Graph Pattern Polynomials

Markus Bläser
Department of Computer Science, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
mblaeser@cs.uni-saarland.de

Balagopal Komarath
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
bkomarath@cs.uni-saarland.de

Karteek Sreenivasaiah[1]
Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, India
karteek@iith.ac.in

Abstract

Given a *host* graph G and a *pattern* graph H, the induced subgraph isomorphism problem is to decide whether G contains an induced subgraph that is isomorphic to H. We study the time complexity of induced subgraph isomorphism problems when the pattern graph is fixed. Nešetřil and Poljak gave an $O(n^{k\omega})$ time algorithm that decides the induced subgraph isomorphism problem for *any* $3k$ vertex pattern graph (the universal algorithm), where ω is the matrix multiplication exponent. Improvements are not known for *any* infinite pattern family.

Algorithms faster than the universal algorithm are known only for a finite number of pattern graphs. In this paper, we show that there exists infinitely many pattern graphs for which the induced subgraph isomorphism problem has algorithms faster than the universal algorithm.

Our algorithm works by reducing the pattern detection problem into a multilinear term detection problem on special classes of polynomials called graph pattern polynomials. We show that many of the existing algorithms including the universal algorithm can also be described in terms of such a reduction. We formalize this class of algorithms by defining graph pattern polynomial families and defining a notion of reduction between these polynomial families. The reduction also allows us to argue about relative hardness of various graph pattern detection problems within this framework. We show that solving the induced subgraph isomorphism for any pattern graph that contains a k-clique is at least as hard detecting k-cliques. An equivalent theorem is not known in the general case.

In the full version of this paper, we obtain new algorithms for P_5 and C_5 that are optimal under reasonable hardness assumptions. We also use this method to derive new combinatorial algorithms – algorithms that do not use fast matrix multiplication – for paths and cycles. We also show why graph homomorphisms play a major role in algorithms for subgraph isomorphism problems. Using this, we show that the arithmetic circuit complexity of the graph homomorphism polynomial for $K_k - e$ (The k-clique with an edge removed) is related to the complexity of many subgraph isomorphism problems. This generalizes and unifies many existing results.

2012 ACM Subject Classification Theory of computation → Probabilistic computation, Theory of computation → Problems, reductions and completeness

Keywords and phrases algorithms, induced subgraph detection, algebraic framework

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.18

[1] Part of this work was done while the author was at Saarland University, Saarbrücken, Germany.

Related Version The full version of this paper is available on arXiv: `https://arxiv.org/abs/1809.08858`.

Acknowledgements The authors thank Cornelius Brand and Holger Dell for helpful discussions during the early parts of this work. The authors also thank the anonymous reviewers for comments that helped improve the presentation in the paper.

1 Introduction

The *induced subgraph isomorphism problem* asks, given simple and undirected graphs G and H, whether there is an induced subgraph of G that is isomorphic to H. The graph G is called the host graph and the graph H is called the pattern graph. This problem is NP-complete (See [8], problem [GT21]). If the pattern graph H is fixed, there is a simple $O(n^{|V(H)|})$ time algorithm to decide the induced subgraph isomorphism problem for H. We study the time complexity of the induced subgraph isomorphism problem for fixed pattern graphs on the Word-RAM model.

The earliest non-trivial algorithm for this problem was given by Itai and Rodeh [9] who showed that the number of triangles can be computed in $O(n^\omega)$ time on n-vertex graphs, where ω is the exponent of matrix multiplication. Later, Nešetřil and Poljak[11] generalized this algorithm to count K_{3k} in $O(n^{k\omega})$ time, where K_{3k} is the clique on $3k$ vertices. Eisenbrand and Grandoni [3] extended this algorithm further to count K_{3k+j} for $j \in \{0, 1, 2\}$ using rectangular matrix multiplication in $O(n^{\omega(k+\lceil j/2\rceil, k, k+\lfloor j/2\rfloor)})$ time. Here $\omega(i, j, k)$ denotes the exponent of the running time of matrix multiplication when multiplying an $i \times j$ matrix with a $j \times k$ matrix. It is known that detecting/counting any k-vertex pattern is easier than detecting/counting K_k. Therefore, these algorithms are called "universal" algorithms.

Our Contributions

Algorithms that improve the universal algorithm for specific pattern graphs are only known for small fixed values of k. For example, the induced subgraph isomorphism problem for P_4 can be solved in $O(n + m)$ time [1] and all 4-vertex graphs other than K_4 can be detected in $O(n^\omega)$ time [14]. In Section 5, we give the first algorithm that detects infinitely many pattern graphs faster than the universal algorithm.

Our algorithm works by reducing the induced subgraph isomorphism problem into detecting multilinear terms in a related polynomial. This idea has been previously used by many authors (See [13], [10], [5], and [7] for its application to subgraph isomorphism problems) to solve combinatorial problems efficiently. A major contribution of our work is a general framework that can describe many existing algorithms for subgraph isomorphism problems. We show that graph pattern[2] detection problems can be reformulated as the problem of detecting multilinear terms in special classes of polynomials called *graph pattern polynomials* (Defined in Section 4).

We also define a notion of reduction between these polynomials that allows us to argue about the relative hardness of the graph pattern detection problems. It is known that detecting an induced path on $2k$ vertices is at least as hard as detecting a K_k [6]. Intuitively,

[2] Examples of graph patterns include subgraph isomorphisms, induced subgraph isomorphisms, and graph homomorphisms

any pattern graph H that contains a K_k (or equivalently, an independent set on k vertices) should be as hard to detect as a K_k. But this is not known. In Section 6, we show that the graph pattern polynomial for K_k can be reduced to the polynomial that corresponds to the induced subgraph isomorphism problem for H for any H that contains a K_k. This shows that if we can obtain better algorithms for H using our framework, then we obtain better algorithms for K_k. We show that all existing algorithms for induced subgraph isomorphism problems can be either described using our framework or we can obtain algorithms with matching running times using our framework. Therefore, these reductions can be viewed as showing the limitations of current methods for solving subgraph isomorphism problems.

