New Plain-Exponential Time Classes for Graph Homomorphism

Magnus Wahlström

wahl@mpi-inf.mpg.de
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. A homomorphism from a graph G to a graph H (in this paper, both simple, undirected graphs) is a mapping $f : V(G) \to V(H)$ such that if $uv \in E(G)$ then $f(u)f(v) \in E(H)$. The problem $\mathrm{Hom}(G, H)$ of deciding whether there is a homomorphism is NP-complete, and in fact the fastest known algorithm for the general case has a running time of $O^*\left(n(H)^{cn(G)}\right)$,[1] for a constant $0 < c < 1$. In this paper, we consider restrictions on the graphs G and H such that the problem can be solved in plain-exponential time, i.e. in time $O^*\left(c^{n(G)+n(H)}\right)$ for some constant c. Previous research has identified two such restrictions. If $H = K_k$ or contains K_k as a core (i.e. a homomorphically equivalent subgraph), then $\mathrm{Hom}(G, H)$ is the k-coloring problem, which can be solved in time $O^*\left(2^{n(G)}\right)$ (Björklund, Husfeldt, Koivisto); and if H has treewidth at most k, then $\mathrm{Hom}(G, H)$ can be solved in time $O^*\left((k+3)^{n(G)}\right)$ (Fomin, Heggernes, Kratsch, 2007). We extend these results to cases of bounded cliquewidth: if H has cliquewidth at most k, then we can count the number of homomorphisms from G to H in time $O^*\left((2k+1)^{\max(n(G),n(H))}\right)$, including the time for finding a k-expression for H. The result extends to deciding $\mathrm{Hom}(G, H)$ when H has a core with a k-expression, in this case with a somewhat worse running time.
If G has cliquewidth at most k, then a similar result holds, with a worse dependency on k: We are able to count $\mathrm{Hom}(G, H)$ in time roughly $O^*\left((2k+1)^{n(G)} + 2^{2kn(H)}\right)$, and this also extends to when G has a core of cliquewidth at most k with a similar running time.

1 Introduction

A homomorphism from a graph G to a graph H is a mapping of the vertices of G to the vertices of H that preserves adjacency (i.e. neighbours in G are mapped to neighbours in H); we write $G \to H$ if one exists. The graph homomorphism problem $\mathrm{Hom}(G, H)$, of deciding whether $G \to H$, occupies an interesting place in terms of exact time complexity; let us make an overview of what is known and not.

[1] The notation $O^*(\cdot)$ signifies that polynomial factors have been ignored

In the following, $\mathcal{G}$ and $\mathcal{H}$ are graph classes restricting the input: $\mathrm{Hom}(\mathcal{G}, \mathcal{H})$ is the class of problems $\mathrm{Hom}(G, H)$ for $G \in \mathcal{G}$, $H \in \mathcal{H}$. A dash (as in $\mathrm{Hom}(-, \mathcal{H})$) represents the class of all graphs, i.e. no restrictions. We are assuming that all graphs are simple and undirected. For more on the homomorphism problem in general, we refer to the book by Hell and Nešetřil [16]. Also, in addition to the decision and counting problems considered in this paper, optimization variants have been considered and studied in [14, 13, 12] and in [18].

1.1 Background

Beginning in some sense from the bottom, the polynomial cases of $\mathrm{Hom}(\mathcal{G}, -)$ and $\mathrm{Hom}(-, \mathcal{H})$ are completely known (assuming standard complexity theoretical assumptions). For $\mathrm{Hom}(-, \mathcal{H})$, there is a dichotomy due to Hell and Nešetřil [15]: $\mathrm{Hom}(-, \mathcal{H})$ is in P if every $H \in \mathcal{H}$ is bipartite, in which case the problem $\mathrm{Hom}(G, H)$ corresponds to checking whether G is bipartite, and NP-complete otherwise (including $\mathrm{Hom}(-, \{H\})$ for a single non-bipartite graph H). For example, $\mathrm{Hom}(-, \{K_3\})$ corresponds to 3-coloring.

