty subset $X \subset V(G) \setminus S$. The only contribution of h are to those terms in the sum which count S-saturating homomorphisms from F to $G[V(G) \setminus W]$ for a subset $W \subseteq X$. The total contribution of h in the sum, taking into account the signs, is then

$$\sum_{i=0}^{|X|} \binom{|X|}{i} (-1)^i = (1-1)^{|X|} = 0,$$

which concludes the proof. □

We can now prove the next theorem.

THEOREM 4. *Let G be an n-vertex graph and $S \subseteq V(G)$ be an independent set of G. There is an algorithm which counts the number of perfect matchings of G in time $2^{n-|S|} \cdot n^{\mathcal{O}(1)}$.*

Proof. Let F be a matching of $n/2$ edges. Then sub(F,G)=#PM(G). By Theorem 3, we have that

$$\mathrm{inj}(F,G) = \sum_{W \subseteq V(G)\setminus S} (-1)^{|W|} \ \ S\text{-hom}(F, G \setminus W).$$

To prove the theorem, we show how to compute the value of S-hom$(F, G \setminus W)$. Let $S = \{v_1, \ldots, v_a\}$; then

$$S\text{-hom}(F, G \setminus W) = \binom{\frac{n}{2}}{a} a! \left(\prod_{i=1}^{a} \left(2 \cdot deg_{V(G)\setminus W}(v_i) \right) \right) \cdot \left(2 \cdot |E(G \setminus (W \cup S))| \right)^{\frac{n}{2} - a}.$$

To see this, first observe that S is an independent set in G. Hence, every S-saturating homomorphism from F to $G[V(G) \setminus W]$ has the property that for every vertex $x \in S$, it maps a unique edge of F to an edge incident to x. So we first choose a edges from $n/2$ edges of F, say, $\{f_1, f_2, \ldots, f_a\}$, and then map them to the edges incident to the vertices in S. Having selected these edges, we can assign them to the vertices in S in $a!$ ways. Fix an edge f_i; then it can be mapped to the edges incident on a vertex, $v_j \in S$, in $2 \cdot deg_{V(G)\setminus W}(v_j)$ ways. This follows from the fact that an edge f_i can map to any of the $deg_{V(G)\setminus W}(v_j)$ edges incident to v_j in $V(G) \setminus W$ and each edge can be mapped to another edge in two ways. The remaining $\frac{n}{2} - a$ edges are mapped to edges in $G \setminus (W \cup S)$. Proposition 2 combined with the fact that an edge can be mapped to another edge in two ways give the factor of $\left(2 \cdot |E(G \setminus (W \cup S))|\right)^{\frac{n}{2}-a}$ in the formula. Furthermore, aut(F) is equal to $2^{n/2}(n/2)!$. □

It is well known that the chromatic number of a graph is always at most its average degree (or degeneracy) plus one. Also, by Brooks' theorem, the chromatic number of a graph is at most the maximum vertex degree, unless the graph is complete or an odd cycle. Thus, by Theorem 4, we obtain the following result.

COROLLARY 1. *Let G be an n-vertex graph and let d and Δ be its average and maximum degrees. Then* $\#\mathrm{PM}(G)$ *is computable in time* $\min\{2^{n-\frac{n}{d+1}}, 2^{n-\frac{n}{\Delta}}\} \cdot n^{\mathcal{O}(1)}$. *In particular, if G is a bipartite graph, then one can find* $\#\mathrm{PM}(G)$ *in time* $2^{n/2} \cdot n^{\mathcal{O}(1)}$.

5.2. Subgraph isomorphism when *F* has bounded treewidth. The treewidth of a graph is one of the most fundamental notions in graph theory and graph algorithms. Here, we give an algorithm for counting subgraphs isomorphic to F in G, when F is given together with a tree decomposition of width t. We first mention an algorithm to compute $\hom(F, G)$, when F is a graph of bounded treewidth.

PROPOSITION 3 (see [22]). *Let F and G be two graphs on n_F and n vertices, respectively, given together with a tree decomposition of width t of F. Then* $\hom(F, G)$ *is computable in time* $\mathcal{O}(n_F \cdot n^{t+1} \min\{t, n\})$ *and space* $\mathcal{O}(\log n_F \cdot n^{t+1})$.

THEOREM 5. *Let F and G be two graphs on $n_F \le n$ vertices, respectively, given together with a tree decomposition of width t of F. Then* $\mathrm{sub}(F, G)$ *is computable in time*

$$\mathcal{O}\left(\sum_{i=0}^{n_F} \binom{n}{i} \cdot {n_F}^{t+2} \cdot t\right)$$

and space $\mathcal{O}(\log n_F \cdot n^{t+1})$.

Proof. By Theorem 2, we have

$$\mathrm{inj}(F, G) = \sum_{Y \subseteq V(G), |Y| \le n_F} (-1)^{n_F - |Y|} \binom{n - |Y|}{n_F - |Y|} \hom(F, G[Y]).$$

Hence, to compute $\mathrm{inj}(F, G)$, it is sufficient to go through all vertex subsets Y of G of size at most n_F and for each such subset to count homomorphisms from F to the subgraph induced by Y. By Proposition 3, each term $\hom(F, G[Y])$ in the above sum can be computed in time $\mathcal{O}(n_F \cdot {n_F}^{t+1} t)$ (since $t = \min\{t, n_F\}$) and space $\mathcal{O}(\log n_F \cdot {n_F}^{t+1})$. Thus, one can compute the value of $\mathrm{inj}(F, G)$ in time

$$\mathcal{O}\left(\sum_{i=0}^{n_F} \binom{n}{i} \cdot n_F \cdot {n_F}^{t+1} \cdot t\right)$$

and space $\mathcal{O}(\log n_F \cdot {n_F}^{t+1})$. By Proposition 1, we know that $\mathrm{sub}(F, G) = \mathrm{inj}(F, G) / \mathrm{aut}(F)$. Note that $\mathrm{aut}(F) = \mathrm{inj}(F, F)$. Using Theorem 1 together with Proposition 3, we can compute $\mathrm{aut}(F)$ in time $\mathcal{O}(2^{n_F} \cdot {n_F}^{t+2} \cdot t)$ and space $\mathcal{O}(\log n_F \cdot {n_F}^{t+1})$. This concludes the proof of the theorem. ☐

5.3. Bandwidth. The BANDWIDTH problem is the well-studied graph layout problem. A *layout* of a graph G on n vertices is a map $f : V(G) \to \{1, \dots, n\}$. In the BANDWIDTH problem, the objective is to find a layout function for a given graph G such that $\max_{uv \in E(G)} |f(u) - f(v)|$ is minimized.

The following lemma formulates the BANDWIDTH problem as an instance of the SUBGRAPH ISOMORPHISM problem. By P_n we denote a path on n vertices. For a graph G, the rth power of the graph is denoted by G^r. This graph is on the same vertex set $V(G)$, but we add an edge between two distinct vertices u and v if there is a path of length at most r between them in G. The following result is well known; see, e.g., [19].

