

Counting Subgraphs via Homomorphisms*

Omid Amini[†] Fedor V. Fomin[‡] Saket Saurabh[‡]

May 31, 2009

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, Gottlieb, Kohn, and Karp 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 $\mathcal{O}(2^{n+o(n)})$; 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: 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 non-isomorphic 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.

Contents

1	Introduction	2
1.1	Our results and related work	3
2	Preliminaries	5
3	Relating Counting Subgraphs to Counting Homomorphism	6

*An extended abstract of this paper has been presented at ICALP’09

[†]CNRS-DMA, École Normale Supérieure, Paris, France. omid.amini@m4x.org

[‡]Department of Infomatics, University of Bergen, Norway. [fedor.fomin|saket.saurabh}@ii.uib.no](mailto:{fedor.fomin|saket.saurabh}@ii.uib.no).

4	Classical Results	7
4.1	Counting Hamiltonian Cycles – Kohn-Gottlieb-Kohn-Karp Algorithm	7
4.2	Chromatic Polynomial – Björklund-Husfeldt-Koivisto Algorithm	7
4.3	Number of Perfect Matchings in Bipartite Graphs – Ryser’s Formula	9
5	New Applications	9
5.1	Set Saturating Homomorphisms and Ryser’s Formula	9
5.2	Subgraph Isomorphism when F has bounded Treewidth	10
5.3	Bandwidth	11
5.4	Degree Constrained Spanning Tree Problem	12
5.5	Counting Graphs Excluding a Fixed Minor	13
5.6	\mathcal{H} -Packing and some of its Variants	14
6	Color Coding	15
6.1	Deterministic Algorithm	15
6.2	Improved Randomized Version of Color-Coding	17
7	Conclusion and Discussions	18

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 . Many combinatorial structures in F , for example independent sets and proper vertex colorings, may be viewed as graph homomorphisms to a particular graph G , see the book of Hell and Nešetřil [30] 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.* [16] 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 [29]. Dyer and Greenhill [21] 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 graph with all loops present, a complete bipartite graph without loops, or a disjoint union of these graphs. Extending a result of Grohe [27], Dalmau and Jonsson [19] 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 subgraphs isomorphic to a given graph F in a general graph. 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 like

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 G isomorphic to a graph F to counting homomorphisms from F to G . Let $\text{sub}(F, G)$ denote the number of distinct 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 [38, 16] gave the following identities relating $\text{hom}(F, G)$ and $\text{inj}(F, G)$. For an equivalence relation Θ on $V(F)$, let F/Θ denote the graph obtained by identifying nodes that belong to the same equivalent class of Θ . Then

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

where

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

with the product running over all the equivalence classes of Θ and the sum running over all the equivalence relations. 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 like the graph containing n isolated vertices. We give an alternative formula which is helpful in counting “simple structures” in time vertex exponential in the size of the right hand graph G . Let us denote by $\text{aut}(F, F)$ the number of automorphisms, that is bijective homomorphisms, from F to itself. Our result shows that if $|V(F)| = |V(G)|$, then

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

This formula can be seen as a generalization of inclusion-exclusion based formulas which were used for many problems including counting the number of perfect matchings in a graph [10, 45], counting Hamiltonian cycles [6, 33, 35], and computing the chromatic polynomial of a graph [11, 36]. The basic idea of inclusion-exclusion based approach is that we express the number of objects we want to count as a sum of other objects which are easier to count. The main advantage of using graph homomorphisms is that despite of their expressive power, graph homomorphisms from many structures can be counted efficiently.

1.1 Our results and related work

We start by proving (1), 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, 33, 35]). 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 [11, 36]). 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 [45], see also Björklund and Husfeldt [11]).

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 of G is minimized. And this parameter, denoted by $bw(G)$, is called bandwidth of G . The best known approximation algorithm for this problem which is due to Krauthgamer *et al.* [37] provides $\mathcal{O}(\log^3 n)$ factor approximation. Saxe has shown that the bandwidth of a graph G , $bw(G)$, can be computed in time $\mathcal{O}(n^{bw(G)+1})$ [46]. When parameterized by bandwidth bw , BANDWIDTH is also known to be $W[t]$ -hard for all $t \geq 1$, in the realm of parameterized complexity [14].

