Exact Algorithms for Graph Homomorphisms*

Fedor V. Fomin† Pinar Heggernes† Dieter Kratsch‡

Abstract

Graph homomorphism, also called H-coloring, is a natural generalization of graph coloring: There is a homomorphism from a graph G to a complete graph on k vertices if and only if G is k-colorable. During the recent years the topic of exact (exponential-time) algorithms for NP-hard problems in general, and for graph coloring in particular, has led to extensive research. Consequently, it is natural to ask how the techniques developed for exact graph coloring algorithms can be extended to graph homomorphisms. By the celebrated result of Hell and Nešetřil, for each fixed simple graph H, deciding whether a given simple graph G has a homomorphism to H is polynomial-time solvable if H is a bipartite graph, and NP-complete otherwise.

The case where H is the cycle of length 5, is the first NP-hard case different from graph coloring. We show that for an odd integer $k \geq 5$, whether an input graph G with n vertices is homomorphic to the cycle of length k, can be decided in time $\min\{\binom{n}{\lfloor n/k \rfloor}, 2^{n/2}\} \cdot n^{\mathcal{O}(1)}$. We extend the results obtained for cycles, which are graphs of treewidth two, to graphs of bounded treewidth as follows: If H is of treewidth at most t, then whether input graph G with n vertices is homomorphic to H can be decided in time $(t+3)^n \cdot n^{\mathcal{O}(1)}$.

1 Introduction

Given two undirected graphs G and H, a *homomorphism* from G to H is a mapping $\varphi : V(G) \longrightarrow V(H)$ that satisfies the following: $\{x, y\} \in E(G) \Longrightarrow \{\varphi(x), \varphi(y)\} \in E(H)$ for every $x, y \in V(G)$. When there is a homomorphism from G to H we say that G is *homomorphic* to H. The problem of deciding whether graph G is homomorphic to graph H is called HOM(G, H). This problem can be seen as labeling, or coloring, the vertices of G by the vertices of H, and this is why it is often also called the H-coloring problem. Note that for the

*This work is supported by the AURORA mobility programme for research collaboration between France and Norway. A preliminary version was presented at the 15th International Symposium on Fundamentals of Computation Theory - FCT 2005.

†Department of Informatics, University of Bergen, N-5020 Bergen, Norway. Emails: `fomin@ii.uib.no` and `pinar@ii.uib.no`

‡Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine, 57045 Metz Cedex 01, France. Email: `kratsch@univ-metz.fr`

special case when H is a complete graph on k vertices, G is homomorphic to H if and only if the chromatic number of G is at most k. We refer to the recent book of Hell and Nešetřil [13] for a thorough introduction to the topic.

For graph classes $\mathcal{G}$ and $\mathcal{H}$ we denote by $\mathrm{HOM}(\mathcal{G}, \mathcal{H})$ the problem of deciding whether a given graph $G \in \mathcal{G}$ is homomorphic to a given graph $H \in \mathcal{H}$. If $\mathcal{G}$ consists of a single graph G, or $\mathcal{H}$ consists of a single graph H, we use G and H instead of $\{G\}$ and $\{H\}$, for simplicity of notation. If $\mathcal{G}$ or $\mathcal{H}$ is the class of all graphs then they are denoted by the placeholder '_'. The computational complexity of graph homomorphism was studied from different 'sides'.

'Left side' of homomorphisms. For any fixed graph G, $\mathrm{HOM}(G, \_)$ is trivially solvable in polynomial time. Several authors independently showed that $\mathrm{HOM}(\mathcal{G}, \_)$ is solvable in polynomial time if all graphs in $\mathcal{G}$ have bounded treewidth [7, 8, 11]. In this case polynomial-time algorithms can be obtained even for counting homomorphisms [8]. Grohe, concluding from the results of Dalmau et al. [7], showed that $\mathrm{HOM}(\mathcal{G}, \_)$ is solvable in polynomial time if and only if the cores of all graphs in $\mathcal{G}$ have bounded treewidth (under some parameterized complexity theoretic assumptions) [11].

'Right side' of homomorphisms. Hell and Nešetřil showed that for any fixed simple graph H, the problem $\mathrm{HOM}(\_, H)$ is solvable in polynomial time if H is bipartite, and NP-complete if H is not bipartite [12]. This resolves the complexity classification of the whole right side of homomorphisms, and provides a P vs. NP dichotomy. Consequently the study of the right side of homomorphisms for undirected graphs almost stopped, and research has been mainly concentrated on finding polynomial-time algorithms for special graph classes from the 'left side'.

