Journal of Computer and System Sciences 78 (2012) 698–706

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Faster algorithms for finding and counting subgraphs

Fedor V. Fomin [a,*,1], Daniel Lokshtanov [a], Venkatesh Raman [b], Saket Saurabh [b], B.V. Raghavendra Rao [c]

[a] Department of Informatics, University of Bergen, Bergen, Norway
[b] The Institute of Mathematical Sciences, Chennai, India
[c] Department of Computer Science, Universitaet des Saarlandes, Germany

ARTICLE INFO

Article history:
Received 23 July 2010
Received in revised form 26 September 2011
Accepted 10 October 2011
Available online 20 October 2011

Keywords:
Parameterized complexity
Subgraph Isomorphism
Homomorphism
Counting
Treewidth

ABSTRACT

In the Subgraph Isomorphism problem we are given two graphs F and G on k and n vertices respectively as an input, and the question is whether there exists a subgraph of G isomorphic to F. We show that if the treewidth of F is at most t, then there is a randomized algorithm for the Subgraph Isomorphism problem running in time $\mathcal{O}^*(2^k n^{2t})$. Our proof is based on a novel construction of an arithmetic circuit of size at most $n^{\mathcal{O}(t)}$ for a new multivariate polynomial, Homomorphism Polynomial, of degree at most k, which in turn is used to solve the Subgraph Isomorphism problem. For the counting version of the Subgraph Isomorphism problem, where the objective is to count the number of distinct subgraphs of G that are isomorphic to F, we give a deterministic algorithm running in time and space $\mathcal{O}^*(\binom{n}{k/2} n^{2p})$ or $\binom{n}{k/2} n^{\mathcal{O}(t \log k)}$. We also give an algorithm running in time $\mathcal{O}^*(2^k \binom{n}{k/2} n^{5p})$ and taking $\mathcal{O}^*(n^p)$ space. Here p and t denote the pathwidth and the treewidth of F, respectively. Our work improves on the previous results on Subgraph Isomorphism, it also extends and unifies most of the known results on sub-path and subtree isomorphisms.

1. Introduction

In this paper we consider the classical problem of finding and counting a fixed pattern graph F on k vertices in an n-vertex host graph G, when we restrict the treewidth of the pattern graph F by t. More precisely the problems we consider are the Subgraph Isomorphism problem and the #Subgraph Isomorphism problem. In the Subgraph Isomorphism problem we are given two graphs F and G on k and n vertices respectively as an input, and the question is whether there exists a subgraph in G which is isomorphic to F? In the #Subgraph Isomorphism problem the objective is to count the number of distinct subgraphs of G that are isomorphic to F. Recently #Subgraph Isomorphism, in particular when F has bounded treewidth, has found applications in the study of biomolecular networks. We refer to Alon et al. [1] and references therein for further details.

In a seminal paper Alon et al. [3] introduced the method of Color-Coding for the Subgraph Isomorphism problem, when the treewidth of the pattern graph is bounded by t and obtained randomized as well as deterministic algorithms running in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(t)}$. This algorithm was derandomized using k-perfect hash families. In particular, Alon et al. [3] gave a randomized $\mathcal{O}^*(5.4^k)$[2] time algorithm and a deterministic $\mathcal{O}^*(c^k)$ time algorithm, where c is a large constant, for

* Corresponding author.
E-mail addresses: fomin@ii.uib.no (F.V. Fomin), daniello@ii.uib.no (D. Lokshtanov), vraman@imsc.res.in (V. Raman), saket@imsc.res.in (S. Saurabh), bvrr@cs.uni-sb.de (B.V.R. Rao).
1 Supported by the European Research Council advanced grant.
2 We use $\mathcal{O}^*()$ notation that hides factors polynomial in n and the parameter k.

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2011.10.001

the k-Path problem, a special case of Subgraph Isomorphism where F is a path of length k. There have been a lot of efforts in parameterized algorithms to reduce the base of the exponent of both deterministic as well as the randomized algorithms for the k-Path problem. In the first of such attempts, Chen et al. [10] and Kneis et al. [17] independently discovered the method of Divide and Color and gave a randomized algorithm for k-Path running in time $\mathcal{O}^*(4^k)$. Chen et al. [10] also gave a deterministic algorithm running in time $\mathcal{O}^*(4^{k+o(k)})$ using an application of universal sets. While the best known deterministic algorithm for k-Path problem still runs in time $\mathcal{O}^*(4^{k+o(k)})$, the base of the exponent of the randomized algorithm for the k-Path problem has seen a drastic improvement. Koutis [18] used an algebraic approach based on group algebras for k-Path and gave a randomized algorithm running in time $\mathcal{O}^*(2^{3k/2}) = \mathcal{O}^*(2.83^k)$. Williams [21] augmented this approach with more random choices and several other ideas and gave an algorithm for k-Path running in time $\mathcal{O}^*(2^k)$. Currently the fastest randomized algorithm for the problem is due to Björklund et al. [6], which runs in time $\mathcal{O}^*(1.66^k)$.

While there has been a lot of work on the k-Path problem, there has been almost no progress on other cases of the Subgraph Isomorphism problem until last year. Cohen et al. gave a randomized algorithm that for an input digraph D decides in time $\mathcal{O}^*(5.704^k)$ if D contains a given out-tree with k vertices [11]. They also showed how to derandomize the algorithm in time $\mathcal{O}^*(6.14^k)$. Amini et al. [4] introduced an inclusion–exclusion based approach in the classical Color-Coding and using it gave a randomized $5.4^k n^{\mathcal{O}(t)}$ time algorithm and a deterministic $5.4^{k+o(k)} n^{\mathcal{O}(t)}$ time algorithm for the Subgraph Isomorphism problem, when F has treewidth at most t. Koutis and Williams [19] generalized their algebraic approach for k-Path to k-Tree, a special case of Subgraph Isomorphism problem where F is a tree on k-vertices, and obtained a randomized algorithm running in time $\mathcal{O}^*(2^k)$ for k-Tree. In this work we generalize the results of Koutis and Williams by extending the algebraic approach to much more general classes of graphs, namely, graphs of bounded treewidth. More precisely, we give a randomized algorithm for the Subgraph Isomorphism problem running in time $\mathcal{O}^*(2^k(nt)^t)$, when the treewidth of F is at most t. The road map suggested by Koutis and Williams [19] and Williams [21] is to reduce the problem to checking a multilinear term in a specific polynomial of degree at most k. However, the construction of such polynomial is non-trivial and requires new ideas. Our first contribution is the introduction of a new polynomial of degree at most k, namely Homomorphism Polynomial, using a relation between graph homomorphisms and injective graph homomorphisms for testing whether a graph contains a subgraph which is isomorphic to a fixed graph F. We show that if the treewidth of the pattern graph F is at most t, then it is possible to construct an arithmetic circuit of size $\mathcal{O}^*((nt)^t)$ for Homomorphism Polynomial which combined with a result of Williams [21] yields our first theorem.