Feige and Kilian [22] provided an exact algorithm computing the optimal bandwidth in time $10^n n^{\mathcal{O}(1)}$. Recently an improved algorithm with running time $5^n n^{\mathcal{O}(1)}$ was given by Cygan and Pilipczuk [18]. None of the known algorithms can be adapted to count the number of optimal bandwidth assignments. Feige and Talwar [24] showed that the bandwidth of a graph of treewidth at most t can be $(1 + \varepsilon)$ -approximated in time $2^{\mathcal{O}(\log n(t + \sqrt{\frac{t}{\varepsilon}}))}$. Vassilevska, Williams and Woo [48] 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 $2^{t \log_2 n + n^{\mathcal{O}(1)}}$ and space $2^{t \log_2 n} n^{\mathcal{O}(1)}$. This 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 [48].

Number of 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 [47]. For bipartite graphs, the best known exact algorithm for counting perfect matchings is to apply the Ryser's formula for the permanent [45], which runs in time $\mathcal{O}(1.414^n)$. Björklund and Husfeldt [10] showed how to compute the number of perfect matchings of a graph in time $2^n n^{\mathcal{O}(1)}$ and polynomial space. They also showed how to count perfect matchings in time $\mathcal{O}(1.732^n)$ 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 $k \geq \lfloor n/2 \rfloor$.

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 subgraph with specific properties in an n -vertex graph. For example, in time $2^{\mathcal{O}(n)}$, 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. The last result requires a new combinatorial bound on the number of non-isomorphic unlabeled M -minor-free graphs which implies as a corollary the main theorem of Norine, Seymour, Thomas, and Wollan from [42] on minor-closed families. This resolves an open problem of Bernardi, Noy and Welsh [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. Let us remark that prior to our work, the only known algorithm for many of the problems above was the trivial edge subset enumerating algorithm running in time $2^{\mathcal{O}(n^2)}$.

Packing problems: We also show how to solve in time $2^{\mathcal{O}(n)}$ some a priori more difficult problems, like finding a maximum vertex disjoint packing from a class \mathcal{H} , where \mathcal{H} is a graph class 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 algorithm were known before.

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], Alon *et al.* 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 *et al.* [32] that was used to speed-up the above mentioned algorithm of Alon *et al.* 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 remaining of the paper is organized as follows. In Section 2 and Section 3, we provide necessary definitions and some preliminary results. In Section 4, we show that several classical results from the area of exact algorithms can be obtained by making use of graph homomorphisms. Section 5 provides new applications of our approach that we briefly mentioned above. In Section 6, we revisit the Color Coding approach of Alon, Yuster, and Zwick.

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 $G[W]$. For a given vertex $v \in V(G)$ and a subset $W \subseteq V(G)$, we denote by $\text{deg}_W(v)$ 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 $(v, w) \in E(G)$, 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 $\text{tw}(G)$.

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 \mathcal{G} is *minor closed* if any minor of any graph in \mathcal{G} is also an element of \mathcal{G} . A minor closed graph class \mathcal{G} is *M -minor-free* or simply *M -free* if $M \notin \mathcal{G}$.

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 $f : V(F) \rightarrow V(G)$, 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 $\text{hom}(F, G)$, $\text{inj}(F, G)$ and $\text{sub}(F, G)$ denote the number of

homomorphisms from F to G , the number of injective homomorphisms from F to G and the number of distinct copies of F in G , respectively. We need the following result relating $\text{sub}(F, G)$ and $\text{inj}(F, G)$. This result is a folklore and we omit its proof here.

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

This proposition allows us to focus on computing the value of $\text{inj}(F, G)$, as one can compute $\text{aut}(F, F)$ 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 .

3 Relating Counting Subgraphs to Counting Homomorphism

We first give a formula expressing the number of injective homomorphisms from F to G in terms of the number of graphs 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*

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

Proof. 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, in the right hand side sum we also count maps which are not injective. 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 $\tilde{V} = \text{im}(h)$ be the image of h in $V(G)$ and $X = V(G) \setminus \tilde{V} \neq \emptyset$. We now observe that h is counted only when we are counting homomorphisms from $V(F)$ to $G[V(G) \setminus W]$ such that $W \subseteq X$. The total contribution of h in the sum, taking into account the signs, is

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

The last part of the theorem can be proved using similar arguments. And the theorem follows. \square

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 $|F| \leq |G| = n$ and $W \subseteq V(G)$. Then as a consequence of Theorem 1, we can compute the value of $\text{inj}(F, G)$ 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 $\text{inj}(F, G[V'])$. But this will take time $\mathcal{O}\left(\binom{n}{n_F} 2^{n_F} t(n)\right)$, which in the worst case, could be equal to $\mathcal{O}(3^n \cdot t(n))$. In the remaining 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 $|V(F)| = n_F \leq |V(G)| = n$. Then*

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

¹In fact, here for a given graph F , they solve a harder problem of computing its automorphism group and its generators. Please refer to Section 7 of [7] for further discussion.