For $\mathrm{Hom}(\mathcal{G}, -)$, there is a dichotomy due to Grohe [11]: $\mathrm{Hom}(\mathcal{G}, -)$ is in P if every $G \in \mathcal{G}$ either has bounded treewidth or is *homomorphically equivalent* to a graph of bounded treewidth (its *core* – see Section 2 for definitions). In every other case, the problem is W[1]-hard, thus not in P unless FPT=W[1] (the classes FPT and W[1] come from the field of parameterized complexity [6, 9]). In particular, $\mathrm{Hom}(\mathcal{G}, -)$ is trivially in P for every finite $\mathcal{G}$.

There may be other polynomial cases of $\mathrm{Hom}(\mathcal{G}, \mathcal{H})$ where both sides are restricted, but we are not aware of this having been studied.

Moving on, the most famous special cases of homomorphism correspond to restricting $\mathcal{G}$ or $\mathcal{H}$ to being cliques. Specifically, $\mathrm{Hom}(G, K_k)$ is the problem of finding a k-colouring of G (thus explaining why the graph homomorphism problem is also known as H-colouring). For this problem, algorithms were recently, famously, presented by Björklund, Husfeldt, and Koivisto [1] which solve it in time $O^*\left(2^{n(G)}\right)$ for all k (older results, with different methods, exist with running times $O^*\left(c^{n(G)}\right)$ for $c > 2.4$ [19, 7, 2]). On the other side, $\mathrm{Hom}(K_k, H)$ amounts to finding a k-clique in H, which can of course be done in time $O^*\left(2^{n(H)}\right)$ for all k. These two cases also correspond to two other lower bounds on the time complexity of graph homomorphism (from [10]):

- If $\mathrm{Hom}(G, H)$ could be solved in time $O\left(p(n(G)) \cdot f(n(H))\right)$ for a polynomial $p(n)$ and any function $f(n)$, then there would be a polynomial-time algorithm for the NP-complete k-colouring problems (implying P=NP).
- If $\mathrm{Hom}(G, H)$ could be solved in time $O\left(f(n(G)) \cdot p(n(H))\right)$ for a polynomial $p(n)$ and any function $f(n)$, then the k-clique problem would be in FPT, implying FPT=W[1].

However, there is still a lot of room between these lower bounds and the best upper bound, which is $O^*\left(n(H)^{cn(G)}\right)$ for a constant $c < 1$, using an algorithm by Williams [21]. In particular, as asked by Fomin, Heggernes, and Kratsch [10], can $\mathrm{Hom}(G, H)$ be solved in time $O^*\left(c^{n(G)+n(H)}\right)$ for a constant c?

This question is still open, but for a generalisation of graph homomorphism we know (again under a complexity theoretical assumption) that the answer is negative. A $(d, 2)$-*CSP* instance consists of n variables, with d possible values, and arbitrary binary constraints (i.e. constraints on the values of pairs of variables). Thus, $\mathrm{Hom}(G, H)$ is an instance of $(n(H), 2)$-CSP. Traxler [20] recently showed that assuming ETH (i.e. unless 3-SAT has subexponential time algorithms [17]), $(d, 2)$-CSP requires time $\Omega^*(d^{cn})$ for some constant $c > 0$. [2]

1.2 Our contributions

In this paper, we focus on restricted cases of graph homomorphism for which we can show the existence of plain-exponential time algorithms, i.e. cases of $\mathrm{Hom}(-, \mathcal{H})$ for which we can show algorithms running in time $O^*\left(c^{n(G)}\right)$ (and to a lesser extent the converse for $\mathrm{Hom}(\mathcal{G}, -)$). We are aware of two previous results of this type. One is the k-coloring problems, and instances equivalent to them (if $\chi(H) = \omega(H) = k$, then $\mathrm{Hom}(G, H)$ is again equivalent to k-coloring). The other is the case of bounded treewidth: if the treewidth of H is at most k, then Fomin, Heggernes, and Kratsch showed an algorithm for deciding $\mathrm{Hom}(G, H)$ with a running time of $O^*\left((k+3)^{n(G)}\right)$ [10].