In the second part of the paper we consider the problem of counting the number of pattern subgraphs, that is, the #Subgraph Isomorphism problem. A natural question here is whether we can solve the #Subgraph Isomorphism problem in $\mathcal{O}^*(c^k)$ time, when the k-vertex graph F is of bounded treewidth or whether we can even solve the #k-Path problem in $\mathcal{O}^*(c^k)$ time? Flum and Grohe [13] showed that the #k-Path problem is #W[1]-hard and hence it is very unlikely that the #k-Path problem can be solved in time $f(k)n^{\mathcal{O}(1)}$ where f is any arbitrary function of k. In another negative result, Alon and Gutner [2] have shown that one cannot hope to solve #k-Path better than $\mathcal{O}(n^{k/2})$ using the method of Color-Coding. They show this by proving that any family $\mathcal{F}$ of "balanced hash functions" from $\{1, \dots, n\}$ to $\{1, \dots, k\}$, must have size $\Omega(n^{k/2})$. On the positive side, very recently Vassilevska and Williams [20] studied various counting problems and among various other results gave an algorithm for the #k-Path problem running in time $\mathcal{O}^*(2^k (k/2)! \binom{n}{k/2})$ and space polynomial in n. Björklund et al. [5] introduced the method of "meet-in-the-middle" and gave an algorithm for the #k-Path problem running in time and space $\mathcal{O}^*(\binom{n}{k/2})$. They also gave an algorithm for #k-Path problem running in time $\mathcal{O}^*(3^{k/2}\binom{n}{k/2})$ and polynomial space, improving on the polynomial space algorithm given in [20]. We extend these results to the #Subgraph Isomorphism problem, when the pattern graph F is of bounded treewidth or pathwidth. And here also graph homomorphisms come into play! By making use of graph homomorphisms we succeed to extend the applicability of the meet-in-the-middle method to much more general structures than paths. Combined with other tools—inclusion–exclusion, the Disjoint Sum problem, separation property of graph of bounded treewidth or pathwidth and the trimmed variant of Yate's algorithm presented in [7]—we obtain the following results. Let F be a k-vertex graph and G be an n-vertex graph of pathwidth p and treewidth t. Then #Subgraph Isomorphism is solvable in time $\mathcal{O}^*(\binom{n}{k/2} n^{2p})$ and $\binom{n}{k/2} n^{\mathcal{O}(t \log k)}$ and space $\mathcal{O}^*(\binom{n}{k/2})$. We also give an algorithm for #Subgraph Isomorphism that runs in time $\mathcal{O}^*(2^k \binom{n}{k/2} n^{3p} t^{2t})$ (respectively $2^k \binom{n}{k/2} n^{\mathcal{O}(t \log k)}$) and takes $\mathcal{O}^*(n^p)$ space (respectively $\mathcal{O}^*(n^t)$ space). Thus our work not only improves on known results on Subgraph Isomorphism of Alon et al. [3] and Amini et al. [4] but it also extends and generalize most of the known results on k-Path and k-Tree of Björklund et al. [5], Koutis and Williams [19] and Williams [21].

The main theme of both algorithms, for finding and for counting a fixed pattern graph F, is to use graph homomorphisms as the main tool. Counting homomorphisms between graphs has found applications in 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. [8] for more references on counting homomorphisms. One of the main advantages of using graph homomorphisms is that in spite of their expressive power, graph homomorphisms between many structures can be counted efficiently. Secondly, it allows us to generalize various algorithm for counting subgraphs with an ease. We combine counting homomorphisms with the recent advancements on computing different transformations efficiently on subset lattice.

2. Preliminaries

Let G be a simple undirected graph without self loops and multiple edges. We denote the vertex set of G by $V(G)$ and the set of edges by $E(G)$. For a subset $W \subseteq V(G)$, by $G[W]$ we mean the subgraph of G induced by W.

2.1. Treewidth, pathwidth and nice tree-decomposition

A *tree decomposition* of an (undirected) graph G is a pair (U, T) where T is a tree whose vertices we will call *nodes* and $U = (\{U_i \mid i \in V(T)\})$ is a collection of subsets of $V(G)$ such that

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

The U_i's are called bags. The *width* of a tree decomposition $(\{U_i \mid i \in V(T)\}, T)$ equals $\max_{i \in V(T)}\{|U_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. We use notation $\mathbf{tw}(G)$ to denote the treewidth of a graph G. When in the definition of the treewidth, we restrict ourselves to paths, we get the notion of *pathwidth* of a graph and denote it by $\mathbf{pw}(G)$. We also need a notion of *nice* tree decomposition for our algorithm. A *nice* tree decomposition of a graph G is a tuple (U, T, r), where T is a tree rooted at r and (U, T) is a tree decomposition of G with the following properties. The tree T is a binary tree and every node τ of the tree is one of the following types.

1. τ has two children, say τ_1 and τ_2, and $U_\tau = U_{\tau_1} = U_{\tau_2}$; then it is called *join node*.
2. τ has one child τ_1, $|U_\tau| = |U_{\tau_1}| + 1$ and $U_{\tau_1} \subseteq U_\tau$; then it is called *introduce node*.
3. τ has one child τ_1, $|U_{\tau_1}| = |U_\tau| + 1$ and $U_\tau \subseteq U_{\tau_1}$; then it is called *forget node*.
4. τ is a leaf node of T; then it is called *base node*.

Given a tree-decomposition of width t, one can obtain a nice tree-decomposition of width t in linear time.

2.2. Graph homomorphisms

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. By $\mathrm{hom}(F, G)$, $\mathrm{inj}(F, G)$ and $\mathrm{sub}(F, G)$ we 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 denote by $\mathrm{aut}(F, F)$ the number of automorphisms from F to itself, that is bijective homomorphisms. The set $\mathrm{HOM}(F, G)$ denotes the set of homomorphisms from F to G.

2.3. Functions on the subset lattice

For two functions $f_1 : D_1 \to R_1$ and $f_2 : D_2 \to R_2$ such that for every $x \in D_1 \cap D_2$, $f_1(x) = f_2(x)$ we define the *gluing* operation $f_1 \oplus f_2$ to be a function from $D_1 \cup D_2$ to $R_1 \cup R_2$ such that $f_1 \oplus f_2(x) = f_1(x)$ if $x \in D_1$ and $f_1 \oplus f_2(x) = f_2(x)$ otherwise.