In Section 7, we discuss the results in the full version of this paper. In Section 3, we show how to use graph pattern polynomials to obtain a linear-time algorithm for detecting paths on four vertices.

2 Preliminaries

For a polynomial f, we use $deg(f)$ to denote the degree of f. A monomial is called multilinear, if every variable in it has degree at most one. We use $ML(f)$ to denote the multilinear part of f, that is, the sum of all multilinear monomials in f. An arithmetic circuit computing a polynomial $P \in K[x_1, \ldots, x_n]$ is a circuit with $+$, $\times$ gates where the input gates are labelled by variables or constants from the underlying field and one gate is designated as the output gate. The size of an arithmetic circuit is the number of wires in the circuit. For indeterminates $x_1, \ldots, x_n$ and a set $S = \{s_1, \ldots, s_p\} \subseteq \{1, \ldots, n\}$ of indices, we write x_S to denote the product $x_{s_1} \cdots x_{s_p}$.

An induced subgraph isomorphism from H to G is an injective function $\phi : V(H) \overset{ind}{\mapsto} V(G)$ such that $\{u, v\} \in E(H) \iff \{\phi(u), \phi(v)\} \in E(G)$. Any function from $V(H)$ to $V(G)$ can be extended to unordered pairs of vertices of H as $\phi(\{u, v\}) = \{\phi(u), \phi(v)\}$. A subgraph isomorphism from H to G is an injective function $\phi : V(H) \overset{sub}{\mapsto} V(G)$ such that $\{u, v\} \in E(H) \implies \{\phi(u), \phi(v)\} \in E(G)$. Two subgraph isomorphisms or induced subgraph isomorphisms are considered different only if the set of edges in the image are different. A graph homomorphism from H to G is a function $\phi : V(H) \overset{hom}{\mapsto} V(G)$ such that $\{u, v\} \in E(H) \implies \{\phi(u), \phi(v)\} \in E(G)$. Unlike isomorphisms, we consider two distinct functions that yield the same set of edges in the image as distinct graph homomorphisms. We define $\phi(S) = \{\phi(s) : s \in S\}$.

We write $H \sqsubseteq H'$ ($H \sqsupseteq H'$) to specify that H is a subgraph (supergraph) of H'. The number $tw(H)$ stands for the treewidth of H. We denote the number of automorphisms of H by $\#aut(H)$. The graph K_n is the complete graph on n vertices labelled using $[n]$. We use the fact that $\#aut(H) = 1$ for almost all graphs in many of our results. In this paper, we will frequently consider graphs where vertices are labelled by tuples. A vertex (i, p) is said to have *label* i and *colour* p. An edge $\{(i_1, p_1), (i_2, p_2)\}$ has *label* $\{i_1, i_2\}$ and *colour* $\{p_1, p_2\}$. We will sometimes write this edge as $(\{i_1, i_2\}, \{p_1, p_2\})$. Note that both $\{(i_1, p_1), (i_2, p_2)\}$ and $\{(i_2, p_1), (i_1, p_2)\}$ are written as $(\{i_1, i_2\}, \{p_1, p_2\})$. But the context should make it clear which edge is being rewritten.

We will often use the following short forms to denote specific pattern graphs:

K_ℓ :	A clique on ℓ vertices	I_ℓ :	An independent set on ℓ vertices
$K_\ell - e$:	A K_ℓ with an edge removed	$K_\ell + e$:	A K_ℓ and one more edge on $\ell + 1$ vertices
P_ℓ :	A Path on ℓ vertices	C_ℓ :	A cycle on ℓ vertices

3 A Motivating Example: Induced-P_4 Isomorphism

In this section, we sketch a one-sided error, randomized $O(n^2)$ time algorithm for the induced subgraph isomorphism problem for P_4 to illustrate the techniques used to derive algorithms in this paper.

We start by giving an algorithm for the subgraph isomorphism problem for P_4. Consider the following polynomial:

$$N_{P_4,n} = \sum_{(p,q,r,s):p<s} y_p y_q y_r y_s x_{\{p,q\}} x_{\{q,r\}} x_{\{r,s\}}$$

where the summation is over all quadruples over $[n]$ where all four elements are distinct. Each of the y variables corresponds to a vertex of a possible P_4 and the x variables correspond to the edges. Hence each monomial in the above polynomial corresponds naturally to a P_4 on the vertices p, q, r, s chosen in the summation. The condition $p < s$ ensures that each path has exactly one monomial corresponding to it.

Given an n-vertex host graph G and an arithmetic circuit for $N_{P_4,n}$, we can construct an arithmetic circuit for the polynomial $N_{P_4,n}(G)$ on the y variables obtained by substituting $x_e = 0$ when $e \notin E(G)$ and $x_e = 1$ when $e \in E(G)$. The polynomial $N_{P_4,n}(G)$ can be written as $\sum_X a_X y_X$ where the summation is over all four vertex subsets X of $V(G)$ and a_X is the number of P_4s in the induced subgraph $G[X]$. Therefore, we can decide whether G has a subgraph isomorphic to P_4 by testing whether $N_{P_4,n}(G)$ is identically 0. Since the degree of this polynomial is a constant k, this can be done in time linear in the size of the arithmetic circuit computing $N_{P_4,n}$.