On the other hand extensive work has been done recently for graph coloring, a special case of graph homomorphism, resulting in faster and faster exponential-time algorithms. The recent best bounds are an $\mathcal{O}(1.3289^n)$-time algorithm for 3-coloring [4], an $\mathcal{O}(1.7504^n)$-time algorithm for 4-coloring [5], an $\mathcal{O}(2.1020^n)$-time algorithm for 5-coloring [6], and an $\mathcal{O}(2.1809^n)$-time algorithm for 6-coloring [6]. For $k \geq 7$, the k-coloring problem can be solved in time $\mathcal{O}(2.4023^n)$ [5].

Despite considerable progress on exponential-time algorithms for graph coloring problems, not much is known on exponential-time algorithms for the graph homomorphism problem. By the result of Hell and Nešetřil, $\mathrm{HOM}(\_, H)$ is polynomial-time solvable when H is bipartite. Another 'easy' case is when $\chi(H) = \omega(H)$, i.e., the chromatic number of H is equal to its maximum clique size. It is not hard to show that in this case the $\mathrm{HOM}(\_, H)$ problem is equivalent to the k-coloring problem with $k = \chi(H)$ (see Lemma 1). Consequently the $\mathrm{HOM}(\_, H)$ problem is equivalent to the $\chi(H)$-coloring problem for all perfect graphs H.

All this motivates us to study exact algorithms for $\mathrm{HOM}(\_, H)$ for fixed graphs H satisfying $\chi(H) > \omega(H)$. Thus chordless cycles of odd length are the first natural candidates to study exponential-time algorithms for graph homomorphisms. For the cycle C_3 on 3 vertices $\mathrm{HOM}(\_, C_3)$ is equivalent to 3-

coloring, but already for the cycle C_5 on 5 vertices no better deterministic algorithm than the brute-force $\mathcal{O}^*(5^n)$ time algorithm has been known. (Throughout this paper, in addition to the standard big-Oh notation $\mathcal{O}$, we sometimes use a modified big-Oh notation $\mathcal{O}^*$ that suppresses all polynomially bounded factors. For functions f and g we write $f(n) = \mathcal{O}^*(g(n))$ if $f(n) = g(n) \cdot n^{\mathcal{O}(1)}$.)

Our results. In this paper we initiate the study of exponential time complexity of graph homomorphism problems beyond graph coloring. We show that for an input graph G on n vertices and an odd integer $k \geq 5$, HOM($\_, C_k$) is solvable in $\mathcal{O}^*(\min\{\binom{n}{\lfloor n/k \rfloor}, 2^{n/2}\})$ time, where C_k is the cycle on k vertices. In particular, the running time of our algorithm is $\mathcal{O}^*(2^{n/2})$ when $k \in \{5, 7, 9\}$, and $\mathcal{O}(\alpha^n)$ with $\alpha < \sqrt{2}$ for all $k \geq 11$. It is interesting to note that, for $k \geq 13$, our algorithm for homomorphism to C_k is faster than the fastest known 3-coloring algorithm. Hence the natural conjecture that HOM($\_, C_k$) is at least as difficult as 3-coloring for every odd $k \geq 5$ might be mistaken. Our algorithms use 2-SAT expressions to search for suitable extensions of an initial partial homomorphism: a maximal independent set of G to be mapped to a carefully chosen subset of vertices of H. To enumerate all possible preliminary choices we use known algorithms to enumerate all maximal independent sets.

Treewidth and tree decompositions are of great importance in structural graph theory and graph algorithms. Many NP-hard graph problems become polynomial-time or even linear-time solvable when the input is restricted to graphs of bounded treewidth. We refer to [3] for a survey on this parameter. It seems that the treewidth can be a useful tool to design exponential-time algorithms as well. We use dynamic programming techniques similar to bounded treewidth techniques to decide whether an input graph G is homomorphic to a graph H in time $\mathcal{O}^*((t+3)^{|V(G)|})$, where H is a graph of treewidth at most t and a tree decomposition of H of width at most t is supposed to be known in advance.