PROPOSITION 4. *Let G be a graph on n vertices. Then G has a layout of bandwidth b if and only if there is an injective homomorphism from G to P_n^b.*

Using Proposition 4 together with Theorem 5, we obtain the following theorem.

THEOREM 6. *Given a graph G on n vertices together with a tree decomposition of width t, it is possible to find the number of optimum bandwidth layouts in time $\mathcal{O}(2^n \cdot n^{t+2} \cdot t)$ and space $\mathcal{O}(\log n \cdot n^{t+1})$.*

In particular, when G is a tree, then we can compute the number of optimum bandwidth layouts in time $\mathcal{O}(2^n \cdot n^3)$.

By the result of Alon, Seymour, and Thomas [1], every graph on n vertices that does not contain a graph M as a minor has treewidth at most $|V(M)|^{3/2}\sqrt{n}$.

THEOREM 7 (Alon, Seymour, and Thomas [1]). *Let h be an integer and let G be a graph with n vertices and with treewidth at least $h^{3/2}\sqrt{n}$. Then G has the complete graph K_h as a minor.*

By Theorem 6, we have the following.

COROLLARY 2. *The number of optimum bandwidth layouts of an n-vertex graph which excludes some fixed graph M as a minor is computable in time $2^{n+\mathcal{O}(\sqrt{n})}$.*

Theorem 6 can be used to improve a hybrid algorithm given in [58], which after a polynomial time test either computes the optimum bandwidth of a graph in time $4^{n+o(n)}$ or provides $\gamma(n)\log^2 n \log\log n$-approximation in polynomial time for any unbounded function γ

COROLLARY 3. *The* BANDWIDTH *problem admits an algorithm that given an n-vertex graph, G always produces after a polynomial time test, either a layout achieving the minimum bandwidth in $4^n \cdot n^{\mathcal{O}(1)}$ time or an $\mathcal{O}(\log^{3/2} n)$-approximation in polynomial time.*

Proof. Feige, Hajiaghayi, and Lee [30] gave a polynomial time algorithm which for any graph of treewidth k finds a tree decomposition of width at most $ck\sqrt{\log k}$. We run this algorithm first and find a tree decomposition of width $\omega(G)$. If $\omega(G) \geq n/\log n$, then the treewidth of the graph G is at least $\frac{n}{c(\log^{3/2} n)}$ and hence the optimum bandwidth of the graph G is at least $\frac{n}{c(\log^{3/2} n)}$. Now we output any layout function f for the input graph G. This gives us a factor $c(\log^{3/2} n)$ approximation algorithm for the BANDWIDTH problem. Else, $\omega(G) < n/\log n$, and now we use Theorem 6 to find the number of optimum bandwidth layouts of graph G in time $4^n \cdot n^{\mathcal{O}(1)}$. ☐

5.4. Degree constrained spanning tree problem. The HAMILTONIAN PATH is one of the earliest known problems for which an exact algorithm with time complexity $2^n n^{\mathcal{O}(1)}$ was known. This problem can also be seen as a special case of finding a spanning tree with certain degree constrains on the vertices. More precisely, the DEGREE CONSTRAINED SPANNING TREE problem is defined as follows: Given a connected undirected graph G and a vector of size n, $\hat{a} = (a_1, a_2, \ldots, a_n)$, find a spanning tree T of G (if one exists) such that there is a bijective mapping $g : V(G) \to \{a_1, a_2, \ldots, a_n\}$ with the property that $deg_T(v) = g(v)$. A variation of the DEGREE CONSTRAINED SPANNING TREE called the MODIFIED DEGREE CONSTRAINED SPANNING TREE is defined by replacing the condition of $deg_T(v) = g(v)$ with $deg_T(v) \leq g(v)$ in the DEGREE CONSTRAINED SPANNING TREE.

The HAMILTONIAN PATH is an instance of the degree constraint spanning tree problem with the vector $(1, 2, \ldots, 2, 1)$. Other well-known problems which can be formulated as an instance of either the DEGREE CONSTRAINED SPANNING TREE or the MODIFIED DEGREE CONSTRAINED SPANNING TREE include the FULL DEGREE SPANNING TREE (a spanning tree which maximizes the number of vertices having the same degree in the graph and the tree) [42] and the MINIMUM DEGREE SPANNING

TREE (a spanning tree for which the maximum degree is minimized) [33, 34] problems. To solve the DEGREE CONSTRAINED SPANNING TREE and the MODIFIED DEGREE CONSTRAINED SPANNING TREE problems we need the following classical result of Otter from 1948 [53].

PROPOSITION 5 (Otter [53]). *The number of unlabeled trees on n vertices* $\mathcal{T}(n) \sim C\alpha^n n^{-5/2}$ *as* $n \to \infty$, *where* $C = 0.53495\ldots$ *and* $\alpha = 2.95576\ldots$.

Moreover, by the result of Beyer and Hedetniemi [8], it is possible to enumerate all nonisomorphic (unlabeled) trees in time $\mathcal{O}(\mathcal{T}(n))$.

THEOREM 8 (Beyer and Hedetniemi [8]). *There is a total ordering on the set of (unlabeled) rooted trees of size n and an algorithm to generate all the (unlabeled) rooted trees of size n by starting from the first element in the order and by following the ordering such that in addition the average time per step is bounded by a constant independent of n. As a consequence, it is possible to generate all the rooted trees of size n in time* $\mathcal{O}(\mathcal{T}(n))$.

We can now state the main result of this section.

THEOREM 9. *Let G be a graph on n vertices and* $\hat{a} = (a_1, \ldots, a_n)$ *be a vector of length n. Then we can count the number of feasible solutions to the* DEGREE CONSTRAINED SPANNING TREE *and the* MODIFIED DEGREE CONSTRAINED SPANNING TREE *in time* $\mathcal{O}(5.912^n)$.

Proof. We start with an algorithm that finds a feasible solution to the DEGREE CONSTRAINED SPANNING TREE (or the MODIFIED DEGREE CONSTRAINED SPANNING TREE) problem on a graph G on n vertices.

Step 1: Enumerate all nonisomorphic unordered trees T on n vertices and for the given tree T proceed as follows:
    Step 2: Check whether T is feasible with respect to the vector $\hat{a}$;
    Step 3: Count the number $\text{sub}(T, G)$ of subgraphs of G isomorphic to T;
Step 5: Output the sum of $\text{sub}(T, G)$ taken over all enumerated trees T.