Proof. By Theorem 1,

$$\text{inj}(F, G) = \sum_{W \subseteq V(G)} (-1)^{|V|-|W|} \text{hom}(F, G[W]). \quad (2)$$

Now

$$\begin{aligned} \text{inj}(F, G) &= \sum_{W \subseteq V(G), |W|=n_F} \text{inj}(F, G[W]) \\ &\stackrel{\text{by (2)}}{=} \sum_{W \subseteq V(G), |W|=n_F} \left(\sum_{Y \subseteq W} (-1)^{|W|-|Y|} \text{hom}(F, G[Y]) \right) \\ &= \sum_{W \subseteq V(G), |W|=n_F} \left(\sum_{Y \subseteq W} (-1)^{n_F-|Y|} \text{hom}(F, G[Y]) \right) \\ &= \sum_{Y \subseteq V(G), |Y| \leq n_F} (-1)^{n_F-|Y|} \binom{n-|Y|}{n_F-|Y|} \text{hom}(F, G[Y]). \end{aligned}$$

The last equality follows from the fact that for any subset Y with $|Y| \leq n_F$, the value of $\text{hom}(F, G[Y])$ 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 , $\text{hom}(F, G[Y])$ 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. \square

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 $\#\text{HAM}(G)$ denote the number of Hamiltonian cycles in a graph G and let $F = C_n$ be the cycle of length n , then $\text{sub}(F, G) = \#\text{HAM}(G)$. It is very easy to see that $\text{aut}(C_n, C_n) = 2n$, and for any graph H , $\text{hom}(C_n, H) = \text{tr}(A_H^n) = \sum_{i=1}^n \lambda_i^n$, where A_H is the adjacency matrix of H and $\lambda_1, \dots, \lambda_n$ are its eigenvalues (see for example [16]). Using these results and Theorem 1, we can compute $\#\text{HAM}(G)$ in time $2^n n^{\mathcal{O}(1)}$ and polynomial space.

4.2 Chromatic Polynomial – Björklund-Husfeldt-Koivisto Algorithm

A *proper k -coloring* of a graph G is a function $f : V(G) \rightarrow \{1, \dots, k\}$ 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 $k \geq 1$, $\chi(G; k)$ is the number of proper k -colorings of G . Which is also $\chi(G; k) = \text{hom}(G, K_k)$, where K_k is a complete graph of size k . (Remark that the chromatic number of G is the smallest integer $k > 0$ for which $\chi(G; k) > 0$.) However, to compute the chromatic polynomial of a graph, we have to look at homomorphisms from “the other side”.

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 $F = \cup_{i=1}^k K_{t_i}$ in \overline{G} . Thus as far as we know the correct sizes of cliques, graph coloring of 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 $F(\zeta) = \cup_i K_{t_i}$. Then

$$\chi(G; k) = \sum_{\zeta \in \mathcal{P}_k(n)} k! \cdot \text{sub}(F(\zeta), \overline{G}). \quad (3)$$

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 in [28]:

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

Furthermore one can give a polynomial delay enumeration algorithm for partitions of n into k parts [41]. 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 (3). To compute (3), we have to know the value of $\text{sub}(F(\zeta), \overline{G})$, and to compute this value we use Theorem 1. To implement Theorem 1, we have to compute the values of $\text{aut}(F(\zeta))$, and $\text{hom}(F(\zeta), \overline{G}[V(G) \setminus W])$, where $W \subseteq V(G)$. The computation of $\text{aut}(F(\zeta))$ 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 either maps a component (complete graph) 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 , $p \leq k$, be the sizes of the components in $F(\zeta)$. Let us 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 $\text{aut}(F(\zeta)) = \prod_{x \in \{x_1, x_2, \dots, x_p\}} n(x)!x!$, and this value is computable in polynomial time for each ζ .

To compute $\text{hom}(F(\zeta), \overline{G}[V(G) \setminus W])$, 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. [16].

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

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

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 we have mentioned it already, 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 $\zeta = (t_1, t_2, \dots, t_k) \in \mathcal{P}_k(n)$, and compute the inner sum in (3). As far as the table \mathcal{T} is computed, the computations of $\text{hom}(F(\zeta), \overline{G}[V \setminus W])$ in (3) for every $W \subseteq V$, takes polynomial time. Thus for every partition ζ , it takes time $2^n \cdot n^{\mathcal{O}(1)}$ to compute $\text{sub}(F(\zeta), \overline{G})$. 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 the Ryser’s Formula [45] says that