2 Preliminaries

We consider undirected and simple graphs, where $V(G)$ denotes the set of vertices and $E(G)$ denotes the set of edges of a graph G. We denote the number of vertices $|V(G)|$ of (input) graph G by n, if there is no ambiguity. For a given subset S of $V(G)$, $G[S]$ denotes the subgraph of G induced by S, and $G - S$ denotes the graph $G[V(G) \setminus S]$. A vertex set S of G is an *independent set* if $G[S]$ is a graph with no edges, and S is a *clique* if $G[S]$ is a complete graph. The set of neighbors of a vertex v in G is denoted by $N_G(v)$, and the set of neighbors of a vertex set S is $N_G(S) = \bigcup_{v \in S} N_G(v) \setminus S$. A *connected component* of a graph G is the vertex set of a maximal connected subgraph of G. A subset of vertices $S \subseteq V(G)$ is a *separator* of the graph G if $G - S$ is disconnected.

The complete graph on k vertices is denoted by K_k and the chordless cycle on k vertices is denoted by C_k. A *coloring* of a graph G is a function assigning a color to each vertex of G such that adjacent vertices have different colors. A

k-coloring of a graph uses at most k colors, and the smallest number of colors in a coloring of G is denoted by $\chi(G)$. The maximum size of a clique in a graph G is denoted by $\omega(G)$

Given a mapping $\varphi : V(G) \longrightarrow V(H)$ and a set $U \subseteq V(H)$, we denote by $\varphi^{-1}(U)$ the set of all those vertices of G that are mapped to a vertex of U.

The notion of treewidth was introduced by Robertson and Seymour [17]. A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i : i \in I\},\ T)$, where $\{X_i : i \in I\}$ is a collection of subsets of G (these subsets are called *bags*) and $T = (I, F)$ is a tree such that the following three conditions are satisfied:

1. $\bigcup_{i \in I} X_i = V(G)$.
2. For all $\{v, w\} \in E(G)$, there is an $i \in I$ such that $v, w \in X_i$.
3. For all $i, j, k \in I$, if j is on a path from i to k in T then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition $(\{X_i : i \in I\},\ T)$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* of a graph G, denoted by $\mathbf{tw}(G)$, is the minimum width over all its tree decompositions. A tree decomposition of G of width $\mathbf{tw}(G)$ is called an *optimal* tree decomposition of G.

For the sake of completeness, we conclude this section with a proof of the following lemma which was mentioned in the introductory section.

Lemma 1. *Let H be a graph such that $\chi(H) = \omega(H)$. Then for any graph G the following are equivalent*

- *There is a homomorphism from G to H;*
- *G is colorable in $\chi(H)$ colors.*

Proof. Let $k = \chi(H)$. If G is homomorphic to H, then G is trivially k-colorable since H is k-colorable. If G is k-colorable, then G is homomorphic to K_k. Since H contains a clique of size k, we have that the vertices of G can be mapped to the vertices of this clique, and the result follows. □

3 Homomorphisms to odd cycles

Recall that $\mathrm{HOM}(\_, C_k)$ is solvable in polynomial time if k is even, and NP-complete if k is odd. We study the case when $k \geq 5$ is an odd integer. Throughout the remainder of this section we assume the input graph G to be non bipartite, since every bipartite graph is homomorphic to K_2, and thus also homomorphic to C_k for all $k \geq 3$.

For a given graph G and a vertex subset $S \subseteq V(G)$, we define the levels of breadth first search starting at S as follows:

- $L_0(S) = S$;
- $L_i(S) = N_G(L_{i-1}(S)) \setminus \bigcup_{j<i} L_j(S)$, for $i > 0$.

The following technical lemma will be used in the proof of the main results of this section.

Lemma 2. *Let $k \geq 3$ be an odd integer. A non bipartite graph $G = (V, E)$ is homomorphic to C_k if and only if there is a set $S \subseteq V$ such that*

- $|S| \leq |V(G)|/k$,
- *the levels $L_0(S), L_1(S), L_2(S), \ldots, L_{\lfloor \frac{k}{2} \rfloor - 1}(S)$ are independent sets in G,*
- *the graph $G - S$ is bipartite, and*
- *there is a coloring of the vertices of $G - S$ in Red and Blue such that every pair of adjacent vertices in $\bigcup_{i=1}^{\lfloor \frac{k}{2} \rfloor} L_i(S)$ from different levels have the same color, and every pair of adjacent vertices in $\bigcup_{i \geq \lfloor \frac{k}{2} \rfloor} L_i(S)$ have different colors.*

Proof. Let us choose a vertex $v \in V(C_k)$ and let $R = (v, r_1, r_2, \ldots, r_{\lfloor k/2 \rfloor})$ and $B = (v, b_1, b_2, \ldots, b_{\lfloor k/2 \rfloor})$ be the two edge disjoint paths in C_k of length $\lfloor k/2 \rfloor$ starting at v.