For a universe U of size n, we consider functions from 2^U (the family of all subsets of U) to $\mathbb{Z}$. For such a function $f : 2^U \to \mathbb{Z}$, the zeta transform of f is a function $f\zeta : 2^U \to \mathbb{Z}$ such that $f\zeta(S) = \sum_{X \subseteq S} f(X)$. Given f, computing $f\zeta$ using this equation in a naïve manner takes time $\mathcal{O}^*(3^n)$. However, one can do better, and compute the zeta transform in time $\mathcal{O}^*(2^n)$ using a classical algorithm of Yates [22]. In this paper we will use a "trimmed" variant of Yates's algorithm [7] that works well when the non-zero entries of f all are located at the bottom of the subset lattice. In particular, it was shown in [7] that if $f(X)$ only can be non-zero when $|X| \leqslant k$ then $f\zeta$ can be computed from f in time $\mathcal{O}^*(\sum_{i=1}^{k} \binom{n}{i})$. In our algorithm we will also use an efficient algorithm for the DISJOINT SUM problem, defined as follows. Input is two families $\mathcal{A}$ and $\mathcal{B}$ of subsets of U and two weight functions $\alpha : \mathcal{A} \to \mathbb{Z}$ and $\beta : \mathcal{B} \to \mathbb{Z}$. The objective is to calculate

$$\mathcal{A} \boxtimes \mathcal{B} = \sum_{A \in \mathcal{A}} \sum_{B \in \mathcal{B}} \begin{cases} \alpha(A)\beta(B) & \text{if } A \cap B = \emptyset, \\ 0 & \text{if } A \cap B \neq \emptyset. \end{cases}$$

Following an algorithm of Kennes [14], Björklund et al. [5] gave an algorithm to compute $\mathcal{A} \boxtimes \mathcal{B}$ in time $\mathcal{O}(n(|{\downarrow}\mathcal{A}| + |{\downarrow}\mathcal{B}|))$, where ${\downarrow}\mathcal{A} = \{X\colon \exists A \in \mathcal{A},\ X \subseteq A\}$ is the *down-closure* of $\mathcal{A}$.

2.4. Arithmetic circuits

An arithmetic circuit (or a straight line program) C over a specified ring $\mathbb{K}$ is a directed acyclic graph with nodes labeled from $\{+, \times\} \cup \{x_1, \ldots, x_n\} \cup \mathbb{K}$, where $X = \{x_1, \ldots, x_n\}$ are the input variables of C. Nodes with zero out-degree are called output nodes and those with labels from $X \cup \mathbb{K}$ are called input nodes. The *Size* of an arithmetic circuit is the number of gates in it. The *Depth* of C is the length of the longest path between an output node and an input node. The nodes in C are sometimes referred to as *gates*. It is not hard to see that with every output gate g of the circuit C we can associate a polynomial $f \in \mathbb{K}[x_1, \dots, x_n]$. For more details on arithmetic circuits see [9].

A polynomial $f \in \mathbb{K}[x_1, \dots, x_n]$ is said to have a multilinear term if there is a term of the form $c_S \prod_{i \in S} x_i$ with $c_S \neq 0$ and $\emptyset \neq S \subseteq \{1, \dots, n\}$ in the standard monomial expansion of f.

3. Algorithm for finding a subgraph

In this section we give our first result and show that the Subgraph Isomorphism problem can be solved in time $\mathcal{O}^*(2^k(nt)^t)$ when the pattern graph F has treewidth at most t. The main idea of our algorithm follows that of Koutis and Williams [19] and Williams [21] for the k-Tree problem and the k-Path problem, respectively. However, we need additional ideas for our generalizations.

First, given two graphs F and G, we will associate a polynomial $\mathcal{P}_G(X)$ where $X = \{x_v \mid v \in V(G)\}$ such that: (a) the degree of $\mathcal{P}_G(X)$ is k; (b) there is a one-to-one correspondence between the monomials of $\mathcal{P}_G$ and homomorphisms between F and G; and (c) $\mathcal{P}_G$ contains a multilinear monomial of degree k if and only if G contains a subgraph isomorphic to F. The polynomial we associate with F and G to solve the Subgraph Isomorphism problem is given by the following.

$$\text{Homomorphism Polynomial} = \mathcal{P}_G(x_1, \dots, x_n) = \sum_{\Phi \in \mathrm{HOM}(F,G)} \prod_{u \in V(F)} x_{\Phi(u)}.$$

We first show that $\mathcal{P}_G$ is "efficiently" computable by an arithmetic circuit.

Lemma 1. *Let F and G be graphs with $|V(F)| = k$ and $|V(G)| = n$. Then the polynomial $\mathcal{P}_G(x_1, \dots, x_n)$ is computable by an arithmetic circuit of size $\mathcal{O}^*((nt)^t)$ where t is the treewidth of F.*

Proof. Let F, G, k, n and t be as given in the lemma. Let $D = (U, T, r)$ be a *nice* tree decomposition of F rooted at r. We define a polynomial $f_G(T, \tau, U_\tau, S, \psi) \in \mathbb{Z}[X]$, where

- τ is a node in T;
- $U_\tau \subseteq V(F)$ is the vertex subset associated with τ;
- S be a multi-set (an element can repeat itself) of size at most $t+1$ with elements from the set $V(G)$;
- $\psi : F[U_\tau] \to G[S]$ is a multiplicity respecting homomorphism between the subgraphs induced by U_τ and S respectively; and
- $X = \{x_v \mid v \in V(G)\}$ is the set of variables.

Let V_τ denote the union of vertices contained in the bags corresponding to the nodes of subtree of T rooted at τ. At an intuitive level $f_G(T, \tau, U_\tau, S, \psi)$ represents the polynomial which contains sum of monomials of the form $\prod_{u \in V_\tau \setminus U_\tau} x_{\phi(u)}$, where ϕ is a homomorphism between $F[V_\tau]$ and G consistent with ψ, that is, ϕ is an extension of ψ to $F[V_\tau]$. Formally, the polynomial f_G can be defined inductively by going over the tree T bottom up as follows.

Case 1 *(Base case).* The node τ is a leaf node in T. Since $V_\tau = U_\tau$, there is only one homomorphism between $F[V_\tau]$ and G that is an extension of ψ, hence $f_G(T, \tau, U_\tau, S, \psi) = 1$.