We connect the question to the concept of cliquewidth [4, 5]: we show that if H has cliquewidth at most k (written $\mathrm{cwd}(H) \leq k$), then $\mathrm{Hom}(G, H)$ can be counted in time $O^*\left((2k+1)^{\max(n(G),n(H))}\right)$, and if the core of H has cliquewidth at most k, then $\mathrm{Hom}(G, H)$ can be decided in time $(O(k))^{\max(n(G),n(H))}$. Both results include the time for finding a k-expression. As far as the existence of plain-exponential time algorithms goes, this extends both previous results, as cliques have cliquewidth 2, and the cliquewidth of a graph is bounded by a function of its treewidth [5].

For restrictions $\mathrm{Hom}(\mathcal{G}, -)$, we show a plain-exponential algorithm for the same case: if $\mathrm{cwd}(G) \leq k$ for every $G \in \mathcal{G}$, then $\mathrm{Hom}(\mathcal{G}, -)$ can be counted in time $O^*\left(c_k^{\max(n(G),n(H))}\right)$, where the constant c_k depends on k. Here, the dependency is worse: we get $c_k = 2^{2k}$.

The organisation of the paper is as follows: Section 2 contains preliminaries, and a simple exponential-time algorithm for finding k-expressions for bounded k. Section 3 presents our result for $\mathrm{Hom}(-, \mathcal{H})$, Section 4 presents our result for $\mathrm{Hom}(\mathcal{G}, -)$, and Section 5 contains some conclusions and further questions.

[2] Let us also remark that [20] excludes the existence of algorithms with a running time of $O^*\left(c^{d+n}\right)$ for $(d, 2)$-CSP, as performing the variable reduction of [20] with a target domain size of $\log n$ would produce a running time like $d^{O(n/\log\log n)}$ for $(d, 2)$-CSP, which would count as subexponential.

2 Preliminaries

2.1 Graph Homomorphism

Definition 1. *Given graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, a* homomorphism *from G to H is a mapping $f : V_G \to V_H$ such that for any edge uv in G, $f(u)f(v)$ is an edge in H. If one exists, we say that G is homomorphic to H, written $G \to H$. The graph homomorphism problem* $\mathrm{Hom}(G, H)$ *is the problem of deciding whether $G \to H$. For classes of graphs $\mathcal{G}$ and $\mathcal{H}$,* $\mathrm{Hom}(\mathcal{G}, \mathcal{H})$ *is the graph homomorphism problem restricted to inputs $G \in \mathcal{G}$, $H \in \mathcal{H}$. A graph class given as $-$ (e.g.* $\mathrm{Hom}(-, \mathcal{H})$ *and* $\mathrm{Hom}(\mathcal{G}, -)$*) means the class of all graphs.*

We need a few basic facts about homomorphisms (see [16] for more).

Proposition 1. *Homomorphisms compose: if $G \to H$ and $H \to H'$, then $G \to H'$. Also, $G \to K_k$ if and only if G is k-colourable.*

Definition 2. *G,H are* homomorphically equivalent *if $G \to H$ and $H \to G$. A* core *is a graph G which is not homomorphically equivalent to any proper subgraph of itself; a core of a graph G is a minimal subgraph of G which is homomorphically equivalent to G.*

Proposition 2. *All cores of a graph G are isomorphic, thus we speak of* the *core of G. Also, the core of G is an induced subgraph of G.*

2.2 Cliquewidth

Definition 3. *A* k-expression *for a graph G is an expression using k different labels that constructs G using the following steps [4, 5]:*