The first step of the algorithm is done using the result of Beyer and Hedetniemi [8, Proposition 8], which gives an algorithm to enumerate all nonisomorphic (unlabeled) trees in time $\mathcal{T}(n)n^{\mathcal{O}(1)}$. By Proposition 5, we know that the number of unordered trees enumerated in Step 1 is at most 2.9558^n. Checking for the feasibility can be done by writing the degree sequence of T and the vector $\hat{a}$ in increasing order and checking whether the corresponding vectors are equal. Finally, the last step of the algorithm can be done using Theorem 5 in time $\mathcal{O}(2^n n^3)$ and space polynomial in n. Hence the running time of the algorithm is bounded by $\mathcal{O}(2.9558^n \cdot 2^n n^3) = \mathcal{O}(5.912^n)$. ∎

We solve the MINIMUM DEGREE SPANNING TREE problem by finding the smallest $2 \leq i \leq n-1$ for which the MODIFIED DEGREE CONSTRAINED SPANNING TREE problem returns yes with $\hat{a} = (i, i, \ldots, i)$, resulting in the following.

COROLLARY 4. *The* MINIMUM DEGREE SPANNING TREE *on a graph on n vertices can be solved in time* $\mathcal{O}(5.912^n)$.

5.5. Counting graphs excluding a fixed minor. In this section we apply our results to count planar subgraphs of maximum size or more generally maximum sized subgraphs that do not contain some fixed graph M as a minor. More precisely, we consider the MAXIMUM PLANAR SUBGRAPH and the MAXIMUM M-MINOR FREE SUBGRAPH problems. Here given a graph G the objective is to find a subset $E' \subseteq E(G)$ of maximum size such that the graph $G_{E'}$ on the vertex set $V(G)$ and the edge set E' is planar and M-minor-free, respectively.

A naïve algorithm for the above problems is to enumerate all edge subsets of the given graph for each subset test whether the subgraph induced by the edge set has the desired properties and output the feasible subgraph with the maximum number of edges. For a graph G on n vertices and m edges this algorithm will take $2^m \cdot n^{\mathcal{O}(1)}$ time. Let us remark that even for the decision version of these problems, no vertex exponential $c^n n^{\mathcal{O}(1)}$ time algorithms were known. The basic ideas used here are similar to the ones used for trees, namely, to prove that all unlabeled graphs on n vertices in the considered class can be enumerated in time $\mathcal{O}(c^n)$ for some constant c, and then for each element of the enumerated class, applying Theorem 5 to count the number of subgraphs of G isomorphic to it.

Let M be a fixed graph. Norine et al. [52] proved that the number of labeled n-vertex graphs of size n in a family of graphs excluding M as a minor is at most $n!c^n$ for some constant c (depending on M). We prove a more general result here, namely, that the number of unlabeled M-minor-free n-vertex graphs is at most c^n for some constant c depending only on M. Let us remark that since the number of labelings is at most $n!$, our result immediately yields the main theorem from [52].

THEOREM 10. *Let $\mathscr{G}$ be a family of unlabeled n-vertex graphs that do not contain some fixed graph M as a minor. Then there exists a constant c_M such that the number of nonisomorphic graphs in $\mathscr{G}$ is at most c_M^n. Moreover, the elements of $\mathscr{G}$ can be enumerated in time $\mathcal{O}(c_M^n)$.*

We prove the theorem by using results about geometric representation of minor-free graphs. *Book embedding* is a generalization of planar embedding to nonplanar surfaces in the form of a *book*, a collection of pages (half planes) joined together at the spine of the book (the shared boundary of all the half planes). The spine is identified with the real line and the vertices of the graph are embedded on the spine on integers 1 to n. The edges are distributed on the pages, so that edges residing on the same page do not intersect (forms a planar embedding of a subgraph of G). The minimum number of pages in which a graph can be embedded is its *page-number*. Malitz proved in [48] that any graph of genus g has page-number $\mathcal{O}(\sqrt{g})$. This result has been extended to minor free classes of graph by Blankenship and Oporowski [15].

THEOREM 11 (Blankenship and Oporowski [15]). *Let M be a fixed graph and $\mathcal{C}$ be the class of graphs excluding M as a minor. Then there is a constant $h = h(M)$ such that the page-number of every graph in $\mathcal{C}$ is at most h.*

Using these results, we can now present the proof of Theorem 10.

Proof of Theorem 10. This follows by combining the above theorem with a result of Munro and Raman [49] which encodes h-page graphs in $4hn + o(hn)$ bits with constant time adjacency queries.[2] Nevertheless, we provide a simple direct proof here without referring to the results of [49].

We prove the following claim by induction on the number of pages. The number of unlabeled n-vertex graphs that can be embedded in p pages is at most 2^{9pn}. For $p = 0$, i.e., when graphs have no edges, the claim follows trivially. Let us assume that the claim holds for some $p \geq 0$. Every graph embeddable into $p + 1$ pages can be formed from a graph with page-number p by adding one more page. Thus, the number of graphs with page-number $p + 1$ is at most the number of graphs that can be embedded into one page times the number of graphs of page-number at most p. Hence, it will be enough to obtain an upper bound on the number of graphs which can be embedded on one page.

But before proceeding, we first show that for all $n \geq 2$, the number of edges in a page is bounded by $2n - 3$ and this is tight. To see this, one can proceed by induction on n: the base cases $n = 2, 3$ are trivial. So suppose that $n \geq 4$. Assuming that the

[2]Special thanks to the referees for pointing us to this reference.

bound holds for all integers $m \leq n-1$, we show that it also holds for n. Either there is an edge between two vertices $i < j$ such that $2 \leq j-i \leq n-2$, or no such exists. If there is such an edge, between $i < j$ with $2 \leq j-i \leq n-2$, there is no edge from vertices inside the interval (i, j) to the vertices in $[1, i) \cup (j, n]$. By induction, the number of edges between vertices in the interval $[i, j]$ is bounded by $2(j-i+1)-3$. Similarly, removing all the vertices in (i, j) (which is nonempty) from the graph, by induction the number of edges between vertices in $[1, i] \cup [j, n]$ can be bounded by $2(n-j+i+1)-3$. The edge ij is counted twice; thus the total number of edges is bounded by $2(j-i+1)-3+2(n-j-i+1)-3-1 = 2n-3$. This bound is obviously tight by the inductive argument.

Otherwise, if there is no edge between vertices $i < j$ with $2 \leq j-i \leq n-2$, the total number of edges will be bounded by n which is smaller than $2n-3$, since $n \geq 4$.