Case 2. The node τ is a *join node*. Let τ_1 and τ_2 be the two children of τ and T_1 and T_2 denote the sub-trees rooted at τ_1 and τ_2 respectively. Note that $U_\tau = U_{\tau_1} = U_{\tau_2}$ and $(V_{\tau_1} \cap V_{\tau_2}) \setminus U_\tau = \emptyset$. Hence, any extension of ψ to a homomorphism between $F[V_{\tau_1}]$ and G is independent of an extension of ψ to a homomorphism between $F[V_{\tau_2}]$ and G. Thus we have

$$f_G(T, \tau, U_\tau, S, \psi) = f_G(T_1, \tau_1, U_{\tau_1}, S, \psi) f_G(T_2, \tau_2, U_{\tau_2}, S, \psi). \tag{1}$$

Case 3. The node τ is an *introduce* node in T, let τ_1 be the only child of τ, and $\{u\} = U_\tau \setminus U_{\tau_1}$. Also, let T_1 denote the sub-tree of T rooted at τ_1. In this case any extension of ψ to a homomorphism between $F[V_\tau]$ and G is in fact an extension of $\psi|_{U_{\tau_1}}$ and thus we get

$$f_G(T, \tau, U_\tau, S, \psi) = f_G\big(T_1, \tau_1, U_{\tau_1}, S \setminus \{\psi(u)\}, \psi|_{U_{\tau_1}}\big). \tag{2}$$

Case 4. The node τ is a *forget* node in T, and τ_1 is the only child of τ in T. Now, U_{τ_1} contains an extra vertex along with U_τ. Thus any extension of ψ to a homomorphism between $F[V_\tau]$ and G is a direct sum of an extension of ψ to include u and that of V_{τ_1}, where $\{u\} = U_{\tau_1} \setminus U_\tau$. Define, $Y \triangleq \{v \mid v \in V(G),\ \forall w \in U_\tau,\ wu \in E(F) \Rightarrow \psi(w)v \in E(G)\}$. For $v \in Y$, let $\psi_v : U_{\tau_1} \to S \cup \{v\}$ be such that $\psi_v|_{U_\tau} = \psi$ and $\psi_v(u) = v$. Then,

$$f_G(T, \tau, U_\tau, S, \psi) = \begin{cases} \sum_{v \in Y} (f_G(T_1, \tau_1, U_{\tau_1}, S \cup \{v\}, \psi_v) x_v) & \text{if } Y \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Let $\textsc{Hom}(U_r, G)$ denote the set of all homomorphisms between the subgraph of F induced by U_r and G. In order to consider all homomorphisms between F and G, we run through all homomorphisms ψ between $F[U_r]$ and G, and then compute $f_G(T, r, U_r, \text{Image}(\psi), \psi)$ multiplied by the monomial corresponding to ψ. Now we define

$$\mathcal{H}_G(T, r, U_r) = \sum_{\psi \in \textsc{Hom}(U_r, G)} f_G(T, r, U_r, S_\psi, \psi) \left(\prod_{u \in U_r, v = \psi(u)} x_v \right) \tag{4}$$

where we consider the set $S_\psi = \text{Image}(\psi)$ as a multi set. Now we need to show that $\mathcal{H}_G$ is efficiently computable and $\mathcal{P}_G = \mathcal{H}_G$. We first show that $\mathcal{H}_G$ is computable by an arithmetic circuit of size $\mathcal{O}^*((nt)^t)$.

Claim 1. *$\mathcal{H}_G(T, r, U_r)$ is a polynomial of degree k and is computable by an arithmetic circuit of size $\mathcal{O}^*((nt)^t)$. Here r is the root of the tree T.*

Proof. In the above definition of f_G, the only place where the degree of the polynomial increases is at forget nodes of T. The number of forget nodes in T is exactly $k - |U_r|$. Thus the degree of any f_G is $k - |U_r|$ and hence the degree of $\mathcal{H}_G$ is k.

From the definitions in Eqs. (1)–(4) above, $\mathcal{H}_G(T, r, U_r)$ can be viewed as an arithmetic circuit C with $X = \{x_v \mid v \in V(G)\}$ as variables and gates from the set $\{+, \times\}$. Any node of C is labeled either by variables from U or a function of the form $f_G(T, \tau, U_\tau, S, \psi)$. The size of the circuit is bounded by the number of possible labelings of the form $f_G(T, \tau, U_\tau, S, \psi)$, where T and U_τ are fixed. But this is bounded by $|V(T)| \cdot n^{t+1} \cdot (t+1)^{t+1} = (nt)^{t+\mathcal{O}(1)} = \mathcal{O}^*((nt)^t)$. □

Next we show that $\mathcal{H}_G$ defined above is precisely $\mathcal{P}_G$ and satisfies all the desired properties.

Claim 2. *Let $\phi : V(F) \to V(G)$. Then $\phi \in \textsc{Hom}(F, G)$ if and only if the monomial $\prod_{u \in V(F)} x_{\phi(u)}$ has a non-zero coefficient in $\mathcal{H}_G(T, r, U_r)$. In other words, we have that*

$$\mathcal{H}_G(T, r, U_r) = \mathcal{P}_G(x_1, \dots, x_n) = \sum_{\phi \in \textsc{Hom}(F, G)} \prod_{u \in V(F)} x_{\phi(u)}.$$

Proof. We first give the forward direction of the proof. Let $\phi \in \textsc{Hom}(F, G)$ and $\psi = \phi|_{U_r}$. We show an expansion of $\mathcal{H}_G(T, r, U_r)$ which contains the monomial $\prod_{u \in V(F)} x_{\phi(u)}$. We first choose the term $f_G(T, r, U_r, S_\psi, \psi) \times \prod_{u \in U_r} x_{\psi(u)}$. We expand $f_G(T, r, U_r, S_\psi, \psi)$ further according to the tree structure of T. We describe this in a generic way. Consider the expansion of $f_G(T', \tau, U_\tau, S, \chi)$. If τ is a join node we recursively expand both the sub polynomials according to Eq. (1). When τ is an introduce node we use Eq. (2). In the case when τ is a forget node, we first note that $Y \neq \emptyset$ (this is the same Y as defined in Case 4) and also that $\phi(u) \in Y$, where $u \in U_\tau \setminus U_{\tau_1}$. The last assertion follows from the definition of Y. Here, we choose the term which contains $x_{\phi(u)}$, note that there exists exactly one such term and proceed recursively.

Let M denote the monomial obtained by the above mentioned expansion. For any node $v \in V(G)$, we have $\deg_M(x_v) = |\phi^{-1}(v)|$, where $\deg_M(x_v)$ denotes the degree of the variable x_v in the monomial M. To see this, in the tree decomposition D, a node $u \in V(F)$ enters the tree through a unique forget node and this is exactly where the variable $x_{\phi(u)}$ is multiplied. Thus we have $M = \prod_{u \in V(F)} x_{\phi(u)}$. Note that this expansion is uniquely defined for a given ϕ.

For the reverse direction, consider an expansion ρ of $\mathcal{H}_G(T, r, U_r)$ into monomials and let $M = \prod x_v^{d_v}$ be a monomial of ρ, where $\sum d_v = k$. We build a $\phi \in \textsc{Hom}(F, G)$ using ρ and the structure of T. Let $f_G(T, r, U_r, S_\psi, \psi)$ be the first term chosen using Eq. (4). For every $u \in U_r$ let $\phi(u) = \psi(u)$. Inductively suppose that we are at a node τ and let T' be the corresponding subtree of T. In the case of Eqs. (1) and (2) there is no need to do anything. In the case of Eq. (3), where τ is a forget node, with $u \in U_{\tau_1} \setminus U_\tau$. If the expansion ρ chooses the term $f_G(T_1, \tau_1, U_{\tau_1}, S \cup \{v\}, \psi_v) \times x_v$, then we set $\phi(u) = v$.