- $\cdot_i$*: Construct a graph with one vertex, giving the vertex the label i.*
- $\rho_{i \to j}(G)$*: Relabel all vertices of G with label i to label j.*
- $\eta_{ij}(G)$*, for $i \neq j$: Add edges between all vertices of labels i and j in G, i.e. add edges uv for any vertices u, v where u has label i and v has label j.*
- $G_1 \oplus G_2$*: Create the disjoint union of G_1 and G_2.*

A graph G has cliquewidth *k, denoted* $\mathrm{cwd}(G) = k$*, if k is the smallest number such that there exists a k-expression for G. A k-expression is* linear *if one side of every disjoint union operation is a single vertex (e.g. $H' \oplus \cdot_i$ rather than a general $H' \oplus H''$). The* linear cliquewidth *of H is the smallest number such that there exists a linear k-expression for H.*

Example 1. A 2-expression for K_4 is $\eta_{12}(\rho_{2\to1}(\eta_{12}(\rho_{2\to1}(\eta_{12}(\cdot_1 \oplus \cdot_2)) \oplus \cdot_2)) \oplus \cdot_2)$. This expression is linear. In general, all cliques K_k with $k \geq 2$ have linear cliquewidth and cliquewidth both 2.

Definition 4. *We say that a k-expression, constructing a graph G, is in* standard form *if for every operation $G_1 \oplus G_2$, $G_i = G[V(G_i)]$ for $i = 1, 2$, i.e. all edges in G between vertices of G_1 (and G_2) have already been created before the $\oplus$ operation. We will throughout this paper assume that all k-expressions are in standard form. Note that a k-expression can be easily transformed to an equivalent k-expression in standard form (by recursively adding η-operations to the expressions for the graphs G_1, G_2 on both sides of a union operation $G_1 \oplus G_2$), and that the length of the result is still polynomial in n.*

Proposition 3 ([5]). *If G is a graph and $G[S]$ for $S \subseteq V(G)$ an induced subgraph, then* $\mathrm{cwd}(G[S]) \leq \mathrm{cwd}(G)$.

The problem of finding the cliquewidth of a graph is NP-hard in general [8], and the complexity of finding a k-expression for a fixed $k > 3$ is unknown – it is not even known whether it is polynomial. (A graph has a 1-expression only if it has no edges, and a 2-expression if and only if it has no induced P_4 (path on four vertices); for $k = 3$ a polynomial-time algorithm exists [3].) However, in this paper, we will not need polynomial-time constructions: since our algorithms take $(O(k))^n$ time, we can afford to use an exact algorithm with the same behaviour for finding a k-expression. This algorithm is given next.

2.3 Constructing a k-Expression for Bounded k

We now show how to find a k-expression for a graph G, provided that we are allowed to use time $(O(k))^n$. We want to stress that we have not made any particular effort to minimize the running time of this construction; at the moment, it is fast enough that it is not the bottleneck of our homomorphism algorithm, which is all we need.

Theorem 1. *If* $\mathrm{cwd}(G) \leq k$, *then a k-expression for G can be found in time* $O^*\big((2k+1)^{n(G)}\big)$.

Proof. We will create a table of size $(k+1)^{n(G)}$ containing a k-expression for each partitioning $S_1, \ldots, S_k$ of each vertex set $S \subseteq V(G)$ (with the vertices of label i being S_i). Let the sets S be considered in order of increasing cardinality. For the base cases $|S| = 1$, the expression is trivial; if $|S| > 1$, then the corresponding expression must contain a union operation $G' \oplus G''$, for graphs $G' = G[S']$ and $G'' = G[S'']$ corresponding to a split of S into vertex sets S', S'', so G' and G'' are graphs for which we already know k-expressions. We will create the expressions for all partitionings of S in two phases, first going through all expressions without relabelling operations ρ (e.g. for creating partitionings of S with k non-empty label classes), then using this data to find expressions which do contain relabelling operations. Thus, first split S all $(2k)^{|S|}$ ways into partitionings of S' and S'', verify that there are k-expressions for these partitionings for the resulting G' and G'', and that the existence of edges connecting G' and G'' follows the pattern of vertex labels. If so, register a k-expression for the

resulting partitioning of S (keeping only one expression per partitioning). This phase takes $O^*\big((2k)^{|S|}\big)$ time.