We can now derive an upper bound on the number of graphs embeddable in one page. It is straightforward to see that each of these graphs can be obtained by the following procedure:

1. Take a parenthesis string consisting of $k \leq 2n-3$ pairs "(,)." Note that k here corresponds to the number of edges of the graph embedded on one page. The total number of parenthesis strings is the Catalan number of order k which is bounded by $4^k \leq 2^{4n}$.
2. For each parenthesis string, let $I_1, J_1, I_2, J_2, \ldots, I_l, J_l$ be the maximal intervals formed by consecutive half-parentheses of the same kind. In other words, I_1 is the first interval of all the consecutive half-parentheses of the form "(," J_1 is the first interval of all the consecutive half-parentheses of the form ")" which come after I_1, I_2 is the interval of all the consecutive half-parentheses of the form "(" which come after J_1, and so on. More precisely, I_s is the the interval of all the consecutive half-parentheses of the form "(" which come after J_{s-1}, and J_s is the the interval of all the consecutive half-parentheses of the form ")" which come after I_s. Partition each interval I_s and J_s into (potentially) smaller subintervals I_s^t and J_s^u. The total number of ways the partitioning is done can be bounded by the product of the number of ways a given interval I_s (resp., J_s) is partitioned, and this is bounded by $2^{\sum_s |I_s|} \times 2^{\sum_s |J_s|} = 4^k \leq 2^{4n}$.
3. Form all the pairs (parenthesis string, partition) consisting of a parenthesis system of step 1 and a partition of the intervals of step 2. The total number of these pairs is at most $2^{4n} \times 2^{4n} = 2^{8n}$. For each pair (S, P) consisting of a parenthesis string and a partitioning, form a graph $G(S, T)$ as follows. First from S form a graph Γ consisting of k different edges on the vertex set $1, \ldots, 2k$ respecting the parenthesis string. Namely, for a pair of parentheses "(,)," if "(" appears on the ith place and ")" appears on the jth place in the string (evidently $i \leq j$), then i is connected to j. Now identify all the vertices which are within the same interval of the partitioning P. More precisely, form the graph Γ/P where each interval I_s^t is contracted to one vertex (similarly, each interval J_s^u is contracted to one vertex). Note that the vertices of $G(S, T)$ have a natural total ordering induced by the ordering of natural numbers $1, \ldots, 2k$.
4. Let $G(S, T)$ be the graph obtained from step 3 for a pair (S, T). If the number of vertices of $G(S, T)$, denoted by $n_{S,T}$, is not larger than n, then choose $n_{S,T}$ numbers from $1, \ldots, n$ and identify the vertices of $G(S, T)$ in an order preserving way with these numbers. For each sequence $I \subset \{1, \ldots, n\}$, one obtains in this way a graph $G(S, T, I)$ embedded in one page. Note that there are $\leq 2^n$ ways to choose a subset I of $\{1, \ldots, n\}$

It follows from the above description of the graphs embeddable in one page that the total number of these graphs is at most $2^{8n} \times 2^n = 2^{9n}$. By the induction assumption, the number of graphs embeddable in p pages is at most 2^{9pn} and the claim follows.

By Theorem 11, all graphs from $\mathscr{G}$ have page-number bounded by a constant h_M depending only on M. Thus, the total number of unlabeled graphs of the class $\mathscr{G}$ on n vertices is bounded by $2^{9h_M n} = c_M^n$.

Finally, let us remark that an algorithm enumerating all the graphs of size n from $\mathscr{G}$, and running in time $\mathcal{O}(c_M^n)$, can be easily constructed following the steps of the proof. We enumerate all graphs with page-number at most h_M and for each graph check if it contains M as a minor. By the seminal work of Robertson and Seymour [54], this last step can be done in polynomial time, namely, testing if a graph G has a given graph M as minor can be done in polynomial time. $\square$

For simpler classes of graphs, such as planar graphs, one can obtain faster algorithms. For planar graphs we do not need to use heavy Robertson–Seymour machinery to check if a given graph is planar. There is a linear time algorithm of Hopcroft and Tarjan for checking if the input graph is planar [39]. Also to bound the number of nonisomorphic planar graphs, we can use the following result from the theory of succinct data structures.

PROPOSITION 6 (Bonichon et al. [17]). *Every connected planar graph with n vertices and m edges can be encoded in linear time with at most $4.91n + o(n)$ bits or $2.82m + o(m)$ bits.*

Combining our Theorem 10 and Proposition 6 with what we proved in Theorem 5 and the result of Alon, Seymour, and Thomas,[3] Theorem 7, we obtain the main result of this section.

THEOREM 12. *Given a graph G on n vertices, the counting version of the* MAXIMUM M-MINOR FREE SUBGRAPH *problem can be solved in time $\mathcal{O}(c^n) = 2^{\mathcal{O}(n)}$ for some constant $c = c_M$. In particular, for the counting version of* MAXIMUM PLANAR SUBGRAPH *we can obtain an algorithm running in time $2^{4.91n+o(n)}$. All these algorithms use $n^{\mathcal{O}(\sqrt{n})}$ space.*

5.6. $\mathcal{H}$-packing and some of its variants.

Let $\mathcal{H}$ and $\mathcal{G}$ be two graph classes. By *$\mathcal{H}$-subgraph* of G we mean any subgraph of G that belongs to $\mathcal{H}$. Given a graph $G \in \mathcal{G}$, the COVERING (or HITTING) problem asks for finding a subset W of $V(G)$ of minimum size which covers all the $\mathcal{H}$-subgraphs of G. Thus for any $\mathcal{H}$-subgraph H of G, $W \cap V(H) \neq \emptyset$.

On the other hand, the PACKING problem asks for finding a maximum number of vertex disjoint copies of $\mathcal{H}$-subgraphs in G. In other words, the packing number of G with respect to the class $\mathcal{H}$ is defined as

$$\begin{aligned}\mathbf{pack}_{\mathcal{H}}(G) = \max\ \{k \mid \exists \text{ a partition}\{V_1, \dots, V_k\} \text{ of } V(G) \text{ such that}\\ \forall i \in \{1, \dots, k\},\ \exists_{H \in \mathcal{H}} H \subseteq G[V_i]\}.\end{aligned}$$

Let M be a fixed graph. In this section we show that if $\mathcal{H}$ is a graph class excluding M as a minor (that is, no $H \in \mathcal{H}$ containing M as a minor), then there exists a constant c depending only on M such that it is possible to compute the value of $\mathbf{pack}_{\mathcal{H}}(G)$ in time $c^n n^{\mathcal{O}(1)}$ and space $n^{\mathcal{O}(\sqrt{n})}$ for any graph G on n vertices.

THEOREM 13. *Let G be a graph on n vertices, M be a fixed graph, and $\mathcal{H}$ be a subclass of M-minor-free graphs such that testing if a graph $H \in \mathcal{H}$ can be*

[3]Note that this theorem in particular says that the treewidth of an n-vertex graph excluding a fixed graph as a minor is $\mathcal{O}(\sqrt{n})$.

performed in time $|V(H)|^{\mathcal{O}(1)}$. Then the value of $\mathbf{pack}_{\mathcal{H}}(G)$ *can be computed in time $c^n n^{\mathcal{O}(1)} = 2^{\mathcal{O}(n)}$ and space $n^{\mathcal{O}(\sqrt{n})}$, where c is a constant depending only on M.*

Proof. Given a graph class $\mathcal{H}$ and the input graph G, to compute the value of $\mathbf{pack}_{\mathcal{H}}(G)$, we proceed as follows:

- (i) For every pair (l, p), $0 \le l \le p \le n$, proceed as follows.
- (ii) Enumerate the unordered partitions of p into l parts.
- (iii) For a fixed partition $\zeta = (p_1, p_2, \dots, p_l)$, enumerate all the elements $(H_1, \dots, H_l)$ of the product $\mathcal{H}_{p_1} \times \mathcal{H}_{p_2} \times \cdots \mathcal{H}_{p_l}$, where $\mathcal{H}_{p_i}$ is the set of all elements of $\mathcal{H}$ of size p_i.
- (iv) Let F be the disjoint union of $H_1, \dots, H_l$. Compute $\text{sub}(F, G)$.
- (v) Return the maximum l for which there exists a p such that in step (iv) the value of $\text{sub}(F, G)$ is nonzero.