It remains to show that the map $\phi : V(F) \to V(G)$ as built above is indeed a homomorphism. We prove this by showing that for any edge $uu' \in E(F)$ we have that $\phi(u)\phi(u') \in E(G)$. If uu' is an edge such that both $u, u' \in U_r$ then we are done, as by definition $\phi|_{U_r} \in \textsc{Hom}(U_r, G)$ and thus ϕ preserves all the edges between the vertices from U_r. So we assume that at least one of the end points of the edge uu' is not in U_r. By the property of tree decomposition there is a $\tau' \in T$ such that $\{u, u'\} \in U_{\tau'}$. Now since at least one of the endpoints of uu' is not in U_r, there is a node on the path between r and τ' such that either u or u' is forgotten. Let τ'' be the first node on the path starting from τ' to r in the tree T such that it does not contain both u and u'. Without loss of generality let $u \notin U_{\tau''}$ and thus τ'' is a forget node which forgets u. At any forget node, since the target node v is from the set Y, we have that ϕ preserves the edge relationships among the vertices in $U_{\tau''}$ and u. Now from Eq. (3), the property of Y and the fact that $u' \in U_{\tau''}$ we have that $\phi(u)\phi(u') \in E(G)$. □

Now by setting $\mathcal{P}_G(X) = \mathcal{H}_G(T, r, U_r)$ the lemma follows which concludes the proof. □

We also need the following proposition proved by Williams [21], which tests if a polynomial of degree k has a multilinear monomial with non-zero coefficient in time $\mathcal{O}(2^k s(n))$ where $s(n)$ is the size of the arithmetic circuit.

Proposition 1. *(See [21].) Let $P(x_1,\dots,x_n)$ be a polynomial of degree at most k, represented by an arithmetic circuit of size $s(n)$ with $+$ gates (of unbounded fan-in), $\times$ gates (of fan-in two), and no scalar multiplications. There is a randomized algorithm that on every P runs in $\mathcal{O}(2^k s(n) n^{\mathcal{O}(1)})$ time, outputs "yes" with high probability if there is a multilinear term in the sum–product expansion of P, and always outputs "no" if there is no multilinear term.*

Lemma 1 and Proposition 1 together yield our first theorem.

Theorem 1. *Let F and G be two graphs on k and n vertices respectively and $\mathbf{tw}(F) \leqslant t$. Then, there is a randomized algorithm for the* Subgraph Isomorphism *problem that runs in time $\mathcal{O}^*(2^k(nt)^t)$.*

4. Algorithms for counting subgraphs

In this section, we give algorithms for the #Subgraph Isomorphism problem, when F has either bounded treewidth or pathwidth.

4.1. Counting subgraphs with meet in the middle

When $|V(F)| = k$, the pathwidth of F is p and $|V(G)| = n$, then the running time of our algorithm for #Subgraph Isomorphism is $\mathcal{O}(\binom{n}{k/2} n^{2p+\mathcal{O}(1)})$. Roughly speaking, our algorithm decomposes $V(F)$ into three parts, the left part L, the right part R, and the separator S. Then the algorithm guesses the position of S in G, and for each such position counts the number of ways to map L and R into G, such that the mappings can be glued together at S. Thus our result is a generalization of the meet in the middle algorithm for # k-Path in an n-vertex graph by Björklund et al. [5]. However, our algorithm differs from that of Björklund et al. [5] conceptually in two important points. First, we count the number of injective homomorphisms from F to G instead of counting the number of subgraphs of G that are isomorphic to F. To get the number of subgraphs of G that are isomorphic to F we simply divide the number of injective homomorphisms from F to G by the number of automorphisms of F. The second difference is that we give an algorithm that given a k-vertex graph F of pathwidth p and an n-vertex graph G computes in time $\mathcal{O}^*(\binom{n}{k} n^p)$ the number of injective homomorphisms from F to $G[S]$ for every k-vertex subset S of G. In the #k-Path algorithm of Björklund et al. [5], a simple dynamic programming algorithm to count k-paths in $G[S]$ for every k-vertex subset S, running in time $\mathcal{O}^*(\binom{n}{k})$ is presented, however this algorithm does not seem to generalize to more complicated pattern graphs F. Interestingly, our algorithm to compute the number of injective homomorphisms from F to $G[S]$ for every S is instead based on inclusion–exclusion and the trimmed variant of Yates's algorithm presented in [7]. In order to implement the meet-in-the-middle approach, we will use the following fact about graphs of bounded pathwidth.

Proposition 2 *(Folklore). Let F be a k-vertex graph of pathwidth p. Then there exists a partitioning of $V(F)$ into $V(F) = L \uplus S \uplus R$, such that $|S| \leqslant p$, $|L|, |R| \leqslant k/2$ and no edge of F has one endpoint in L and the other in R.*

Proof. The vertices of a graph F of pathwidth p can be ordered as $v_1 \dots v_k$ such that for any $i \leqslant k$ there is a subset $S_i \subseteq \{v_1 \dots v_i\}$ with $|S_i| \leqslant p$, such that there are no edges of F with one endpoint in $\{v_1 \dots v_i\} \setminus S_i$ and the other in $\{v_{i+1}, \dots, v_k\}$. Such an ordering is obtained, for example, in [15]. Choose $L' = \{v_1 \dots v_{\lceil k/2 \rceil}\}$, $S = S_{\lceil k/2 \rceil}$, $L = L' \setminus S$ and $R = \{v_{\lceil k/2 \rceil+1} \dots v_k\}$. Then L, S and R have the claimed properties. □

Let $V(F) = L \uplus S \uplus R$ be a partitioning of $V(F)$ as given by Proposition 2, and let $L^+ = L \cup S$ and $R^+ = R \cup S$. For a map $g : S \to V(G)$ and a set S' such that $S \subseteq S'$ and a set Q we define $\hom_g(F[S'], G[Q])$ to be the number of injective homomorphisms from $F[S']$ to $G[Q]$ coinciding with g on S. Similarly we let $\mathrm{inj}_g(F[S'], Q)$ to be the number of homomorphisms from F to $G[Q]$ coinciding with g on S. If we guess how an injective homomorphism maps $F[S]$ to G we get $\mathrm{inj}(F, G) = \sum_g \mathrm{inj}_g(F, G)$, where the sum is taken over all injective maps g from S to $V(G)$. For a given map g, we define the set of families $\mathcal{L}_g = \{Q \subseteq V(G)\colon\ |Q| = |L|\}$ and $\mathcal{R}_g = \{Q \subseteq V(G)\colon\ |Q| = |R|\}$. The weight of a set $Q \in \mathcal{L}_g$ is defined as $\alpha_g^L(Q) = \mathrm{inj}_g(F[L^+], G[Q \cup g(S)])$ and the weight of a set $Q \in \mathcal{R}_g$ is set to $\alpha_g^R(Q) = \mathrm{inj}_g(F[R^+], G[Q \cup g(S)])$.