In the second phase, go through the possible partitionings of S in order of decreasing number of non-empty partitions, and consider for each partitioning of S each possible single relabelling operation $\rho_{i\to j}$ in turn. If appending $\rho_{i\to j}$ to a k-expression leads to a k-expression for a partitioning of S that previously did not have one, register the new k-expression. This phase uses only polynomial time for each partitioning of S, thus time $O^*\big(k^{|S|}\big)$. Also note that every partitioning of S for which a k-expression is possible will have received a k-expression at the end of this process.

Ignoring polynomial factors, the whole algorithm runs in time

$$\sum_{S\subseteq V}(2k)^{|S|}=\sum_{i=0}^{n}\binom{n}{i}(2k)^i=(2k+1)^n$$

and creates a k-expression for G. □

The case of linear k-expressions is easier:

Theorem 2. *If G has a linear k-expression, then one can be found in time $O^*\big((k+2)^{n(G)}\big)$.*

Proof. The algorithm runs as the previous proof, except that in the first phase for each S, instead of $(2k)^{|S|}$ ways to split $G[S]$ into $G[S']\oplus G[S'']$, there are now only $(k+1)^{|S|}$ ways to split (k labels on one side of $\oplus$, plus one on the other). The second phase is then identical. □

3 Homomorphism to Graphs with Bounded Cliquewidth

In this section, we present our main result: $\mathrm{Hom}(-,\mathcal{H})$ can be counted in time $O^*\big((2k+1)^{\max(n(G),n(H))}\big)$ when every $H\in\mathcal{H}$ has cliquewidth at most k. Per Theorem 1, we can find a k-expression for a given H in this time; we start by showing how to use this expression to solve $\mathrm{Hom}(G,H)$, then consider some variations.

3.1 Homomorphism to a Graph with a k-Expression

We now show how to decide or count $\mathrm{Hom}(G,H)$ in time $O^*\big((2k+1)^{n(G)}\big)$ assuming that a k-expression for H has already been given. The method is a direct application of dynamic programming techniques.

Theorem 3. *Given a k-expression for H, $\mathrm{Hom}(G,H)$ can be counted in time $O^*\big((2k+1)^{n(G)}\big)$.*

Proof. Let the k-expression for H be h. The algorithm will work as a recursion over the disjoint union operations $\oplus$ of h through dynamic programming: for every operation $H' \oplus H''$, we will use caches for H' and H'' giving for each partitioning $S_1, \ldots, S_k$ of a set $S \subseteq V(G)$ the number of homomorphisms from $G[S]$ to H' (resp. to H'') mapping S_i to vertices of label i. We will call a k-expression *trivial* if it contains no disjoint union operation, i.e. the graph H is a single vertex, assume with label i. These cases form the base cases of the recursion, and the corresponding cache is easily constructed (if any $S_j \neq \emptyset$ for $j \neq i$, or if $E(G[S]) \neq \emptyset$ then the answer is 0, otherwise 1).

Assuming that h is not trivial, let h be a sequence of non-$\oplus$ operations, referred to as the *head* of h, applied to some expression $h' \oplus h''$ (where h' and h'' are k-expressions generating the graphs H' resp. H'' with vertex sets V'_H resp. V''_H). As stated, we assume that all edges in H between vertices of H' are added by h' (and likewise for H''). Let there be caches for h' and h'' as described above. We can then create a cache for h that answers the same questions for $\mathrm{Hom}(G[S], H)$ for all possible partitionings $S_1, \ldots, S_k$ of each $S \subseteq V(G)$ by looping through all $(2k+1)^{n(G)}$ assignments of $V(G)$ to either not be included in S, or to one of the $2k$ combinations of vertex label and corresponding graph (H' or H'').