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

where $\# \text{PM}(G)$ is the number of perfect matchings in G . The sum $\sum_{v \notin X} 1_{[uv \in E(G)]}$ counts the number of u ’s neighbors 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 $\# \text{PM}(G) = \text{sub}(F, G)$. By using Theorem 1, it is easy to obtain an algorithm to compute the value of $\# \text{PM}(G)$ in time $2^n n^{\mathcal{O}(1)}$. We will use the notion of saturating homomorphism in Section 5.1 to compute $\# \text{PM}(G)$ in faster time, and in particular in time $\mathcal{O}(2^{n/2} n^2)$ in 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 - $\text{hom}(F, G)$ we denote the number of S -saturating homomorphisms. Observe that for $S = \emptyset$ an S -saturating homomorphism is simply a homomorphism. Along the lines of Theorem 1, we can prove the following theorem whose proof is omitted.

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

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

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 $\text{sub}(F, G) = \#\text{PM}(G)$. By Theorem 3, we have that

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

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

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

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 $\{f_1, f_2, \dots, f_a\}$ 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 other edge in two ways. The remaining $\frac{n}{2} - a$ edges are mapped to edges in $G[V(G) \setminus (W \cup S)]$. Proposition 2 combined with the fact that an edge can be mapped to other edge in two ways give the factor of $|2 \cdot E(G[V(G) \setminus (W \cup S)])|^{\frac{n}{2}-a}$ in the formula. Furthermore, $\text{aut}(F, F)$ is equal to $2^{n/2}(n/2)!$. \square

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. Then by Theorem 4, we obtain the following result.

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

5.2 Subgraph Isomorphism when F has bounded Treewidth

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 $\text{hom}(F, G)$, when F is a graph of bounded treewidth.

Proposition 3 ([20]). *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 $\text{hom}(F, G)$ 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 $n_F \leq n$ vertices respectively, given together with a tree-decomposition of width t of F . Then $\text{sub}(F, G)$ is computable in time*

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

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

Proof. Observe that $\text{aut}(F, F) = \text{inj}(F, F)$. Hence, using Theorem 1 together with Proposition 3 we can compute $\text{aut}(F, F)$ in time $\mathcal{O}(2^{n_F} \cdot n_F^{t+2} \cdot t)$ and space $\mathcal{O}(\log n_F \cdot n_F^{t+1})$. Now we use Theorem 2 and Proposition 3 to compute the value of $\text{inj}(F, G)$ in time

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

and space $\mathcal{O}(\log n_F \cdot n^{t+1})$. By Proposition 1, we know that $\text{sub}(F, G) = \text{inj}(F, G) / \text{aut}(F, F)$ which allows us to conclude the proof of the theorem. \square

5.3 Bandwidth

BANDWIDTH is one of the well studied graph layout problems. A *layout* of a graph G on n vertices is a map $f : V(G) \rightarrow \{1, \dots, n\}$. 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 r^{th} 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. [17].

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 $|V(M)|^{3/2} \sqrt{n}$. 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 $2^{n+O(\sqrt{n})}$.*

Theorem 6 can be used to improve a hybrid algorithm given in [48], 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. *BANDWIDTH 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 $4^n \cdot n^{O(1)}$ time, or an $\mathcal{O}(\log^{3/2} n)$ -approximation in polynomial time.*

Proof. Feige, Hajiaghayi and Lee [23] 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 $4^n \cdot n^{O(1)}$. \square

5.4 Degree Constrained Spanning Tree Problem

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 constraints on the vertices. More precisely, the DEGREE CONSTRAINED SPANNING TREE (DCST) problem is defined as follows: Given a connected undirected graph G and a vector of size n , $\hat{a} = (a_1, a_2, \dots, a_n)$, find a spanning tree T of G (if one exists), such that there is a bijective mapping $g : V(G) \rightarrow \{a_1, a_2, \dots, a_n\}$ with the property that $\deg_T(v) = g(v)$. A variation of DCST called MODIFIED DEGREE CONSTRAINED SPANNING TREE (MDCST) is defined by replacing the condition of $\deg_T(v) = g(v)$ with $\deg_T(v) \leq g(v)$ in DCST.

HAMILTONIAN PATH is an instance of DCST problem with the vector $(1, 2, \dots, 2, 1)$. Other well known problems which can be formulated as an instance of either DCST or MDCST include FULL DEGREE SPANNING TREE (a spanning tree which maximizes the number of vertices having the same degree in the graph and the tree) [34] and MINIMUM DEGREE SPANNING TREE (a spanning tree for which the maximum degree is minimized) [25, 26] problems. To solve DCST and MDCST problems we need the following classical result of Otter from 1948 [43].