For any $Q_1 \in \mathcal{L}_g$ and $Q_2 \in \mathcal{R}_g$ such that $Q_1 \cap Q_2 = \emptyset$, if we take an injective homomorphism h_1 from $F[L^+]$ to $G[Q_1 \cup g(S)]$ coinciding with g on S and another injective homomorphism h_2 from $F[R^+]$ to $G[Q_2 \cup g(S)]$ coinciding with g on S and glue them together, we obtain an injective homomorphism $h_1 \oplus h_2$ from F to G. Furthermore two homomorphisms from F to G can only be equal if they coincide on all vertices of F. Thus, if $Q_1' \in \mathcal{L}_g$, $Q_2' \in \mathcal{R}_g$ and h_1' and h_2' are injective homomorphisms from $F[L^+]$ to $G[Q_1' \cup g(S)]$ and from $F[R^+]$ to $G[Q_2' \cup g(S)]$ respectively we have that $h_1 \oplus h_2 = h_1' \oplus h_2'$ if and only if $h_1' = h_1$ and $h_2' = h_2$. Also, for any injective homomorphism h from F to G that coincides with g on S we can decompose it into an injective homomorphism h_1 from $F[L^+]$ to $G[S \cup Q_1]$ and another injective homomorphism h_2 from $F[R^+]$ to $G[S \cup Q_2]$ such that $Q_1 \in \mathcal{L}_g$, $Q_2 \in \mathcal{R}_g$ and $Q_1 \cap Q_2 = \emptyset$. Then $\mathrm{inj}_g(F, G) = \mathcal{L}_g \boxtimes \mathcal{R}_g$ and hence

$$\mathrm{inj}(F, G) = \sum_g \mathcal{L}_g \boxtimes \mathcal{R}_g. \tag{5}$$

Proposition 3. *(See [5,14].) Given two families $\mathcal{A}$ and $\mathcal{B}$ together with weight functions $\alpha : \mathcal{A} \to \mathbb{N}$ and $\beta : \mathcal{B} \to \mathbb{N}$ we can compute the disjoint sum $\mathcal{A} \boxtimes \mathcal{B}$ in time $\mathcal{O}(n(|{\downarrow}\mathcal{A}| + |{\downarrow}\mathcal{B}|))$ where n is the number of distinct elements covered by the members of $\mathcal{A}$ and $\mathcal{B}$. Here ${\downarrow}\mathcal{A} = \{X\colon \exists A \in \mathcal{A},\ X \subseteq A\}$.*

We would like to use Proposition 3 together with Eq. (5) in order to compute $\mathrm{inj}(F,G)$. Thus, given the mapping $g : S \to V(G)$ we need to compute $\mathcal{L}_g$, $\mathcal{R}_g$, α_g^L and α_g^R. Listing $\mathcal{L}_g$ and $\mathcal{R}_g$ can be done easily in $\binom{n}{k/2} + \binom{n}{k/2}$ time, so it remains to compute efficiently α_g^L and α_g^R.

Lemma 2. *Let G be an n-vertex graph, F be an ℓ-vertex graph of treewidth t, $S \subseteq V(F)$ and g be a function from S to $V(G)$. There is an algorithm to compute $\mathrm{inj}_g(F, G[Q \cup g(S)])$ for all $\ell - |S|$ sized subsets Q of $V(G) \setminus g(S)$ in time $\mathcal{O}^*((\sum_{j=1}^{\ell-|S|}\binom{n}{j}) \cdot n^p)$.*

Proof. We claim that the following inclusion–exclusion formula holds for $\mathrm{inj}_g(F, G[Q \cup g(S)])$.

$$\mathrm{inj}_g\big(F, G\big[Q \cup g(S)\big]\big) = \sum_{X \subseteq Q} (-1)^{|Q|-|X|} \mathrm{hom}_g\big(F, G\big[X \cup g(S)\big]\big). \tag{6}$$

To prove the correctness of Eq. (6), we first show that if there is an injective homomorphism f from F to $G[Q \cup g(S)]$ coinciding with g on S then its contribution to the sum is exactly one. Notice that since $|S| + |Q| = |V(F)|$, all injective homomorphisms that coincide with g on S only contribute when $X = Q$ and thus are counted exactly 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 homomorphism h from F to $G[S \cup Q]$, which coincides with g on S, is not an injective homomorphism then its total contribution to the sum is zero, which will conclude the correctness proof of the equation. Observe that since h is not an injective homomorphism it misses some vertices of Q. Thus $h(V(F)) \cap Q = W$ for some subset $W \subset Q$. We now observe that h is counted only when we are counting homomorphisms from F to $G[X \cup g(S)]$ such that $W \subseteq X$. The total contribution of h in the sum, taking into account the signs, is

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

Thus, we have shown that if h is not an injective homomorphism then its contribution to the sum is zero, and hence Eq. (6) holds.

Observe that since $|Q| = \ell - |S|$, we can rewrite $(-1)^{|Q|-|X|}$ as $(-1)^{\ell-|S|-|X|}$. Define $\gamma(X) = (-1)^{\ell-|S|-|X|}\mathrm{hom}_g(F, G[X \cup g(S)])$, then we can rewrite Eq. (6) as follows:

$$\mathrm{inj}_g\big(F, G\big[Q \cup g(S)\big]\big) = \gamma\zeta(Q).$$

We start by pre-computing a table containing $\gamma(Q')$ for every Q' with $|Q'| \leqslant \ell - |S|$. To do this we need to compute $\mathrm{hom}_g(F, G[Q' \cup g(S)])$ for all subsets Q' of $V(G) \setminus g(S)$ of size at most $\ell - |S|$. There are at most $\sum_{j=1}^{\ell-|S|}\binom{n}{j}$ such subsets, and for each subset Q' we can compute $\mathrm{hom}_g(F, G[Q' \cup g(S)])$, and hence also $\alpha(Q')$ using the dynamic programming algorithm of Díaz et al. [12] in time $\mathcal{O}^*(n^p)$. Now, to compute $\gamma\zeta(Q)$ for all $Q \subseteq V(G) \setminus g(S)$ of size $\ell - |S|$ we apply the algorithm for the trimmed zeta transform (Algorithm Z) from [7]. This algorithm runs in time $\mathcal{O}^*(\sum_{j=1}^{\ell-|S|}\binom{n}{j})$. Thus the total running time of the algorithm is then $\mathcal{O}^*((\sum_{j=1}^{\ell-|S|}\binom{n}{j}) \cdot n^p)$. This concludes the proof. □

We are now in position to prove the main theorem of this section.