Specifically, for each such considered assignment of vertices, verify in polynomial time that the split of S into $S' = S'_1, \ldots, S'_k$ mapping to H' and $S'' = S''_1, \ldots, S''_k$ mapping to H'' does not break any edge (i.e. for each edge $u'u''$ in G such that u' is assigned to label i in H' and u'' to label j in H'', check that an η_{ij} operation exists in the head of h), query the caches for the corresponding numbers of homomorphisms $\mathrm{Hom}(G[S'], H')$ and $\mathrm{Hom}(G[S'''], H'')$ following the partitionings of S', S'', and if all tests pass, add the resulting number of homomorphisms to the right entry of the cache for h (remembering to account for relabelling operations ρ in the head of h).

As constructing the cache for a trivial expression $\mathrm{Hom}(G[S], \cdot_i)$ is trivial, and every further cache can be constructed from previous caches, by induction the cache corresponding to the outermost query $\mathrm{Hom}(G, H)$ can be created in time $O^*\big((2k+1)^{n(G)}\big)$, and the number of homomorphisms $G \to H$ can be calculated from this in time $O^*\big(k^{n(G)}\big)$. □

Remark 1. Variations of the problem can be solved using the same method; we sketch two here. First, a homomorphism $f : V(G) \to V(H)$ is *vertex-surjective* if $f^{-1}(v) \neq \emptyset$ for every $v \in V(H)$. We can make our method consider only vertex-surjective homomorphisms by enforcing the condition in the base cases $\cdot_i$; the property will then be maintained in the inductive steps.

Second, Gutin et al. [14] considered an optimization variant, where for every $u \in V(G)$ and $v \in V(H)$ there is an associated *cost* $c_v(u)$ of mapping u to v, and a homomorphism f minimizing $\sum_{u \in V(G)} c_{f(v)}(u)$ is sought. The property that the caches only contain information on minimum-cost homomorphisms for each query can be enforced in the base cases (where each suggested mapping has one fix cost only – the operations $\cdot_i$ are assumed to now also give the name of the new vertex), and maintained through the inductive steps (where the cost

of a homomorphism to H_0 built from $H' \oplus H''$ is the sum of the costs of the homomorphisms to H', H'').

Corollary 1. $\mathrm{Hom}(G, H)$ *can be counted in time* $O^*\big((2k+1)^{\max(n(G),n(H))}\big)$ *if the cliquewith of* H *is at most* k.

If we settle for linear k-expressions, the base $(2k+1)$ in the running time can be improved to $(k+2)$. The proof is omitted, as it is very close to Theorem 3.

Theorem 4. $\mathrm{Hom}(G, H)$ *when* H *has a linear* k*-expression can be counted in time* $O^*\big((k+2)^{\max(n(G),n(H))}\big)$.

3.2 Extension to Core of H

If H is itself not a core, then it may be that H has large cliquewidth but that the core H_C of H has bounded cliquewidth, such as the example that $\chi(H) = \omega(H)$ (in which case a clique of H is the core). For the decision case, we can handle these cases as well, as $G \to H$ if and only if $G \to H_C$, at the cost of a bigger base of the running time.

Theorem 5. *If a graph* G *has a core* G_C *with* $\mathrm{cwd}(G_C) \leq k$*, then* G_C *and a* k*-expression for* G_C *can be found in time* $O^*\big((2(2k+1))^{n(G)}\big)$.