The correctness of the algorithm is easy to see. To see the time complexity, observe that the time taken in step (i) is bounded by n^2 and step (ii) takes $2^{\mathcal{O}(\sqrt{n})}$ using the asymptotic formula (3). Step (iii) of the algorithm depends of the size of $\mathcal{H}_{\mathcal{P}}$. Hence if the number of graphs on x vertices in $\mathcal{H}$ is bounded by d^x for d a constant, then $|\mathcal{H}_{\mathcal{P}}| \le d^p$. Since $\mathcal{H}$ is a graph class excluding a fixed graph M as a minor, by Theorem 10 there exists a constant c_M such that $|\mathcal{H}_x| \le c_M^x$ for all $x \in \mathbb{N}$. We enumerate all M-minor-free graphs by making use of Theorem 10, and for each graph we check in polynomial time if it belongs to $\mathcal{H}$. This estimates the time taken in this enumeration step of the algorithm. Because F excludes some fixed graph as a minor, its treewidth is $\mathcal{O}(\sqrt{n})$. Then step (iv) of the algorithm can be done in time $2^{n+\mathcal{O}(\sqrt{n})}$ and space $n^{\mathcal{O}(\sqrt{n})}$ using Theorem 5. Choosing $c = 2c_M$ completes the theorem. ∎

In what follows we give a few corollaries of Theorem 13 when $\mathcal{H}$ is some specific graph class. Let $\mathcal{H}^c = \{C_q \mid \text{simple cycle of length } q,\ q \in \mathbb{N},\ q \ge 3\}$. It is easy to see that for a simple undirected graph G, the value of $\mathbf{pack}_{\mathcal{H}^c}(G)$ is equal to the maximum number of vertex disjoint cycles in G. For every fixed partition $\zeta = (p_1, \dots, p_l)$ of p into l integers with $\sum_{i=1}^{l} p_i = p$ and $p_i \ge 3$ we have $|\mathcal{H}^c_{\mathcal{P}}| = 1$. In this case step (iv) of the algorithm takes $\mathcal{O}(2^n n^5)$ time by Theorem 5. Similarly if we replace $\mathcal{H}^c$ with $\mathcal{H}^o$, which contains all odd cycles of length at least 3, we get the problem of computing the maximum number of vertex disjoint odd cycles in G. If we want to find the maximum number of vertex disjoint triangles or the maximum number of vertex disjoint cycles of fixed length l, then we do not need the partition based enumeration. In this case we just guess the number of copies of the l-length cycle in the input graph. The problem of finding the maximum number of vertex disjoint cycles of length l is called MAXIMUM l-CYCLE PACKING. The other problems corresponding to finding a maximum number of vertex disjoint cycles or finding the maximum number of odd cycles are similarly defined. Let us remark that if all graphs in class $\mathcal{H}$ from Theorem 13 are of treewidth at most t, then the space $n^{\mathcal{O}(\sqrt{n})}$ claimed in Theorem 13 can be improved to $n^{\mathcal{O}(t)}$. Because the treewidth of a cycle is two, this brings us to the following corollary.

COROLLARY 5. *Given a graph G on n vertices, the* MAXIMUM VERTEX DISJOINT CYCLES *and* MAXIMUM ODD SIZED VERTEX DISJOINT CYCLES *problems can be solved in time $2^{n+\mathcal{O}(\sqrt{n})}$, whereas the* MAXIMUM l-CYCLE PACKING *problem can be solved in time $\mathcal{O}(n^6 \cdot 2^n)$. All these algorithms take polynomial space.*

6. Poly-space color coding.

In this section we show how the ideas of counting homomorphisms and inclusion-exclusion combined with the color coding technique of Alon, Yuster, and Zwick [2] provide polynomial space parameterized algorithms.

6.1. Deterministic algorithm. Let $c\colon V(G) \to \{1, 2, \ldots, k\}$ be a coloring (not necessarily proper) of the vertex set of a graph G in k colors. Thus $V_i = c^{-1}(i)$ is not necessarily an independent set. For a graph F on k vertices, we say that an injective homomorphism f from F to G is *colorful* if each vertex of the image of F is colored by a distinct color. We denote the number of colorful injective homomorphisms from a graph F to a colored graph G by $\text{col-inj}(F, G)$. Let us remark that the number of colorful copies of F in G is equal to $\text{col-inj}(F, G)/\text{aut}(F)$. Let G^* be the graph obtained from G by deleting the monochromatic edges, that is, by turning each color class V_i into an independent set. The following simple relation between the number of colorful copies of F in G and in G^* follows directly from the definition of colorful homomorphisms.

LEMMA 1. *Let $c\colon V(G) \to \{1, 2, \ldots, k\}$ be a coloring of G. Then* $\text{col-inj}(F, G) = \text{col-inj}(F, G^*)$.

The following theorem is the main reason why the dynamic programming algorithm in the color coding technique of Alon, Yuster, and Zwick can be replaced by a polynomial space algorithm.

THEOREM 14. *Let $c\colon V(G) \to \{1, 2, \ldots, k\}$ be a coloring of G and $V_i = c^{-1}(i)$. Then*

$$\text{col-inj}(F, G) = \text{col-inj}(F, G^*) = \sum_{I \subseteq \{1,2,\ldots,k\}} (-1)^{|I|} \hom(F, G^* \setminus \cup_{i \in I} V_i)$$

$$= \sum_{I \subseteq \{1,2,\ldots,k\}} (-1)^{k-|I|} \hom(F, G^*[\cup_{i \in I} V_i]).$$

Proof. The proof of this theorem is almost identical to the proof of Theorem 1. To prove the theorem, we first show that if there is a colorful injective homomorphism f from F to G, then its contribution to the sum is exactly one. Notice that since $|V(F)| = k$, all colorful injective homomorphisms contribute only when $I = \emptyset$. From this we conclude that colorful injective homomorphisms are counted only once in the right-hand side.