Proposition 5. *The number of unlabelled trees on n vertices $\mathcal{T}(n) \sim C\alpha^n n^{-5/2}$ as $n \rightarrow \infty$, where $C = 0.53495\dots$ and $\alpha = 2.95576\dots$*

Moreover, by the result of Beyer and Hedetniemi [8], it is possible to enumerate all non-isomorphic (unlabelled) trees in time $\mathcal{T}(n)n^{\mathcal{O}(1)}$.

Theorem 7. *Let G be a graph on n vertices and $\hat{a} = (a_1, \dots, a_n)$ be a vector of length n . Then we can count the number of feasible solutions to DCST and MDCST in time $\mathcal{O}(5.912^n)$.*

Proof. We start with an algorithm to find a feasible solution to DCST (or MDCST) problem on a graph G on n vertices.

Step 1: Enumerate all 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 of subgraphs of G isomorphic to T .

Step 4: Output the maximum value of $\text{sub}(T, G)$ taken over all trees T .

The first step of the algorithm is done using the result of Beyer and Hedetniemi [8], which gives an algorithm to enumerate all non-isomorphic (unlabelled) 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 or not. 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 $\mathcal{O}(2.9558^n \cdot 2^n n^3) = \mathcal{O}(5.912^n)$. With some book keeping we can count the number of feasible solutions to DCST or MDCST in the same time. This immediately gives us the theorem. \square

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

Corollary 4. *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 MAXIMUM PLANAR SUBGRAPH and 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 (i.e. $c^n n^{\mathcal{O}(1)}$) 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, Seymour, Thomas, and Wollan [42] proved that the number of labelled 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 unlabelled 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 [42].

Theorem 8. *Let \mathcal{G} be a family of unlabelled 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 non-isomorphic graphs in \mathcal{G} is at most c_M^n . Moreover, the elements of \mathcal{G} can be enumerated in time $\mathcal{O}(c_M^n)$.*

Proof. We prove the theorem by using results about geometric representation of minor-free graphs. *Book embedding* is a generalization of planar embedding to non planar 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 vertices of the graph are embedded on the spine, and the edges are distributed on the faces, 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 [39] 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 [13].

Proposition 6. [13] *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 .*

We prove the following claim by induction on the number of pages. The number of unlabelled n -vertex graphs that can be embedded in p pages is at most 4^{pn} . 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 the number of graphs with page-number at most p times the amount of graphs that can be embedded into one page. In each page the edges embedded on the page form a partial parenthesis system. The total number of complete parenthesis systems of size n is given by the Catalan number of order n which is at most 4^n . So the total number of graphs embeddable in one page is at most 4^n . By induction assumption, the number of graphs embeddable in p pages is at most 4^{pn} and the claim follows.

By Proposition 6, all graphs from \mathcal{G} have page-number bounded by the constant h , thus the total number of unlabeled graphs of the class \mathcal{G} on n vertices is bounded by $4^{hn} = c_M^n$.

Finally, let us remark that the algorithm, enumerating all graphs from \mathcal{G} 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 and for each graph check if it contains M as a minor. By the seminal work of Robertson-Seymour [44], such a test can be done in polynomial time. \square

For simpler classes of graphs, like planar, 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 checking if the input graph is planar [31]. Also to bound the number of non-isomorphic planar graphs, we can use the following result from the theory of succinct data structures.

Proposition 7 ([15]). *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.*

As far as we have Theorem 8 and Proposition 7, by making use of Theorem 5 and the result of Alon et al [1] that the treewidth of an n -vertex graph excluding a fixed graph as a minor is $\mathcal{O}(\sqrt{n})$, we conclude with the main result of this section.

Theorem 9. *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

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

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 $H \in \mathcal{H}$ contains 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 10. *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 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) , $0 \leq l \leq p \leq n$, 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 $\mathcal{H}_{p_1} \times \mathcal{H}_{p_2} \times \dots \times \mathcal{H}_{p_l}$, 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 non zero.

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 4. Step (iii) of the algorithm depends of the size of \mathcal{H}_p . Hence if the number of graphs on x vertices in \mathcal{H} is bounded by d^x , for d a constant, then $|\mathcal{H}_p| \leq d^p$. Since \mathcal{H} is a graph class excluding a fixed graph M as a minor, by Theorem 8 there exists a constant c_M such that $|\mathcal{H}_x| \leq c_M^x$ for all $x \in \mathbb{N}$. We enumerate all M -minor free graphs by making use of Theorem 8, 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. The Step (iv) of the algorithm can be done in time $2^{n+\mathcal{O}(\sqrt{n})}$ using Theorem 5. Choosing $c = 2c_M$ completes the theorem. \square

In what follows we give a few corollaries of Theorem 10 when \mathcal{H} is some specific graph class. Let $\mathcal{H}^c = \{C_q \mid \text{simple cycle of length } q, q \in \mathbb{N}, q \geq 3\}$. It is easy to see that for a simple undirected graph G , the value of $\text{pack}_{\mathcal{H}^c}(G)$ 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 $p_i \geq 3$ we have $|\mathcal{H}_{\zeta}^c| = 1$. 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 maximum number of vertex disjoint cycles or finding the maximum number of odd cycles are similarly defined. This brings us to the following corollary.