Proof. Since $\mathrm{cwd}(G[S]) \leq \mathrm{cwd}(G)$ for induced subgraphs $G[S]$ of G, every induced subgraph of G_C will have a cliquewidth bounded by k. Thus, we can construct k-expressions for progressively larger induced subgraphs $G[S]$, as in Theorem 1, getting in time $O^*\big((2k+1)^{n(G)}\big)$ a list of k-expressions for those induced subgraphs $G[S]$ that have any. Then, for each such $G[S]$ we check $\mathrm{Hom}(G, G[S])$ using Theorem 3, using time bounded by $O^*\big((2(2k+1))^{n(G)}\big)$. If the core G_C has $\mathrm{cwd}(G_C) \leq k$, then eventually we will reach $G_C = G[S]$ and find a homomorphism, thus confirming $G[S] = G_C$. □

Corollary 2. $\mathrm{Hom}(G, H)$ *when the core of* H *has cliquewidth at most* k *can be decided in time* $O\left(k\right)^{\max(n(G),n(H))}$.

4 Left-Side Restrictions

Although our main focus is restrictions on $\mathcal{H}$, as such restrictions more commonly represent typical problem classes (in $\mathrm{Hom}(\mathcal{G}, \mathcal{H})$, and more generally in the case of relational homomorphisms, commonly $\mathcal{H}$ can be seen as representing a problem type, and $\mathcal{G}$ representing an instance of this problem), we show in this section that restricting the cliquewidth of $\mathcal{G}$ also leads to plain-exponential time cases, although with a rather worse dependency on the cliquewidth: if $\mathrm{cwd}(G) = k$, then we can solve $\mathrm{Hom}(G, H)$ in time $O^*\big(2^{2kn(H)}\big)$, plus the time for constructing a k-expression for G. Note, however, that simply converting the trivial running time of $O^*\big(n(H)^{n(G)}\big)$ to $O^*\big(n(H)^{f(k)}\big)$ would have lead to new polynomial cases, including a polynomial-time algorithm for k-clique, and so could not be expected.

Theorem 6. *If a k-expression for G is known, then* $\mathrm{Hom}(G, H)$ *can be counted in time* $O^*\big(2^{2kn(H)}\big)$.

Proof. The problem is once again solved through dynamic programming over the components of the k-expression, though this time, we need to store more information in the caches. Specifically, our cache for a graph G' or G'' occurring in an expression $G' \oplus G''$ will be used to give, for k not necessarily disjoint subsets V_i of $V(H)$, the number of homomorphisms from G' to H such that the targets in $V(H)$ of vertices of label i in G' are exactly V_i (i.e. homomorphisms f such that $V_i = \{f(u) \mid u \in V(G) \text{ has label } i\}$ for $i = 1, \ldots, k$). Once again, the caches for the base cases $\cdot_i$ are trivial (since G' is then a single vertex, it can be mapped to any single target in $V(H)$).

Thus, assume that we are working on a k-expression g_0 creating a graph G_0, consisting of a *head* of edge creations η and relabelling operations ρ, applied to the result of a disjoint union operation $G' \oplus G''$. Every pair of homomorphisms $f' : V(G') \to V(H)$ and $f'' : V(G'') \to V(H)$ compose into one mapping $f : V(G_0) \to V(H)$, and our task is to check whether f is a homomorphism, i.e. to verify that all edges created in the head of g_0 are mapped to edges in H. This can be done using the information in the caches: Assume that all edges between labels i and j are created in the head of g_0, and let V_i' resp. V_j'' be the targets of label i in G' resp. label j in G''. If there are $u \in V_i'$, $v \in V_j''$ such that uv is not an edge of H (in particular, if $u = v$), then some edge in G_0 is broken by f: Since V_i' and V_j'' are the exact targets, there are vertices $u' \in V_i'$ with $f(u') = u$ and $v'' \in V_j''$ with $f(v'') = v$, and an edge $u'v''$ in G_0 which is broken by f. The same check holds for label j in G' and label i in G'', and if both checks pass, then no edge between labels i and j are broken by f. Thus, for a particular combination of targets V_i' of labels of G' and V_i'' of labels of G'', we can decide using a polynomial number of polynomial-time checks whether the created mappings f would be homomorphisms or not.