Theorem 2. *Let G be an n-vertex graph and F be a k-vertex graph of pathwidth p. Then we can solve the #Subgraph Isomorphism problem in time $\mathcal{O}^*(\binom{n}{k/2}n^{2p})$ and space $\mathcal{O}^*(\binom{n}{k/2})$.*

Proof. We apply Proposition 3 together with Eq. (5) in order to compute $\mathrm{inj}(F,G)$. In particular, for every mapping $g : S \to V(G)$ we list $\mathcal{L}_g$ and $\mathcal{R}_g$ and compute α_g^L and α_g^R using the algorithm from Lemma 2. Finally, to compute $\mathcal{L}_g \boxtimes \mathcal{R}_g$ we apply Proposition 3.

The sum in Eq. (5) runs over $\binom{n}{p}p! \leqslant n^p$ different choices for g. For each g, listing $\mathcal{L}_g$ and computing α_g^L, and listing $\mathcal{R}_g$ and computing α_g^R, takes $\mathcal{O}^*(\binom{n}{k/2-|S|}n^p)$ and $\mathcal{O}^*(\binom{n}{k/2}n^p)$ time respectively. Finally, computing $\mathcal{L}_g \boxtimes \mathcal{R}_g$ takes time $\mathcal{O}^*(\binom{n}{k/2})$. Thus the total running time for the algorithm to compute $\mathrm{inj}(F,G)$ is $\mathcal{O}^*(\binom{n}{k/2}n^{2p})$.

To compute the number of occurrences of F as a subgraph in G, we use the basic fact that the number of occurrences of F in G is $\mathrm{inj}(F,G)/\mathrm{aut}(F)$ [4]. Since $\mathrm{aut}(F) = \mathrm{inj}(F,F)$ we can compute $\mathrm{aut}(F)$ using the algorithm for computing $\mathrm{inj}(F,G)$ in time $\mathcal{O}^*(\binom{k}{k/2}n^{2p}) = \mathcal{O}^*(2^k n^{2p})$. This concludes the proof of the theorem. □

4.2. Polynomial space algorithm

In this section we give a polynomial space variant of our algorithm presented in the previous section. Our proof is similar in spirit to the one described by Björklund et al. [5] for the #k-PATH problem.

Theorem 3. [$\star$] *Let G be an n-vertex graph and F be a k-vertex graph of pathwidth p. Then we can solve the* #SUBGRAPH ISOMORPHISM *problem in time $\mathcal{O}^*(\binom{n}{k/2}2^k n^{3p} t^{2t})$ and $\mathcal{O}^*(n^p)$ space.*

Proof. For our proof we need the following proposition which gives a relationship between $\mathrm{inj}(F,G)$ and $\mathrm{hom}(F,G)$.

Proposition 4. *(See [4].) Let F and G be two graphs with $|V(G)| = |V(F)|$. Then*

$$\begin{aligned}\mathrm{inj}(F,G) &= \sum_{W\subseteq V(G)} (-1)^{|W|}\,\mathrm{hom}\bigl(F, G\bigl[V(G)\setminus W\bigr]\bigr)\\ &= \sum_{W\subseteq V(G)} (-1)^{|V|-|W|}\,\mathrm{hom}\bigl(F, G[W]\bigr).\end{aligned}$$

By Eq. (5) we know that $\mathrm{inj}(F,G) = \sum_g \mathcal{L}_g \boxtimes \mathcal{R}_g$. We first show how to compute $\mathcal{L}_g \boxtimes \mathcal{R}_g$ for a fixed map $g : S \to V(G)$. For brevity, we use the Iverson Bracket notation: $[P]=1$ if P is true, and $[P]=0$ if P is false.

$$\begin{aligned}\mathcal{L}_g \boxtimes \mathcal{R}_g &= \sum_{M\in\mathcal{L}_g}\sum_{N\in\mathcal{R}_g}[M\cap N=\emptyset]\alpha_g^L(M)\beta_g^R(N)\\ &= \sum_{M\in\mathcal{L}_g}\sum_{N\in\mathcal{R}_g}\sum_{\{X\subseteq V(G),|X|\leqslant k/2\}}(-1)^{|X|}[X\subseteq M\cap N]\alpha_g^L(M)\beta_g^R(N)\\ &= \sum_{\{X\subseteq V(G),|X|\leqslant k/2\}}(-1)^{|X|}\sum_{M\in\mathcal{L}_g}\sum_{N\in\mathcal{R}_g}[X\subseteq M][X\subseteq N]\alpha_g^L(M)\beta_g^R(N)\\ &= \sum_{\{X\subseteq V(G),|X|\leqslant k/2\}}(-1)^{|X|}\Bigl(\sum_{M\in\mathcal{L}_g,M\supseteq X}\alpha_g^L(M)\Bigr)\Bigl(\sum_{N\in\mathcal{R}_g,N\supseteq X}\beta_g^R(N)\Bigr)\\ &= \sum_{i=1}^{k/2}\sum_{\{X\subseteq V(G),|X|=i\}}(-1)^{i}\Bigl(\sum_{M\in\mathcal{L}_g,M\supseteq X}\alpha_g^L(M)\Bigr)\Bigl(\sum_{N\in\mathcal{R}_g,N\supseteq X}\beta_g^R(N)\Bigr).\end{aligned} \tag{7}$$

For every $M\in\mathcal{L}_g$, by Eq. (6), we know that the following inclusion–exclusion formula holds for $\alpha_g^L(M)$.

$$\begin{aligned}\alpha_g^L(M) &= \mathrm{inj}_g\bigl(F\bigl[L^+\bigr], G\bigl[M\cup g(S)\bigr]\bigr)\\ &= \sum_{M'\subseteq M}(-1)^{|M|-|M'|}\,\mathrm{hom}_g\bigl(F\bigl[L^+\bigr], G\bigl[M'\cup g(S)\bigr]\bigr).\end{aligned}$$