Next we show that if a map h is not a colorful injective homomorphism, then its total contribution to the sum is zero, which will conclude the proof of the theorem. Let $\chi(h(F))$ be the set of colors on the vertices of $h(F)$. Observe that since h is not a colorful injective homomorphism, it misses vertices from some color classes. Hence $X = \{1, \ldots, k\} \setminus \chi(h(F))$ is nonempty. We now observe that h is counted only when we are counting homomorphisms from $V(F)$ to $G^* \setminus \cup_{i \in I'} V_i$ such that $I' \subseteq X$. The total contribution of h in the sum, taking into account the signs, is

$$\sum_{i=0}^{|X|} \binom{|X|}{i} (-1)^i = (1-1)^{|X|} = 0.$$

Thus, we have shown that if h is not a colorful injective homomorphism, then its contribution to the sum is zero. The second equality could be proven similarly, and we omit its proof. ◻

By a classical result of Arnborg, Corneil, and Proskurowski [3], a tree decomposition of a k-vertex graph F of width t, if any, can be computed in $O(k^{t+2})$ time. When this running time is dominated by other steps of the algorithm considered, we will just consider this decomposition as given. Therefore, a combination of Proposition 3 and Theorem 14 yields the following result.

Corollary 6. *Let F be a k-vertex graph of treewidth t. Then for any coloring $c\colon V(G) \to \{1, 2, \ldots, k\}$ of an n-vertex graph G, the value of* col-inj(F, G) *is computable in time $\mathcal{O}(2^k \cdot k \cdot t \cdot n^{t+1})$ and space $\mathcal{O}(\log k \cdot n^{t+1})$.*

Theorem 15. *Let F be a k-vertex graph of treewidth t and let G be an n-vertex graph. A subgraph of G isomorphic to F (if one exists) can be found in either $\mathcal{O}((2e)^k \cdot k \cdot t \cdot n^{t+1})$ expected time and $\mathcal{O}(\log k \cdot n^{t+1})$ space or deterministically in time $\mathcal{O}((2e)^{k+o(k)} \cdot k \cdot t \cdot n^{t+1})$ and space $\mathcal{O}(\log k \cdot n^{t+1})$. Here, e is the base of natural logarithm.*

Proof. The proof of this theorem follows along the lines of [2]. We color the vertices of $V(G)$ uniformly at random from the set $\{1, \ldots, k\}$. Then the probability that a copy of F in $V(G)$, if there is one, has become colorful is at least $k!/k^k > e^{-k}$. Given this random coloring we can compute the value of col-inj(F, G) in time $\mathcal{O}(2^k k n^{t+1} \min\{k, t\})$ using Corollary 6. If col-inj$(F, G) > 0$, we know that there exists a subgraph of G isomorphic to F. Hence the expected running time to find a subgraph of G isomorphic to F (if one exists) is $\mathcal{O}((2e)^k \cdot k \cdot t \cdot n^{t+1})$.

To obtain the deterministic algorithm we need to replace the first step of the algorithm where we color the vertices of $V(G)$ uniformly at random from the set $\{1, \ldots, k\}$ with a deterministic one. This is done by making use of an (n, k, k)-*perfect hash family*. A (n, k, k)-perfect hash family, $\mathcal{H}$, is a set of functions from $\{1, \ldots, n\}$ to $\{1, \ldots, k\}$ such that for every subset $S \subseteq \{1, \ldots, n\}$ of size k there exists a function $f \in \mathcal{H}$ such that f is injective on S. That is, for all $i, j \in S$, $f(i) \neq f(j)$. There exists a construction of an (n, k, k)-perfect hash family of size $\mathcal{O}(e^k \cdot k^{\mathcal{O}(\log k)} \cdot \log n)$ and one can produce this family in time linear in the output size [50]. Using an (n, k, k)-perfect hash family of size $\mathcal{O}(e^k \cdot k^{\mathcal{O}(\log k)} \cdot \log n)$ rather than a random coloring, we get the desired deterministic algorithm. To see this it is enough to observe that if there is a subset $S \subseteq V(G)$ such that $G[S]$ contains F as a subgraph, then there exists a coloring $f \in \mathcal{H}$ such that the vertices of S are distinctly colored. So in our enumeration of colorings from $\mathcal{H}$ we will encounter the desired f. Hence for the given f, when we compute the value of col-inj(F, G) using Corollary 6, we know that col-inj$(F, G) > 0$. This concludes the proof. ∎

6.2. Improved randomized version of color coding. The first step of algorithms based on color coding is to color the vertices of $V(G)$ uniformly at random from the set $\{1, \ldots, k\}$. Then the probability that a copy of F in $V(G)$, if there is one, has become colorful is at least $k!/k^k > e^{-k}$. It is known that we can increase the probability of a copy of F being colorful in G by using more colors than k. Hüffner, Wernicke, and Zichner [40] have shown that the probability that a copy of F in $V(G)$, if there is one, has become colorful is at least $\mathcal{O}(1.752^{-k})$ if we randomly color the vertices of $V(G)$ from the set $\{1, \ldots, 1.3k\}$.

Theorem 16. *Let $c\colon V(G) \to \{1, 2, \ldots, l\}$ be a coloring of G, $k \leq l$ and $V_i = c^{-1}(i)$. Then*

$$\text{col-inj}(F, G) = \text{col-inj}(F, G^*) = \sum_{I \subseteq \{1,2,\ldots,l\}, |I| \leq k} (-1)^{k-|I|} \binom{l - |I|}{k - |I|} \text{hom}(F, G^*[\cup_{i \in I} V_i]).$$

Proof. We use the following formulation of Theorem 14 for our results. Let $c\colon V(G) \to \{1, 2, \ldots, k\}$ be a coloring of G and $V_i = c^{-1}(i)$; then

$$\text{(4)} \qquad \text{col-inj}(F, G) = \text{col-inj}(F, G^*) = \sum_{I \subseteq \{1,2,\ldots,k\}} (-1)^{k-|I|} \text{hom}(F, G^*[\cup_{i \in I} V_i]).$$

To prove our theorem, observe that

$$\begin{aligned}\text{col-inj}(F,G) &= \text{col-inj}(F,G^*)\\ &= \sum_{I'\subseteq\{1,\ldots,l\},|I'|=k} \text{col-inj}(F,G^*[\cup_{i\in I'}V_i])\\ &\overset{\text{by (4)}}{=} \sum_{I'\subseteq\{1,\ldots,l\},|I'|=k}\left(\sum_{I\subseteq I'}(-1)^{k-|I|}\hom(F,G^*[\cup_{i\in I}V_i])\right)\\ &= \sum_{I\subseteq\{1,\ldots,l\},|I|\le k}(-1)^{k-|I|}\binom{l-|I|}{k-|I|}\hom(F,G^*[\cup_{i\in I}V_i]).\end{aligned}$$

The last inequality follows from the fact that for any subset I, $|I|\le k$, the value of $\hom(F,G^*[\cup_{i\in I}V_i])$ is counted precisely for all those subsets I' for which $I\subseteq I'$ and $|I'|=k$. After fixing I we have $\{1,\ldots,l\}\setminus I$ elements left and hence every subset of size $k-|I|$ from $\{1,\ldots,l\}\setminus I$ gives the desired I. The number of such I is precisely $\binom{l-|I|}{k-|I|}$. Furthermore, for all such sets we have the same sign corresponding to I, that is, $(-1)^{k-|I|}$. This completes the proof. ∎

Using Theorem 16 we can obtain the following result.