However, we do not know how to construct a $O(n^2)$ size arithmetic circuit for $N_{P_4,n}$. Instead, we construct a $O(n^2)$ size arithmetic circuit for the following polynomial called the walk polynomial:

$$Hom_{P_4,n} = \sum_{\phi:P_4 \overset{hom}{\mapsto} K_n} \prod_{v \in V(P_4)} z_{v,\phi(v)} y_{\phi(v)} \prod_{e \in E(P_4)} x_{\phi(e)}$$

Similar to $N_{P_4,n}$, the y and x variables correspond to vertices and edges respectively. The z variables play the role of fixing the mapping from P_4 to K_n that is chosen in the summation. This polynomial is also called the homomorphism polynomial for P_4 because its terms are in one-to-one correspondence with graph homomorphisms from P_4 to K_n. As before, we consider the polynomial $Hom_{P_4,n}(G)$ obtained by substituting for the x variables appropriately. The crucial observation is that $Hom_{P_4,n}(G)$ contains a multilinear term if and only if $N_{P_4,n}(G)$ is not identically zero. This is because the multilinear terms of $Hom_{P_4,n}$ correspond to injective homomorphisms from P_4 which in turn correspond to subgraph isomorphisms from P_4. More specifically, each P_4 corresponds to two injective homomorphisms from P_4 since P_4 has two automorphisms. Therefore, we can test whether G has a subgraph isomorphic to P_4 by testing whether $Hom_{P_4,n}(G)$ has a multilinear term. It is known that the polynomial $p_4 = Hom_{P_4,n}$ has $O(n^2)$ size circuits using the following inductive construction:

$$\begin{aligned} p_{1,v} &= y_v, v \in [n] \\ p_{i+1,v} &= z_{i+1,v} \sum_{u \in [n]} p_{i,u} y_v x_{\{u,v\}}, v \in [n], i \geq 1 \\ p_4 &= \sum_{v \in [n]} p_{4,v} \end{aligned}$$

The above construction can be extended to construct p_k for any k and not just $k = 4$. This method is used in [13] to obtain an $O(2^k(n+m))$ time algorithm for the subgraph isomorphism problem for P_k.

In fact, the above method works for any pattern graph H. Extend the definitions above to define $N_{H,n}$ and $Hom_{H,n}$ in the natural fashion. Then, we can test whether an n-vertex graph G has a subgraph isomorphic to H by testing whether $N_{H,n}(G)$ is identically zero which in turn can be done by testing whether $Hom_{H,n}(G)$ has a multilinear term. Therefore, the complexity of subgraph isomorphism problem for any pattern H is as easy as constructing the homomophism polynomial for H. This method is used by Fomin et. al. [7] to obtain efficient algorithms for subgraph isomorphism problems.

We now turn our attention to the induced subgraph isomorphism problem for P_4. We note that the induced subgraph isomorphism problem for P_k is much harder than the subgraph isomorphism problem for P_k. The subgraph isomorphism problem for P_k has a linear time algorithm as seen above but the induced subgraph isomorphism problem for P_k cannot have $n^{o(k)}$ time algorithms unless $\mathsf{FPT} = \mathsf{W}[1]$. We start by considering the polynomial:

$$I_{P_4,n} = \sum_{(p,q,r,s):p<s} y_p y_q y_r y_s x_{\{p,q\}} x_{\{q,r\}} x_{\{r,s\}} (1 - x_{\{p,r\}})(1 - x_{\{p,s\}})(1 - x_{\{q,s\}})$$

The polynomial $I_{P_4,n}(G)$ can be written as $\sum_X y_X$ where the summation is over all four vertex subsets of $V(G)$ that induces a P_4. Notice that all coefficents are 1 because there can be at most 1 induced-P_4 on any four vertex subset. By expanding terms of the form $1 - x_*$ in the above polynomial, we observe that we can rewrite $I_{P_4,n}$ as follows:

$$I_{P_4,n} = N_{P_4,n} - 4N_{C_4,n} - 2N_{K_3+e,n} + 6N_{K_4-e,n} + 12N_{K_4,n}$$

Since the coefficients in $I_{P_4,n}(G)$ are all 0 or 1, it is sufficient to check whether $I_{P_4,n}(G) \pmod 2$ is non-zero to test whether $I_{P_4,n}(G)$ is non-zero. From the above equation, we can see that $I_{P_4,n} = N_{P_4,n} \pmod 2$. Therefore, instead of working with $I_{P_4,n} \pmod 2$, we can work with $N_{P_4,n} \pmod 2$. We have already seen that we can use $Hom_{P_4,n}(G)$ to test whether $N_{P_4,n}(G)$ is non-zero. However, this is not sufficient to solve induced subgraph isomorphism. We want to detect whether $N_{P_4,n}(G)$ is non-zero modulo 2. Therefore, the multilinear terms of $Hom_{P_4,n}(G)$ has to be in one-to-one correspondence with the terms of $N_{P_4,n}(G)$. We have to divide the polynomial $Hom_{P_4,n}(G)$ by 2 before testing for the existence of multilinear terms modulo 2. However, since we are working over a field of characteristic 2, this division is not possible. We work around this problem by starting with $Hom_{P_4,n'}$ for n' slightly larger than n and we show that this enables the "division" by 2.

The reader may have observed that instead of the homomorphism polynomial, we could have taken any polynomial f for which the multilinear terms of $f(G)$ are in one-to-one correspondence with $N_{P_4,n}(G)$. This observation leads to the definition of a notion of reduction between polynomials. Informally, $f \preceq g$ if detecting multilinear terms in $f(G)$ is as easy as detecting multilinear terms in $g(G)$. Additionally, for the evaluation $f(G)$ to be well-defined, the polynomial f must have some special structure. We call such polynomials graph pattern polynomials.