We can compute $\mathrm{hom}_g(F[L^+], G[M'\cup g(S)])$ in $\mathcal{O}^*((nt)^{2p})$ time and $\mathcal{O}^*(n^p)$ space using the dynamic programming algorithm of Díaz et al. [12]. Hence, using this we can compute $\alpha_g^L(M)$ in time $\mathcal{O}^*(2^{|M|}(nt)^{2p})$. Similarly we can compute $\alpha_g^R(N)$ in time $\mathcal{O}^*(2^{|N|}(nt)^{2p})$ for every $N\in\mathcal{R}_g$. Now using Eq. (7) we can bound the running time to compute $\mathcal{L}_g\boxtimes\mathcal{R}_g$ as follows:

$$\begin{aligned}&\sum_{i=1}^{k/2}\Bigl(\binom{n}{i}\binom{n-i}{|L|-i}\mathcal{O}^*\bigl(2^{|L|}(nt)^{2p}\bigr)+\binom{n}{i}\binom{n-i}{|R|-i}\mathcal{O}^*\bigl(2^{|R|}(nt)^{2p}\bigr)\Bigr)\\ &\quad\leqslant \sum_{i=1}^{k/2}\Bigl(2^{k/2}\binom{n}{|L|}\mathcal{O}^*\bigl(2^{|L|}(nt)^{2p}\bigr)+2^{k/2}\binom{n}{|R|}\mathcal{O}^*\bigl(2^{|R|}(nt)^{2p}\bigr)\Bigr)\\ &\quad\leqslant \sum_{i=1}^{k/2}\Bigl(\binom{n}{k/2}\mathcal{O}^*\bigl(2^{k}(nt)^{2p}\bigr)+\binom{n}{k/2}\mathcal{O}^*\bigl(2^{k}(nt)^{2p}\bigr)\Bigr)=k\binom{n}{k/2}\mathcal{O}^*\bigl(2^{k}(nt)^{2p}\bigr).\end{aligned}$$

This implies that the time taken to compute $\mathrm{inj}(F,G) = \sum_g \mathcal{L}_g \boxtimes \mathcal{R}_g$ is upper bounded by $\mathcal{O}^*(2^k\binom{n}{k/2}n^{3p}t^{2t})$, as the total number of choices for g is upper bounded by $\binom{n}{p}p! \leqslant n^p$. Finally, to compute the number of occurrences of F in G, we use the basic fact that the number of occurrences of F in G is $\mathrm{inj}(F,G)/\mathrm{aut}(F)$ [4] as in the proof of Theorem 2. We can compute $\mathrm{aut}(F) = \mathrm{inj}(F, F)$, using the polynomial space algorithm given by Proposition 4 for computing $\mathrm{inj}(F, G)$ and using the dynamic programming algorithm of Díaz et al. [12], in time $\sum_{i=1}^{k} \binom{k}{i} \mathcal{O}^*((kp)^{2p}) = \mathcal{O}^*(2^k k^{4p})$ and space $\mathcal{O}^*(n^p)$. This concludes the proof of the theorem. $\square$

Theorems 2 and 3 can easily be generalized to handle the case when F has treewidth at most t by observing that if $\mathbf{tw}(F) \leqslant t$ then $\mathbf{pw}(F) \leqslant (t+1)\log(k-1)$ [16] and that the dynamic programming algorithm of Díaz et al. [12] works for graphs of bounded treewidth.

5. Conclusion

In this paper we considered the Subgraph Isomorphism problem and the #Subgraph Isomorphism problem and gave the best known algorithms, in terms of time and space requirements, for these problems when the pattern graph F is restricted to graphs of bounded treewidth or pathwidth. Counting graph homomorphisms served as a main tool for all our algorithms. We combined counting graph homomorphisms with various other recently developed tools in parameterized and exact algorithms like meet-in-the-middle, trimmed variant of Yates's algorithm, the Disjoint Sum problem and algebraic circuits and formulas to obtain our algorithms. We conclude with an intriguing open problem about a special case of the Subgraph Isomorphism problem. Can we solve the Subgraph Isomorphism problem in time $\mathcal{O}^*(c^n)$, c a fixed constant, when the maximum degree of F is 3?

Acknowledgment

We thank Mikko Koivisto for pointing us to an error in the preliminary draft of this paper and several useful suggestions.

References

[1] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, S.C. Sahinalp, Biomolecular network motif counting and discovery by color coding, in: ISMB, 2008, pp. 241–249.
[2] N. Alon, S. Gutner, Balanced families of perfect hash functions and their applications, ACM Trans. Algorithms 6 (2010).
[3] N. Alon, R. Yuster, U. Zwick, Color-coding, J. ACM 42 (1995) 844–856.
[4] O. Amini, F.V. Fomin, S. Saurabh, Counting subgraphs via homomorphisms, in: ICALP (1), in: Lecture Notes in Comput. Sci., vol. 5555, 2009, pp. 71–82.
[5] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Counting paths and packings in halves, in: ESA, in: Lecture Notes in Comput. Sci., vol. 5757, 2009, pp. 578–586.
[6] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Narrow sieves for parameterized paths and packings, in: CoRR, 2010, 1007.1161v1 [cs.DS].
[7] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Trimmed Moebius inversion and graphs of bounded degree, Theory Comput. Syst. 47 (2010) 637–654.
[8] C. Borgs, J. Chayes, L. Lovász, V.T. Sós, K. Vesztergombi, Counting graph homomorphisms, in: Topics in Discrete Mathematics, in: Algorithms Combin., vol. 26, Springer, Berlin, 2006, pp. 315–371.
[9] P. Bürgisser, M. Clausen, M. Shokrollahi, Algebraic Complexity Theory, Springer-Verlag, 1997.
[10] J. Chen, S. Lu, S.-H. Sze, F. Zhang, Improved algorithms for path, matching, and packing problems, in: SODA, 2007, pp. 298–307.
[11] N. Cohen, F.V. Fomin, G. Gutin, E.J. Kim, S. Saurabh, A. Yeo, Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem, J. Comput. System Sci. 76 (2010) 650–662.
[12] J. Díaz, M.J. Serna, D.M. Thilikos, Counting h-colorings of partial k-trees, Theoret. Comput. Sci. 281 (2002) 291–309.
[13] J. Flum, M. Grohe, The parameterized complexity of counting problems, SIAM J. Comput. 33 (2004) 892–922.
[14] R. Kennes, Computational aspects of the Moebius transform of a graph, IEEE Trans. Syst. Man Cybernet. 22 (1991) 201–223.
[15] N.G. Kinnersley, The vertex separation number of a graph equals its path-width, Inform. Process. Lett. 42 (1992) 345–350.
[16] T. Kloks, Treewidth, Computations and Approximations, Lecture Notes in Comput. Sci., vol. 842, Springer, 1994.
[17] J. Kneis, D. Mölle, S. Richter, P. Rossmanith, Divide-and-color, in: WG, in: Lecture Notes in Comput. Sci., vol. 4271, 2006, pp. 58–67.
[18] I. Koutis, Faster algebraic algorithms for path and packing problems, in: ICALP (1), in: Lecture Notes in Comput. Sci., vol. 5125, 2008, pp. 575–586.
[19] I. Koutis, R. Williams, Limits and applications of group algebras for parameterized problems, in: ICALP (1), in: Lecture Notes in Comput. Sci., vol. 5555, 2009, pp. 653–664.
[20] V. Vassilevska, R. Williams, Finding, minimizing, and counting weighted subgraphs, in: STOC, 2009, pp. 455–464.
[21] R. Williams, Finding paths of length k in $O^*(2^k)$ time, Inform. Process. Lett. 109 (2009) 315–318.
[22] F. Yates, The design and analysis of factorial experiments, Technical Communication 35 (1937).