COROLLARY 7. *Let F be a k-vertex graph of treewidth t. Then for any coloring $c\colon V(G)\to\{1,2,\ldots,l\}$ of an n-vertex graph G, the value of* col-inj(F,G) *is computable in time*

$$\mathcal{O}\left(k\cdot t\cdot n^{t+1}\cdot\sum_{i=0}^{k}\binom{l}{i}\right)$$

and space $\mathcal{O}(\log k\cdot n^{t+1})$.

THEOREM 17. *Let F be a k-vertex graph of treewidth t and let G be an n-vertex graph. A subgraph of G isomorphic to F (if one exists) can be found in $\mathcal{O}(4.32^k\cdot k\cdot t\cdot n^{t+1})$ expected time using $\mathcal{O}(\log k\cdot n^{t+1})$ space.*

Proof. We color the vertices of $V(G)$ uniformly at random from the set $\{1,\ldots,1.3k\}$. Then the probability that a copy of F in $V(G)$, if there is one, has been has become colorful is at least $\mathcal{O}(1.752^{-k})$ [40]. Given this random coloring we can compute the value of col-inj(F,G) in time $\mathcal{O}(2^{1.3k}kn^{t+1}\min\{k,t\})$ using Corollary 7. If col-inj$(F,G)>0$, then we know that there exists a subgraph of G isomorphic to F. Hence the expected running time to find a subgraph of G isomorphic to F (if one exists) is

$$\mathcal{O}(2^{1.3k}\cdot 1.752^k\cdot k\cdot t\cdot n^{t+1})=\mathcal{O}(4.32^k\cdot k\cdot t\cdot n^{t+1}).$$

This concludes the proof of the theorem. ∎

7. Conclusion and discussions.

In this paper we introduced an approach for counting subgraphs in a graph via counting graph homomorphisms in the realm of exact and parameterized algorithms. This approach yields various new algorithms for many basic problems, such as counting the number of perfect matchings, optimum bandwidth layouts, degree constrained spanning trees, maximum planar subgraphs, and others. On the other hand, it also unified several well-known results in exact algorithms, such as counting coloring and Hamiltonian cycles in general graphs and perfect matchings in bipartite graphs. Most of our results can be easily extended to weighted directed graphs. We believe that our method is generic and will find more applications.

The most important question which remains unanswered is, Can sub(F, G) be computed in $2^{\mathcal{O}(n)}$ time? In particular, we do not know the answer to this question even for the very special case, when the maximum degree of F is 3.

Recently, Wahlström [61] proved that if the cliquewidth of a graph F is at most c, then hom(F, G) can be computed in time $((2c+1)^{n_F} + 2^{cn}) \cdot n^{\mathcal{O}(1)}$, where n_F and n is the number of vertices in F and G, correspondingly. By the results of this paper, it implies that sub(F, G) can be computed in time $2^{\mathcal{O}(n)}$, when the clique-width of F is constant. It is interesting to note that all the natural classes of graphs $\mathcal{F}$ we know that have sub(F, G) computable in time $2^{\mathcal{O}(n)}$ for $F \in \mathcal{F}$ are either graphs of constant cliquewidth or of sublinear treewidth.

Acknowledgment. Many thanks to László Lovász for answering our questions on graph homomorphisms.

REFERENCES

[1] N. Alon, P. Seymour, and R. Thomas, *A separator theorem for nonplanar graphs*, J. Amer. Math. Soc., 3 (1990), pp. 801–808.

[2] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM, 42 (1995), pp. 844–856.

[3] S. Arnborg, D. G. Corneil, and A. Proskurowski , *Complexity of finding embeddings in a k-tree,* SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 277–284.

[4] L. Babai, *Moderately exponential bound for graph isomorphism*, in Fundamentals of Computational Theory, Lecture Notes in Comput. Sci. 117, Springer-Verlag, Berlin, 1981, pp. 34–50.

[5] L. Babai, W. M. Kantor, and E. M. Luks, *Computational complexity and the classification of finite simple groups*, in Proceedings of Foundations of Computer Science, 1983, pp. 162–171.

[6] E. T. Bax, *Algorithms to count paths and cycles*, Inform. Process. Lett., 52 (1994), pp. 249–252.

[7] R. Beals, R. Chang, W. I. Gasarch, and J. Torán, *On finding the number of graph automorphisms*, Chic. J. Theoret. Comput. Sci. (1999).

[8] T. Beyer and S. M. Hedetniemi, *Constant time generation of rooted trees*, SIAM J. Comput., 9 (1980), pp. 706–712.

[9] O. Bernardi, M. Noy, and D. Welsh, *Growth constants of minor-closed classes of graphs*, J. Combin. Theory Ser. B, 100 (2010), pp. 468–484.

[10] A. Björklund, *Determinant sums for undirected Hamiltonicity*, in Proceedings of Foundations of Computer Science, 2010, pp. 173–182.

[11] A. Björklund and T. Husfeldt, *Exact algorithms for exact satisfiability and number of perfect matchings*, Algorithmica, 52 (2008), pp. 226–249.

[12] A. Björklund and T. Husfeldt, *Inclusion-exclusion algorithms for counting set partitions*, in Proceedings of Foundations of Computer Science, 2006, pp. 575–582.

[13] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, *Fourier meets Möbius: Fast subset convolution*, in Proceedings of the Symposium on the Theory of Computing, 2007, pp. 67–74.

[14] A. Björklund, T. Husfeldt, and M. Koivisto, *Set partitioning via inclusion-exclusion*, SIAM J. Comput., 39 (2009), pp. 546–563.

[15] R. Blankenship and B. Oporowski, *Book embeddings of graphs and minor-closed classes*, in Proceedings of the 32nd Southeastern International Conference on Combinatorics, Graph Theory and Computing, 2001.

[16] H. L. Bodlaender, M. R. Fellows, and M. T. Hallett, *Beyond NP-completeness for problems of bounded width (extended abstract): Hardness for the W hierarchy*, in Proceedings of the Symposium on the Theory of Computing, 1994, pp. 449–458.

[17] N. Bonichon, C. Gavoille, N. Hanusse, D. Poulalhon, and G. Schaeffer, *Planar graphs, via well-orderly maps and trees*, Graphs Combin., 22 (2006), pp. 185–202.

[18] C. Borgs, J. Chayes, L. Lovász, V. T. Sós, and K. Vesztergombi, *Counting graph homomorphisms*, in Topics in Discrete Mathematics, Algorithms Combin. 26, Springer, Berlin, 2006, pp. 315–371.