Now the cache for G_0 can be assembled using queries to the caches for G', G'': With $2^{n(H)}$ options for each set of targets, the time for this step is bounded by $O^*\big((2^{n(H)})^{2k}\big) = O^*\big(2^{2kn(H)}\big)$. By induction, the cache for G can be created, and the total number of homomorphisms can be counted from this. □

Corollary 3. *If G has cliquewidth at most k, then* $\mathrm{Hom}(G, H)$ *can be counted in plain-exponential time* $O^*\big((2k+1)^{n(G)} + 2^{2kn(H)}\big)$. *Under the weaker assumption that the core of G has cliquewidth at most k,* $\mathrm{Hom}(G, H)$ *can be decided in time* $O^*\big((2(2k+1))^{n(G)} + 2^{2kn(H)}\big)$.

5 Conclusions

We have showed that homomorphisms from G to H can be counted in plain-exponential time $O^*\Big(c_k^{n(G)+n(H)}\Big)$ when either G or H has cliquewidth at most k (c_k depending on k). This unifies and extends previous plain-exponential time classes for the problem.

Nevertheless, there is a simple case of plain-exponential time for $\mathrm{Hom}(-, \mathcal{H})$ which our results do not cover: if the graphs $H \in \mathcal{H}$ have bounded degree $\Delta(H) \leq d$, then we can trivially solve the problem through branching in time $O^*\left(d^{n(H)}\right)$. Is there yet another case of plain-exponential time $\mathrm{Hom}(-, \mathcal{H})$ which covers both the cliquewidth and the bounded degree cases?

Also, we have to ask about homomorphism for directed graphs, and relational homomorphism in general [16, 11]. These problems cover the CSP problems, so we know that plain-exponential time algorithms are impossible in general [20]. Can we identify plain-exponential time classes for these problems?

References

1. A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion–exclusion. *SIAM J. Comput.* To appear; prelim. versions in FOCS'06.
2. J. M. Byskov. *Exact Algorithms for Graph Colouring and Exact Satisfiability.* PhD thesis, University of Aarhus, 2005.
3. D. G. Corneil, M. Habib, J.-M. Lanlignel, B. A. Reed, and U. Rotics. Polynomial time recognition of clique-width ≤ 3 graphs (extended abstract). In *LATIN'00*, pages 126–134, 2000.
4. B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *J. Comput. System Sci.*, 46:218–270, 1993.
5. B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77 – 114, 2000.
6. R. G. Downey and M. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.
7. D. Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms Appl.*, 7(2):131–140, 2003.
8. M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider. Clique-width minimization is NP-hard. In *STOC'06*, pages 354–362, 2006.
9. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.
10. F. V. Fomin, P. Heggernes, and D. Kratsch. Exact algorithms for graph homomorphisms. *Theory of Comput. Syst.*, 41(2):381 – 393, 2007.
11. M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), 2007.
12. G. Gutin, P. Hell, A. Rafiey, and A. Yeo. A dichotomy for minimum cost graph homomorphisms. *European J. Combin.*, 29:900–911, 2008.
13. G. Gutin, A. Rafiey, and A. Yeo. Minimum cost and list homomorphisms to semicomplete digraphs. *Discrete Applied Mathematics*, 154(6):890–897, 2006.
14. G. Gutin, A. Rafiey, A. Yeo, and M. Tso. Level of repair analysis and minimum cost homomorphisms of graphs. *Discrete Applied Mathematics*, 154(6):881–889, 2006.
15. P. Hell and J. Nešetřil. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990.
16. P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford University Press, 2004.
17. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.
18. P. Jonsson, G. Nordh, and J. Thapper. The maximum solution problem on graphs. In *MFCS'07*, pages 228–239, 2007.

19. E. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5:66–67, 1976.
20. P. Traxler. The time complexity of constraint satisfaction. In *IWPEC'08*, pages 190–201, 2008.
21. R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.