On first glance, it appears hard to generalize this algorithm for P_4 to sparse pattern graphs on an arbitrary number of vertices (For example, P_k) because we have to argue about the coefficients of many N_* polynomials in the expansion. On the other hand, if we consider the pattern graph K_k, we have $I_{K_k} = Hom_{K_k}$. In this paper, we show that for many graph patterns sparser than K_k, the induced subgraph isomorphism problem is as easy as constructing arithmetic circuits for homomorphism polynomials for those patterns (or patterns that are only slightly denser).

4 Graph pattern polynomial families

We will consider polynomial families $f = (f_n)$ of the following form: Each f_n will be a polynomial in variables $y_1, \ldots, y_n$, the vertex variables, and variables $x_1, \ldots, x_{\binom{n}{2}}$, the edge variables, and at most linear in n number of additional variables.The degree of each f_n will usually be constant.

The (not necessarily induced) subgraph isomorphism polynomial family $N_H = (N_{H,n})_{n\geq 0}$ for a fixed pattern graph H on k vertices and ℓ edges is a family of multilinear polynomials of degree $k + \ell$. The n^{th} polynomial in the family, defined over the vertex set $[n]$, is the polynomial on $n + \binom{n}{2}$ variables given by (1):

$$N_{H,n} = \sum_{\phi: V(H) \overset{sub}{\mapsto} V(K_n)} y_{\phi(V(H))} x_{\phi(E(H))} \tag{1}$$

When context is clear, we will often omit the subscript n and simply write N_H. Given a (host) graph G on n vertices, we can substitute values for the edge variables of $N_{H,n}$ depending on the edges of G ($x_e = 1$ if $e \in E(G)$ and $x_e = 0$ otherwise) to obtain a polynomial $N_{H,n}(G)$ on the vertex variables. The monomials of this polynomial are in one-to-one correspondence with the H-subgraphs of G. i.e., a term $a y_{v_1} \cdots y_{v_k}$, where a is a positive integer, indicates that there are a subgraphs isomorphic to H in G on the vertices $v_1, \ldots, v_k$. Therefore, to detect if there is an H-subgraph in G, we only have to test whether $N_{H,n}(G)$ has a multilinear term.

The induced subgraph isomorphism polynomial family $I_H = (I_{H,n})_{n\geq 0}$ for a pattern graph H over the vertex set $[n]$ is defined in (2).

$$I_{H,n} = \sum_{\phi: V(H) \overset{ind}{\mapsto} V(K_n)} y_{\phi(V(H))} x_{\phi(E(H))} \prod_{e \notin E(H)} (1 - x_{\phi(e)}) \tag{2}$$

If we substitute the edge variables of $I_{H,n}$ using a host graph G on n vertices, then the monomials of the resulting polynomial $I_{H,n}(G)$ on the vertex variables are in one-to-one correspondence with the induced H-subgraphs of G. In particular, all monomials have coefficient 0 or 1 because there can be at most one induced copy of H on a set of k vertices. This implies that to test if there is an induced H-subgraph in G, we only have to test whether $I_{H,n}(G)$ has a multilinear term and we can even do this modulo p for any prime p. Also, note that I_H is simply $I_{\overline{H}}$ where all the edge variables x_e are replaced by $1 - x_e$.

The homomorphism polynomial family $Hom_H = (Hom_{H,n})_{n\geq 0}$ for pattern graph H over the vertex set $[n]$ is defined in (3).

$$Hom_{H,n} = \sum_{\phi: V(H) \overset{hom}{\mapsto} V(K_n)} \prod_{v \in V(H)} z_{v,\phi(v)} y_{\phi(v)} \prod_{e \in E(H)} x_{\phi(e)} \tag{3}$$

The variables $z_{a,v}$'s are called the *homomorphism variables*. They keep track how the vertices of H are mapped by the different homomorphisms in the summation. We note that the size of the arithmetic circuit computing $Hom_{H,n}$ is independent of the labelling chosen to define the homomorphism polynomial. The arithmetic circuit complexity of such homomorphism polynomials, with respect to properties of the pattern graph, has been studied in [4].

The induced subgraph isomorphism polynomial for any graph H and subgraph isomorphism polynomials for supergraphs of H are related as follows:

$$I_{H,n} = \sum_{H' \supseteq H} (-1)^{e(H')-e(H)} \#sub(H, H') N_{H',n} \tag{4}$$

Here $e(H)$ is the number of edges in H and $\#sub(H, H')$ is the number of times H appears as a subgraph in H'. The sum is taken over all supergraphs H' of H having the same vertex set as H. Equation 4 is used by Curticapean, Dell, and Marx [2] in the context of counting subgraph isomorphisms.

For any fixed pattern graph H, the degree of polynomial families N_H, I_H, and Hom_H are bounded by a constant depending only on the size of H. Such polynomial families are called constant-degree polynomial families.

▶ **Definition 4.1.** A constant-degree polynomial family $f = (f_n)$ is called a *graph pattern* polynomial family if the n^{th} polynomial in the family has n vertex variables, $\binom{n}{2}$ edge variables, and at most cn other variables, where c is a constant, and every non-multilinear term of f_n has at least one non-edge variable of degree greater than 1.

It is easy to verify that I_H, N_H, and Hom_H are all graph pattern polynomial families. For a graph pattern polynomial f, we denote by $f(G)$ the polynomial obtained by substituting $x_e = 0$ if $e \notin E(G)$ and $x_e = 1$ if $e \in E(G)$ for all edge variables x_e. Note that for any graph pattern polynomial f, we have $ML(f(G)) = ML(f)(G)$. This is because any non-multilinear term in f has to remain non-multilinear or become 0 after this substitution.