[19] P. Z. Chinn, J. Chvatalova, A. K. Dewdney, and N. E. Gibbs, *The bandwidth problem for graphs and matrices—a survey*, J. Graph Theory, 6 (1982), pp. 223–254.

[20] M. Cygan and M. Pilipczuk, *Exact and approximate bandwidth*, Theoret. Comput. Sci., 411 (2010), pp. 3701–3713.

[21] V. Dalmau and P. Jonsson, *The complexity of counting homomorphisms seen from the other side*, Theoret. Comput. Sci., 329 (2004), pp. 315–323.

[22] J. Díaz, M. J. Serna, and D. M. Thilikos, *Counting H-colorings of partial k-trees*, Theoret. Comput. Sci., 281 (2002), pp. 291–309.

[23] F. Dorn *Planar subgraph isomorphism revisited*, in Proceedings of the Symposium on Theoretical Aspects of Computer Science, 2010, pp. 263–274.

[24] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, New York, 1999.

[25] J. Dunagan and S. Vempala, *On Euclidean embeddings and bandwidth minimization*, Proceedings of RANDOM-APPROX, 2001, pp. 229–240.

[26] M. Dyer and C. Greenhill, *The complexity of counting graph homomorphisms*, Random Structures Algorithms, 17 (2000), pp. 260–289.

[27] D. Eppstein, *Subgraph isomorphism in planar graphs and related problems*, J. Graph Algorithms Appl., 3 (1999), pp. 1–27.

[28] D. Eppstein, *Diameter and treewidth in minor-closed graph families*. Algorithmica, 27 (2000), pp. 275–291.

[29] U. Feige, *Coping with the NP-hardness of the graph bandwidth problem*, in Scandinavian Symposium and Workshops on Algorithm Theory, Lecture Notes in Comput. Sci. 1851, Springer-Verlag, Berlin, 2000, pp. 10–19.

[30] U. Feige, M. T. Hajiaghayi, and J. R. Lee, *Improved approximation algorithms for minimum-weight vertex separators*, SIAM J. Comput., 38 (2008), pp. 629–657.

[31] U. Feige and K. Talwar, *Approximating the bandwidth of caterpillars*, Algorithmica, 55 (2009), pp. 190–204.

[32] F. V. Fomin and D. Kratsch, *Exact Cxponential Algorithms*, Texts Theoret. Comput. Sci. EATCS Ser., Springer, Berlin, 2010.

[33] M. Fürer and B. Raghavachari, *Approximating the minimum-degree Steiner tree to within one of optimal*, J. Algorithms, 17 (1994), pp. 409–423.

[34] M. X. Goemans, *Minimum bounded degree spanning trees*, in Proceedings of Foundations of Computer Science, 2006, pp. 273–282.

[35] M. Grohe, *The complexity of homomorphism and constraint satisfaction problems seen from the other side*, J. ACM, 54 (2007).

[36] G. H. Hardy and S. Ramanujan, *Asymptotic formulae in combinatory analysis*, Proc. London Math. Soc., 17 (1918), pp. 75–115.

[37] P. Hell and J. Nešetřil, *On the complexity of H-coloring*, J. Combin. Theory Ser. B, 48 (1990), pp. 92–110.

[38] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*, Oxford Lecture Ser. Math. Appl. 28, Oxford University Press, Oxford, New York, 2004.

[39] J. E. Hopcroft and R. E. Tarjan, *Efficient planarity testing*, J. ACM, 21 (1974) pp. 549–568.

[40] F. Hüffner, S. Wernicke, and T. Zichner, *Algorithm engineering for color-coding with applications to signaling pathway detection*, Algorithmica, 52 (2008), pp. 114–132.

[41] R. M. Karp, *Dynamic programming meets the principle of inclusion and exclusion*, Oper. Res. Lett., 1 (1982), pp. 49–51.

[42] S. Khuller, R. Bhatia, and R. Pless, *On local search and placement of meters in networks*, SIAM J. Comput., 32 (2003), pp. 470–487.

[43] S. Kohn, A. Gottlieb, and M. Kohn, *A generating function approach to the traveling salesman problem*, in Proceedings of the Annual ACM Conference, 1977, pp. 294–300.

[44] M. Koivisto, *An $O(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion*, in Proceedings of Foundations of Computer Science, 2006, pp. 583–590.

[45] M. Koivisto, *Partitioning into sets of bounded cardinality*, in IWPEC 2009, Lecture Notes in Comput. Sci. 5917, Springer-Verlag, Berlin, 2009, pp. 258–263.

[46] J. R. Lee, *Volume distortion for subsets of Euclidean spaces*, Discrete Comput. Geom., 41 (2009), pp. 590–615.

[47] L. Lovász, *Operations with structures*, Acta Math. Hungar., 18 (1967), pp. 321–328.

[48] S. M. Malitz, *Genus g graphs have pagenumber $O(\sqrt{g})$*, J. Algorithms, 17 (1994), pp. 85–109.

[49] J. I. Munro and V. Raman, *Succinct representation of balanced parentheses and static trees*, SIAM J. Comput., 31 (2002), pp. 762–776.

[50] M. Naor, L. J. Schulman, and A. Srinivasan, *Splitters and near-optimal derandomization*, in Proceedings of Foundations of Computer Science, 1995, pp. 182–191.

[51] A. Nijenhuis and H. S. Wilf, *Combinatorial Algorithms*, Academic Press, New York, 1978.

[52] S. Norine, P. D. Seymour, R. Thomas, and P. Wollan, *Proper minor-closed families are small*, J. Combin. Theory, Ser. B, 96 (2006), pp. 754–757.

[53] R. Otter, *The number of trees*, Ann. Math., 49 (1948), pp. 583–599.
[54] N. Robertson and P. D. Seymour, *Graph minors* I-XXIII, J. Combin. Theory Ser. B, pp. 1984–2010.
[55] H. J. Ryser, *Combinatorial mathematics*, Carus Math. Monogr. 14, Mathematical Association of America, Washington, DC, 1963.
[56] J. B. Saxe, *Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time*, SIAM J. Algebraic Discrete Methods, 1 (1980), pp. 363–369.
[57] L. G. Valiant, *The complexity of computing the permanent*, Theoret. Comput. Sci., 8 (1979), pp. 189–201.
[58] V. Vassilevska, R. Williams, and S. L. M. Woo, *Confronting hardness using a hybrid approach*, in Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 1–10.
[59] V. Vassilevska and R. Williams, *Finding, minimizing, and counting weighted subgraphs*, in Proceedings of the Symposium on the Theory of Computing, 2009, pp. 455–464.
[60] J. R. Ullmann, *An algorithm for subgraph isomorphism*, J. ACM, 23 (1976), pp. 31–42.
[61] M. Wahlström, *New plain-exponential time classes for graph homomorphism*, Theory Comput. Syst., 49 (2011), pp. 273–282.