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

6 Color Coding

In this section we show how the ideas of counting homomorphisms and inclusion-exclusion combined with Color Coding technique of Alon, Yster, and Zwick provide polynomial space parameterized algorithms.

6.1 Deterministic Algorithm

Let $c: V(G) \rightarrow \{1, 2, \dots, k\}$ 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 $\text{col-inj}(F, G)/\text{aut}(F, F)$. Let G^* be the graph obtained from G by

deleting the mono-chromatic 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 $c: V(G) \rightarrow \{1, 2, \dots, k\}$ 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 11. *Let $c: V(G) \rightarrow \{1, 2, \dots, k\}$ be a coloring of G and $V_i = c^{-1}(i)$. Then*

$$\begin{aligned} \text{col-inj}(F, G) = \text{col-inj}(F, G^*) &= \sum_{I \subseteq \{1, 2, \dots, k\}} (-1)^{|I|} \text{hom}(F, G^*[V(G^*) \setminus \cup_{i \in I} V_i]) \\ &= \sum_{I \subseteq \{1, 2, \dots, k\}} (-1)^{k-|I|} \text{hom}(F, G^*[\cup_{i \in I} V_i]). \end{aligned}$$

Proof. 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 $X = \{1, \dots, k\} \setminus \chi(h(F))$ is non-empty. We now observe that h is counted only when we are counting homomorphisms from $V(F)$ to $G^*[V(G^*) \setminus \cup_{i \in I'} V_i]$ 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. \square

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 11 yields the following result.

Corollary 6. *Let F be a k -vertex graph of treewidth t . Then for any coloring $c: V(G) \rightarrow \{1, 2, \dots, k\}$ of an n -vertex graph G , the value of $\text{col-inj}(F, G)$ 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 12. *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 Alon *et al.* [2]. We color the vertices of $V(G)$ uniformly at random from the set $\{1, \dots, k\}$. Then the probability that a copy of F in $V(G)$, if there is one, has been assigned different colors or has become colorful is at least $k!/k^k > e^{-k}$. Given this random coloring we can compute the value of $\text{col-inj}(F, G)$ in

time $\mathcal{O}(2^k k n^{t+1} \min\{k, t\})$ using Corollary 6. If $\text{col-inj}(F, G) > 0$, 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 $\{1, \dots, k\}$ to a deterministic one. This is done by making use of (n, k, k) -perfect hash family. A (n, k, k) -perfect hash family, \mathcal{H} , is a set of functions from $\{1, \dots, n\}$ to $\{1, \dots, k\}$ such that for every subset $S \subseteq \{1, \dots, n\}$ 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 (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 [40]. Using (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 $\text{col-inj}(F, G)$ using Corollary 6, we know that $\text{col-inj}(F, G) > 0$. This concludes the proof. \square

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 $\{1, \dots, k\}$. Then the probability that a copy of F in $V(G)$, if there is one, has been assigned different colors or 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 *et al.* [32] 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 $\{1, \dots, 1.3k\}$.

Theorem 13. *Let $c: V(G) \rightarrow \{1, 2, \dots, l\}$ be a coloring of G , $k \leq l$ and $V_i = c^{-1}(i)$. Then*

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

Proof. We use the following formulation of Theorem 11 for our results: Let $c: V(G) \rightarrow \{1, 2, \dots, k\}$ be a coloring of G and $V_i = c^{-1}(i)$, then

$$\text{col-inj}(F, G) = \text{col-inj}(F, G^*) = \sum_{I \subseteq \{1, 2, \dots, k\}} (-1)^{k-|I|} \text{hom}(F, G^*[\cup_{i \in I} V_i]). \quad (5)$$

To prove our theorem, observe that:

$$\begin{aligned} \text{col-inj}(F, G) &= \text{col-inj}(F, G^*) \\ &= \sum_{I' \subseteq \{1, \dots, l\}, |I'|=k} \text{col-inj}(F, G^*[\cup_{i \in I'} V_i]) \\ &\stackrel{\text{by (5)}}{=} \sum_{I' \subseteq \{1, \dots, l\}, |I'|=k} \left(\sum_{I \subseteq I'} (-1)^{k-|I|} \text{hom}(F, G^*[\cup_{i \in I} V_i]) \right) \\ &= \sum_{I \subseteq \{1, \dots, l\}, |I| \leq k} (-1)^{k-|I|} \binom{l-|I|}{k-|I|} \text{hom}(F, G^*[\cup_{i \in I} V_i]). \end{aligned}$$