▶ **Definition 4.2.**

1. A constant degree polynomial family $f = (f_n)$ has circuits of size $s(n)$ if there is a sequence of arithmetic circuits (C_n) such that C_n computes f_n and has size at most $s(n)$.
2. f has uniform $s(n)$-size circuits, if on input n, we can construct C_n in time $O(s(n))$ on a Word-RAM.[3]

We now define a notion of reducibility among graph pattern polynomials. Informally, if $f \preceq g$, then we detecting whether $f_n(G)$ has a multilinear term is as easy as constructing an arithmetic circuit for g_n for all n. First, we define a notion of substitution families that preserves the semantic structure of graph pattern polynomials.

▶ **Definition 4.3.** A *substitution family* $\sigma = (\sigma_n)$ is a family of mappings

$$\sigma_n : \{y_1, \ldots, y_n, x_1, \ldots, x_{\binom{n}{2}}, u_1, \ldots, u_{m(n)}\} \to K[y_1, \ldots, y_{n'}, x_1, \ldots, x_{\binom{n'}{2}}, v_1, \ldots, v_{r(n)}]$$

mapping variables to polynomials such that:

1. σ maps vertex variables to constant-degree monomials containing one or more vertex variables or other variables, and no edge variables.
2. σ maps edge variables to polynomials with constant-size circuits containing at most one edge variable and no vertex variables.
3. σ maps other variables to constant-degree monomials containing no vertex or edge variables and at least one other variable.

σ_n naturally extends to $K[y_1, \ldots, y_n, x_1, \ldots, x_{\binom{n}{2}}, u_1, \ldots, u_{m(n)}]$.

For the reduction to be useful in deriving algorithms, the substitution has to be easily computable. This leads us to the following definition.

▶ **Definition 4.4.** A substitution family $\sigma = (\sigma_n)$ is *constant-time computable* if given n and a variable z in the domain of σ_n, we can compute $\sigma_n(z)$ in constant-time on a Word-RAM. (Note that an encoding of any z fits into on cell of memory.)

[3] Since we are dealing with fine-grained complexity, we have to be precise with the encoding of the circuit. We assume an encoding such that evaluating the circuit is linear time and substituting for variables with polynomials represented by circuits is constant-time.

Finally, we define our notion of reduction.

▶ **Definition 4.5.** Let $f = (f_n)$ and $g = (g_n)$ be graph pattern polynomial families. Then f is reducible to g, denoted $f \preceq g$, via a constant time computable substitution family $\sigma = (\sigma_n)$ if for all n there is an $m = O(n)$ and $q = O(1)$ such that

1. $\sigma_m(ML(g_m))$ is a graph pattern polynomial and
2. $ML(\sigma_m(g_m)) = v_{[q]}ML(f_n)$. (Recall that $v_{[q]} = v_1 \cdots v_q$.)

For any prime p, we say that $f \preceq g \pmod p$ if there exists an $f' = f \pmod p$ such that $f' \preceq g$.

Property 1 of Definition 4.5 and Properties 1 and 3 of Definition 4.3 imply that $\sigma_m(g_m)$ is a graph pattern polynomial because Properties 1 and 3 of Definition 4.3 ensure that non-multilinear terms remain so after the substitution. It is easy to see that $\preceq$ is reflexive via the identity substitution. We can also assume w.l.o.g. that the variables $v_1, \ldots, v_q$ are fresh variables introduced by the substitution family σ.

What is the difference between $\sigma_m(ML(g_m))$ and $ML(\sigma_m(g_m))$ in the Definition 4.5? Every monomial in $ML(\sigma_m(g_m))$ also appears in $\sigma_m(ML(g_m))$, however, the latter may contain further monomials that are not multilinear.

It is easy to see that $\preceq$ is reflexive via the identity substitution. It can be shown that $\preceq$ is transitive by composing substitutions.

We conclude this section by mentioning how to obtain efficient algorithms using $\preceq$. Efficient algorithms are known (See [10]) for detecting multilinear terms of *small* degree with non-zero coefficient modulo primes.

▶ **Theorem 4.6.** *Let k be any constant and let p be any prime. Given an arithmetic circuit of size s, there is a randomized, one-sided error $O(s)$-time algorithm to detect whether the polynomial computed by the circuit has a multilinear term of degree at most k with non-zero modulo p coefficient.*

An important algorithmic consequence of reducibility is stated in Proposition 4.7. This proposition is used to derive algorithms for induced subgraph isomorphism problems in this paper.

▶ **Proposition 4.7.** *Let p be any prime. Let f and g be graph pattern polynomial families. Let $s(n)$ be a polynomially-bounded function. If $f \preceq g \pmod p$ and g has size uniform $s(n)$-size arithmetic circuits, then we can test whether $f_n(G)$ has a multilinear term with non-zero coefficient modulo p in $O(s(n))$ (randomized one-sided error) time for any n-vertex graph G.*

5 Pattern graphs easier than cliques

In this section, we describe a family H_{3k} of pattern graphs such that the induced subgraph isomorphism problem for H_{3k} has an $O(n^{\omega(k,k-1,k)})$ time algorithm when $k = 2^\ell, \ell \geq 1$. Note that for the currently known best algorithms for fast matrix multiplication, we have $\omega(k, k-1, k) < k\omega$. Therefore, these pattern graphs are strictly easier to detect than cliques.