The last inequality follows from the fact that for any subset I , $|I| \leq k$, the value of $\text{hom}(F, G^*[\cup_{i \in I} V_i])$ is counted precisely for all those subsets I' for which $I \subseteq I'$ and $|I'| = k$. After fixing I we have

$\{1, \dots, l\} \setminus I$ elements left and hence every subset of size $k - |I|$ from $\{1, \dots, l\} \setminus I$ 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. \square

Using Theorem 13 we can obtain the following result.

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

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

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

Theorem 14. *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 $\{1, \dots, 1.3k\}$. Then the probability that a copy of F in $V(G)$, if there is one, has been assigned different colors or has become colorful is at least $\mathcal{O}(1.752^{-k})$ [32]. Given this random coloring we can compute the value of $\text{col-inj}(F, G)$ in time $\mathcal{O}(2^{1.3k} k n^{t+1} \min\{k, t\})$ using Corollary 7. If $\text{col-inj}(F, G) > 0$, 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. \square

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 like counting the number of perfect matchings, optimum bandwidth layouts, degree constrained spanning trees, maximum planar subgraphs beside others. On the other hand it also unified several well-known results in exact algorithms including counting coloring, Hamiltonian cycles and perfect matchings of bipartite graphs. These alternate algorithms generalize and unify algorithms for many well known problems. Let us remark that most of our results can be easily extended to weighted directed graphs. We believe that our method is generic and will find many more applications.

The most important question which remains unanswered is: Can $\text{sub}(F, G)$ 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 [49] proved that if the clique-width of a graph F is at most c , then $\text{hom}(F, G)$ 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 $\text{sub}(F, G)$ 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 $\text{sub}(F, G)$ computable in time $2^{\mathcal{O}(n)}$ for $F \in \mathcal{F}$, are either graphs of constant clique-width or of sublinear treewidth.

Acknowledgement 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), 277–284.
- [4] L. BABAI, *Moderately exponential bound for graph isomorphism*, in FCT, vol. 117 of LNCS, 1981, pp. 34–50.
- [5] L. BABAI, W. M. KANTOR, AND E. M. LUKS, *Computational complexity and the classification of finite simple groups*, in FOCS, 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*, Chicago J. Theor. Comput. Sci., 1999 (1999).
- [8] T. BEYER AND S. HEDETNIEMI, *Constant time generation of rooted trees*, SIAM J. Comp., 9 (1980), pp. 706–712.
- [9] O. BERNARDI, M. NOY, AND D. WELSH, *On the growth rate of minor-closed classes of graphs*, <http://arxiv.org/abs/0710.2995>.
- [10] A. BJÖRKLUND AND T. HUSFELDT, *Exact algorithms for exact satisfiability and number of perfect matchings*, Algorithmica, 52(2) (2008), pp. 226–249.
- [11] ———, *Inclusion-exclusion algorithms for counting set partitions*, in FOCS, 2006, pp. 575–582.
- [12] A. BJÖRKLUND, T. HUSFELDT, P. KASKI, AND M. KOIVISTO, *Fourier meets möbius: fast subset convolution*, in STOC, 2007, pp. 67–74.
- [13] 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.
- [14] 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 STOC, 1994, pp. 449–458.
- [15] N. BONICHON, C. GAVOILLE, N. HANUSSE, D. POULALHON, AND G. SCHAEFFER, *Planar graphs, via well-orderly maps and trees*, Graphs and Combinatorics, 22 (2006), pp. 185–202.
- [16] C. BORGS, J. CHAYES, L. LOVÁSZ, V. T. SÓS, AND K. VESZTERGOMBI, *Counting graph homomorphisms*, in Topics in discrete mathematics, vol. 26 of Algorithms Combin., Springer, Berlin, 2006, pp. 315–371.
- [17] 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.