The pattern graph H_{3k} is defined on $3k$ vertices and we consider the canonical labelling of H_{3k} where there is a $(3k-1)$-clique on vertices $\{1, \ldots, 3k-1\}$ and the vertex $3k$ is adjacent to the vertices $\{1, \ldots, 2k-1\}$.

▶ **Lemma 5.1.** $I_{H_{3k}} = N_{H_{3k}} \pmod 2$ *when* $k = 2^\ell, \ell \geq 1$

Proof. We show that the number of times H_{3k} is contained in any of its proper supergraphs is even if k is a power of 2. The graph K_{3k} contains $3k\binom{3k-1}{2k-1}$ copies of H_{3k}. This number is even when k is even. The graph $K_{3k} - e$ contains $2\binom{3k-2}{2k-1}$ copies of H_{3k}. This number is always even. The remaining proper supergraphs of H_{3k} are the graphs $K_{3k-1} + (2k+i)e$, i.e., a $(3k-1)$-clique with $2k+i$ edges to a single vertex, for $0 \leq i < k-2$. There are $m_i = \binom{2k+i}{2k-1}$ copies of the graph H_{3k} in these supergraphs. We observe that the numbers m_i are even when $k = 2^\ell, \ell \geq 1$ by Lucas' theorem. Lucas' theorem states that $\binom{p}{q}$ is even if and only if in the binary representation of p and q, there exists some bit position i such that $q_i = 1$ and $p_i = 0$. To see why m_i is even, observe that in the binary representation of $2k-1$, all bits 0 through ℓ are 1 and in the binary representation of $2k+i, 0 \leq i < k-2$, at least one of those bits is 0. ◀

▶ **Lemma 5.2.** $N_{H_{3k}} \preceq Hom_{H_{3k}}$

Proof. We start with $Hom_{H_{3k}}$ over the vertex set $[n] \times [3k]$ and apply the following substitution.

$$\sigma(z_{a,(v,a)}) = z_a \tag{1}$$
$$\sigma(z_{a,(v,b)}) = z_a^2, a \neq b \tag{2}$$
$$\sigma(y_{(v,a)}) = y_v \tag{3}$$
$$\sigma(x_{(u,a),(v,b)}) = 0, \text{if } a,b \in \{1,\ldots,2k-1\} \text{ and } a < b \text{ and } u > v \tag{4}$$
$$\sigma(x_{(u,a),(v,b)}) = 0, \text{if } a,b \in \{2k,\ldots,3k-1\} \text{ and } a < b \text{ and } u > v \tag{5}$$
$$\sigma(x_{(u,a),(v,b)}) = x_{\{u,v\}}, \text{otherwise} \tag{6}$$

Rule 3 ensures that in any surviving monomial, all vertices have distinct labels. Rule 4 ensures that the vertices coloured $1,\ldots,2k-1$ are in increasing order and Rule 5 ensures that the vertices coloured $2k,\ldots,3k-1$ are in increasing order.

Consider an H_{3k} labelled using $[n]$ where the vertices in the $(3k-1)$-clique are labelled $v_1,\ldots,v_{3k-1}$ and the remaining vertex is labelled v_{3k} which is connected to $v_1 < \ldots < v_{2k-1}$. Also, $v_{2k} < \ldots < v_{3k-1}$. We claim that the monomial corresponding to this labelled H_{3k} (say m) is uniquely generated by the monomial $m' = \prod_{1\leq i\leq 3k} z_{i,(v_i,i)} w$ in $Hom_{H_{3k}}$. Note that the vertices and edges in the image of the homomorphism is determined by the map $i \mapsto (v_i, i)$. The monomial w is simply the product of these vertex and edge variables. It is easy to see that this monomial yields the required monomial under the above substitution. The uniqueness is proved as follows: observe that in any monomial m'' in $Hom_{H_{3k}}$ that generates m, the vertex coloured $3k$ must be v_{3k}. This implies that the vertices coloured $1,\ldots,2k-1$ must be the set $\{v_1,\ldots,v_{2k-1}\}$. Rule 4 ensures that vertex coloured i must be v_i for $1 \leq i \leq 2k-1$. Similarly, the vertices coloured $2k,\ldots,3k-1$ must be the set $\{v_{2k},\ldots,v_{3k-1}\}$ and Rule 5 ensures that vertex coloured i must be v_i for $2k \leq i \leq 3k-1$ as well. But then the monomials m' and m'' are the same. ◀

▶ **Lemma 5.3.** *$Hom_{H_{3k}}$ can be computed by arithmetic circuits of size $O(n^{\omega(k,k-1,k)})$ for $k > 1$.*

Proof. Consider H_{3k} labelled as before. We define the sets $S_{1,k,2k,3k-1} = \{1,\ldots,k,2k\ldots,3k-1\}$, $S_{k+1,3k-1} = \{k+1,\ldots,3k-1\}$, $S_{k+1,2k-1} = \{k+1,\ldots,2k-1\}$, and $S_{1,2k-1} = \{1,\ldots,2k-1\}$. We also define the tuples $V_{1,k} = (v_1,\ldots,v_k)$, $V_{2k,3k-1} = (v_{2k},\ldots,v_{3k-1})$, and $V_{k+1,2k-1} = (v_{k+1},\ldots,v_{2k-1})$ for any set v_i of $3k-1$ distinct vertex labels. The algorithm

also uses the matrices defined below. The dimensions of each matrix are specified as the superscript. All other entries of the matrix are 0. Notice that all entries are constant-sized monomials.

$$A^{n^k \times n^k}_{V_{1,k},V_{2k,3k-1}} = \left(\prod_{i \in S_{1,k,2k,3k-1}} z_{i,v_i} y_{v_i}\right)\left(\prod_{\substack{i,j \in S_{1,k,2k,3k-1} \\ i \neq j}} x_{\{v_i,v_j\}}\right)$$

$$B^{n^k \times n^{k-1}}_{V_{2k,3k-1},V_{k+1,2k-1}} = \left(\prod_{i \in S_{k+1,2k-1}} z_{i,v_i} y_{v_i}\right)\left(\prod_{\substack{i \in S_{k+1,3k-1} \\ j \in S_{k+1,2k-1} \\ i \neq j}} x_{\{v_i,v_j\}}\right)$$

$$C^{n^{k-1} \times n^k}_{V_{k+1,2k-1},V_{1,k}} = x_{\{(v_i,i)_{i \in S_{1,2k-1}}\}} \prod_{\substack{i \in S_{k+1,2k-1} \\ j \in [k] \\ i \neq j}} x_{\{v_i,v_j\}}$$

$$D^{n^k \times n}_{V_{1,k},v_{3k}} = z_{3k,v_{3k}} y_{v_{3k}} \prod_{i \in [k]} x_{\{v_i,v_{3k}\}}$$

$$E^{n \times n^{k-1}}_{v_{3k},V_{k+1,2k-1}} = \prod_{i \in S_{k+1,2k-1}} x_{\{v_i,v_{3k}\}}$$

Compute the matrix products ABC and DE. Replace the n^{2k-1} variables $x_{\{(v_i,i)_{i \in S_3}\}}$ with $(DE)_{V_{1,k},V_{k+1,2k-1}}$. The required polynomial is then just

$$Hom_{H_{3k}} = \sum_{(v_1,\ldots,v_k)} (ABC)_{(v_1,\ldots,v_k),(v_1,\ldots,v_k)}$$

Consider a homomorphism of H_{3k} defined as $\phi : i \mapsto u_i$. The monomial corresponding to this homomorphism is uniquely generated as follows. Let U_* be defined similarly to the tuples V_*. Set $v_i = u_i$ for $i \in [k]$ in the summation and consider the monomial generated by the product $A_{U_{1,k},U_{2k,3k-1}} B_{U_{2k,3k-1},U_{k+1,2k-1}} C_{U_{k+1,2k-1},U_{1,k}}$ after replacing the variable $x_{\{(u_i,i)_{i \in S_3}\}}$ by $(DE)_{U_{1,k},U_{k+1,2k-1}}$ taking the monomial $D_{U_{1,k},u_{3k}} E_{u_{3k},U_{k+1,2k-1}}$ from that entry. It is easy to verify that this generates the required monomial. For uniqueness, observe that this is the only way to generate the required product of the homomorphism variables.

Computing ABC can be done using $O(n^{\omega(k,k-1,k)})$ size circuits. Computing DE can be done using $O(n^{\omega(k,1,k-1)})$ size circuits. The top level sum contributes $O(n^k)$ gates. This proves the lemma. ◀

We conclude this section by stating our main theorem.

▶ **Theorem 5.4.** *The induced subgraph isomorphism problem for H_{3k} has an $O(n^{\omega(k,k-1,k)})$ time algorithm when $k = 2^\ell, \ell \geq 1$.*

6 Lower Bounds for Pattern Graphs with Cliques

Since we can obtain algorithms for induced subgraph isomorphism problems that match the known best algorithms using reductions between graph pattern polynomials, we can interpret the reduction $f \preceq g$ as evidence that detecting the graph pattern corresponding to g is harder than detecting f. It is known that the induced subgraph isomorphism problem for P_{2k} is harder than that for K_k. In general, one would think that detecting any graph H

that contains K_k as a subgraph would be at least as hard as detecting K_k. However, this is known only when H has a K_k that is vertex disjoint from all other K_k in H. The following theorem shows that we can drop this restriction when working with pattern polynomials.

▶ **Theorem 6.1.** *If H contains a k-clique or a k-independent set, then $I_{K_k} \preceq I_H$.*

Proof. We will prove the statement when H contains a k-clique. The other part follows because if H contains a k-independent set, then the graph $\overline{H}$ contains a k-clique and $I_{K_k} \preceq I_{\overline{H}} \preceq I_H$.

Fix a labelling of H where the vertices of a k-clique are labelled using $[k]$ and the remaining vertices are labelled $k+1, \ldots, k+\ell$. Consider the polynomial I_H over the vertex set $([n] \times [k]) \cup \{(n+i, k+i) : 1 \leq i \leq \ell\}$ and apply the following substitution.

$$\sigma(y_{(i,p)}) = \begin{cases} y_i u_p & \text{if } i \in [n] \text{ and } p \in [k] \\ u_p & \text{otherwise} \end{cases} \tag{1}$$

$$\sigma(x_{\{(i_1,p_1),(i_2,p_2)\}}) = \begin{cases} x_{\{i_1,i_2\}} & \text{if } \{p_1,p_2\} \in E(K_k) \text{ and } p_1 < p_2 \text{ and } i_1 < i_2 \\ 1 & \text{if } \{p_1,p_2\} \in E(H) \setminus E(K_k) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Consider a k-clique on the vertices $i_1, \ldots, i_k \in [n]$ on an n-vertex graph where $i_1 < \cdots < i_k$. The monomial in I_{K_k} corresponding to this clique is generated uniquely from the monomial $y_{(i_1,1)} \cdots y_{(i_k,k)} \prod_i y_{(n+i,k+i)} x_{\{(i_1,1),(i_2,2)\}} \cdots x_{\{(i_{k-1},k-1),(i_k,k)\}} w$ in I_H, where w is the product of all edge variables corresponding to edges in H but not in K_k. Note that Rules 1 and 2 ensure that in any surviving monomial, the labels and colours of all vertices are distinct and the colours of the edges must be the same as $E(H)$. The product w is determined by $i_1, \ldots, i_k$. This proves that $ML(\sigma(I_H)) = u_{[k+\ell]} ML(I_{K_k})$. It is easy to verify that the substitution satisfies the other properties. ◀

Using reductions between graph pattern polynomial families, it is possible to give evidence for many "natural" relative hardness results. We see these hardness results as showing the limitations of current methods for solving induced subgraph isomorphism problems.