- [18] M. CYGAN AND M. PILIPCZUK, *Faster exact bandwidth*, in WG, vol. 5344 of LNCS, 2008, pp. 101–109.
- [19] V. DALMAU AND P. JONSSON, *The complexity of counting homomorphisms seen from the other side*, Theoret. Comput. Sci., 329 (2004), pp. 315–323.
- [20] J. DÍAZ, M. J. SERNA, AND D. M. THILIKOS, *Counting h -colorings of partial k -trees*, Theor. Comput. Sci., 281 (2002), pp. 291–309.
- [21] M. DYER AND C. GREENHILL, *The complexity of counting graph homomorphisms*, Random Structures Algorithms, 17 (2000), pp. 260–289.
- [22] U. FEIGE, *Coping with the NP-hardness of the graph bandwidth problem*, in SWAT, vol. 1851 of LNCS, 2000, pp. 10–19.
- [23] U. FEIGE, M. T. HAJIAGHAYI, AND J. R. LEE, *Improved approximation algorithms for minimum-weight vertex separators*, in STOC, 2005, pp. 563–572.
- [24] U. FEIGE AND K. TALWAR, *Approximating the bandwidth of caterpillars*, in APPROX-RANDOM, vol. 3624 of LNCS, 2005, pp. 62–73.
- [25] M. FÜRER AND B. RAGHAVACHARI, *Approximating the minimum-degree steiner tree to within one of optimal*, J. Algorithms, 17 (1994), pp. 409–423.
- [26] M. X. GOEMANS, *Minimum bounded degree spanning trees*, in FOCS, 2006, pp. 273–282.
- [27] M. GROHE, *The complexity of homomorphism and constraint satisfaction problems seen from the other side*, J. ACM, 54 (2007), pp. Art. 1, 24 pp. (electronic).
- [28] G. H. HARDY AND S. RAMANUJAN, *Asymptotic formulae in combinatory analysis*, Proc. London Math. Soc., 17 (1918), pp. 75–115.
- [29] P. HELL AND J. NEŠETŘIL, *On the complexity of H -coloring*, J. Combin. Theory Ser. B, 48 (1990), pp. 92–110.
- [30] ———, *Graphs and homomorphisms*, vol. 28 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2004.
- [31] J. E. HOPCROFT AND R. E. TARJAN, *Efficient planarity testing*, J. ACM, 21 (1974) pp. 549–568.
- [32] F. HFFNER, S. WERNICKE, AND T. ZICHER, *Algorithm engineering for color-coding with applications to signaling pathway detection*, Algorithmica, 52(2) (2008), pp. 114132.
- [33] R. M. KARP, *Dynamic programming meets the principle of inclusion and exclusion*, Oper. Res. Lett., 1 (1982), pp. 49–51.
- [34] S. KHULLER, R. BHATIA, AND R. PLESS, *On local search and placement of meters in networks*, SIAM J. Comp., 32 (2003), pp. 470–487.
- [35] 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.
- [36] M. KOIVISTO, *An $O(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion*, in FOCS, 2006, pp. 583–590.

- [37] R. KRAUTHGAMER, J. R. LEE, M. MENDEL, AND A. NAOR, *Measured descent: A new embedding method for finite metrics*, in FOCS, 2004, pp. 434–443.
- [38] L. LOVÁSZ, *Operations with structures*, Acta Math. Hung., 18 (1967), pp. 321–328.
- [39] S. M. MALITZ, *Genus g graphs have pagenumber $O(\sqrt{g})$* , J. Algorithms, 17 (1994), pp. 85–109.
- [40] M. NAOR, L. J. SCHULMAN, AND A. SRINIVASAN, *Splitters and near-optimal derandomization*, in FOCS, 1995, pp. 182–191.
- [41] A. NIJENHUIS AND H. S. WILF, *Combinatorial Algorithms*, Academic Press, Inc., 1978.
- [42] S. NORINE, P. D. SEYMOUR, R. THOMAS, AND P. WOLLAN, *Proper minor-closed families are small*, J. Comb. Theory, Ser. B, 96 (2006), pp. 754–757.
- [43] R. OTTER, *The number of trees*, Ann. Math., 49 (1948), pp. 583–599.
- [44] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors I-XX*, appearing in J. Combin. Theory Ser. B since 1984.
- [45] H. J. RYSER, *Combinatorial mathematics*, The Carus Mathematical Monographs, No. 14, Published by The Mathematical Association of America, 1963.
- [46] J. B. SAXE, *Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time*, SIAM J. Algebraic Discrete Methods, 1 (1980), pp. 363–369.
- [47] L. G. VALIANT, *The complexity of computing the permanent*, Theoret. Comput. Sci., 8 (1979), pp. 189–201.
- [48] V. VASSILEVSKA, R. WILLIAMS, AND S. L. M. WOO, *Confronting hardness using a hybrid approach*, in SODA, 2006, pp. 1–10.
- [49] M. WAHLSTRÖM, *New Plain-Exponential Time Classes for Graph Homomorphism*, in CSR, 2009, to appear.