7 Discussion

Are there patterns other than the pattern in Theorem 5.4 for which we can use homomorphism polynomials of graphs sparser than K_k for solving the induced subgraph isomorphism problem? In the full version of this paper, we show that we can obtain better algorithms for paths and cycles using our method. More specifically, we show that the induced subgraph isomorphism problems for P_5 and C_5 can be done in $O(n^\omega)$ time which is optimal assuming the optimality of triangle detection. We also show how to speed up algorithms for P_k and C_k when $k \leq 9$.

An interesting class of algorithms for induced subgraph isomorphism problems are the so called combinatorial algorithms – algorithms that do not use fast matrix multiplication. The best combinatorial algorithm known for k-cliques is the trivial $O(n^k)$ time algorithm. Contrary to general algorithms, we know that many patterns have improved combinatorial algorithms. For example, Virginia Williams [12] showed that there is a $O(n^{k-1})$ time combinatorial algorithm for the induced subgraph isomorphism problem for $K_k - e$. In fact, we show that ,from existing results, one can obtain combinatorial algorithms running in time $O(n^{k-1})$ for all patterns except K_k and I_k. Furthermore, for P_k and C_k we show that we can obtain new combinatorial algorithms running in time $O(n^{k-2})$.

In the full version of the paper, we show that the complexity of many pattern detection and counting problems can be linked to the circuit complexity of homomorphism polynomials for $K_k - e$. We show that if there are $O(n^{f(k)})$ size circuits for Hom_{K_k-e}, then:

1. The induced subgraph isomorphism problem for any k-vertex pattern other than K_k, I_k can be solved in $O(n^{f(k)})$ time. This shows that the induced subgraph isomorphism problem for any k-vertex pattern has a $O(n^{k-1})$ time combinatorial algorithm. This also shows that when $k \leq 9$, all patterns other than K_k, I_k have faster algorithms.
2. The number of subgraphs isomorphic to any k-vertex pattern can be counted in $O(n^{f(k)})$ time.
3. If we can count the number of induced subgraphs isomorphic to some k-vertex pattern in $O(t(n))$ time, then we can count all k-vertex patterns in $O(n^{f(k)} + t(n))$ time. This implies that for $k \leq 9$, improved algorithms for counting any k-vertex pattern will improve algorithms for counting k-cliques.

We also explain why homomorphism polynomials feature prominently in many results related to subgraph isomorphism. We show that for any pattern H, if there exists a family of polynomials such that $N_H \preceq f$, then the size complexity of Hom_H is at most the size complexity of f. Therefore, in a concrete sense, homomorphism polynomials are the best graph pattern polynomial families for subgraph isomorphism problems.

We also use reductions between graph pattern polynomial families similar to Theorem 6.1 to show many lower bounds that seem natural but are not known for general algorithms.

1. For almost all pattern graphs H, the induced subgraph isomorphism problem for H is harder than the subgraph isomorphism problem for H (A randomized reduction is to just randomly delete edges from the graph).
2. For almost all pattern graphs H, the subgraph isomorphism problem for H is easier than subgraph isomorphism problems for any supergraph of H.

Note however that we do not know whether these lower bounds imply general algorithmic hardness. But we believe that these results show the limitations of existing methods for solving subgraph isomorphism problems.

References

1 D. Corneil, Y. Perl, and L. Stewart. A Linear Recognition Algorithm for Cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985. doi:10.1137/0214065.

2 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 210–223. ACM, 2017. doi:10.1145/3055399.3055502.

3 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

4 Christian Engels. Dichotomy Theorems for Homomorphism Polynomials of Graph Classes. *J. Graph Algorithms Appl.*, 20(1):3–22, 2016.

5 Peter Floderus, Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Detecting and Counting Small Pattern Graphs. *SIAM J. Discrete Math.*, 29(3):1322–1339, 2015. doi:10.1137/140978211.

6 Peter Floderus, Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Induced subgraph isomorphism: Are some patterns substantially easier than others? *Theor. Comput. Sci.*, 605:119–128, 2015. doi:10.1016/j.tcs.2015.09.001.

7 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B. V. Raghavendra Rao. Faster algorithms for finding and counting subgraphs. *J. Comput. Syst. Sci.*, 78(3):698–706, 2012. doi:10.1016/j.jcss.2011.10.001.

8 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

9 Alon Itai and Michael Rodeh. Finding a Minimum Circuit in a Graph. *SIAM Journal on Computing*, 7(4):413–423, 1978. doi:10.1137/0207033.

10 Ioannis Koutis and Ryan Williams. LIMITS and applications of group algebras for parameterized problems. *ACM Trans. Algorithms*, 12(3):31:1–31:18, 2016. doi:10.1145/2885499.

11 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 026(2):415–419, 1985. URL: http://eudml.org/doc/17394.

12 Virginia Vassilevska. *Efficient Algorithms for Path Problems in Weighted Graphs*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, August 2008.

13 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

14 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding Four-Node Subgraphs in Triangle Time. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1671–1680. SIAM, 2015. doi:10.1137/1.9781611973730.111.