Let G be homomorphic to C_k. Observe that every proper subgraph of C_k is bipartite, so there is a homomorphism from G to a proper subgraph of C_k only if G is bipartite. Thus, since G is not bipartite, every homomorphism from G to C_k is surjective. Consequently, there is a homomorphism τ from G to C_k such that $|\tau^{-1}(v)| \leq |V(G)|/k$. We define $S = \tau^{-1}(v)$. We then choose a homomorphism $\varphi \colon G \longrightarrow C_k$ that minimizes

$$\sum_{1 \leq i \leq \lfloor k/2 \rfloor} \frac{|\varphi^{-1}(r_i)| + |\varphi^{-1}(b_i)|}{i} \tag{1}$$

subject to $\varphi^{-1}(v) = S$.

By (1), every vertex of $\varphi^{-1}(r_i)$, $i \in \{1, 2, \ldots, \lfloor k/2 \rfloor - 1\}$, is adjacent in G to a vertex of $\varphi^{-1}(r_{i-1})$. In fact, suppose on the contrary that there is a vertex $x \in \varphi^{-1}(r_i)$ that is not adjacent in G to any vertex of $\varphi^{-1}(r_{i-1})$. Then there is a homomorphism ϕ from G to C_k such that $\phi(y) = \varphi(y)$ for all $y \neq x$, and $\phi(x) = r_{i+2}$ if $i \leq \lfloor k/2 \rfloor - 2$ and $\phi(x) = b_{\lfloor k/2 \rfloor}$ if $i = \lfloor k/2 \rfloor - 1$. But the existence of such a homomorphism contradicts (1). By similar arguments, every vertex of $\varphi^{-1}(b_i)$ $i \in \{1, 2, \ldots, \lfloor k/2 \rfloor - 1\}$ is adjacent to a vertex of $\varphi^{-1}(b_{i-1})$.

Thus for every $i \in \{1, 2, \ldots, \lfloor k/2 \rfloor - 1\}$, the vertices of $\varphi^{-1}(r_i) \cup \varphi^{-1}(b_i)$ form the level $L_i(S)$ of breadth first search starting at S in G. Furthermore, each of these sets is an independent set. The graph $G - S$ is bipartite because it is homomorphic to a path. For $i \in \{1, 2, \ldots, \lfloor k/2 \rfloor\}$, we color the vertices of $\varphi^{-1}(r_i)$ in Red and the vertices of $\varphi^{-1}(b_i)$ in Blue. Such a coloring satisfies the conditions of the lemma.

Now suppose that there is a vertex set $S \subseteq V(G)$ and a breadth first search starting at S satisfying the conditions of the lemma. We construct a homomorphism from G to C_k by mapping S to v. For $i \in \{1, 2, \ldots, \lfloor k/2 \rfloor - 1\}$, all Red vertices from level $L_i(S)$ are mapped to r_i and all Blue vertices from level $L_i(S)$ are mapped to b_i. For $i \geq \lfloor k/2 \rfloor$, Red vertices from level $L_i(S)$ are mapped to $r_{\lfloor k/2 \rfloor}$ and Blue vertices from level $L_i(S)$ are mapped to $b_{\lfloor k/2 \rfloor}$. □

Lemma 3. *For any odd integer $k \geq 5$, $\mathrm{HOM}(\_, C_k)$ can be solved in time $\mathcal{O}^*(\binom{n}{\lfloor n/k \rfloor})$.*

Proof. By Lemma 2, a non bipartite graph G is homomorphic to C_k if and only if there is a set $S \subseteq V(G)$ satisfying the conditions of the lemma.

For a given (independent) set S, one can decide whether S satisfies the conditions of Lemma 2 as follows.

1. Find the levels $L_0(S), L_1(S), L_2(S), \ldots, L_m(S)$ of breadth first search at S. If all sets $L_0(S), L_1(S), L_2(S), \ldots, L_{\lfloor \frac{k}{2} \rfloor - 1}(S)$ are independent sets in G proceed to step 2.
2. Check if $G - S$ is bipartite. If it is bipartite proceed to step 3.
3. To decide whether there is a coloring of the vertices of $G - S$ which meets the condition of Lemma 2, we reduce the problem to 2-SAT as follows. We encode every vertex x of $G - S$ by a boolean variable x such that $x = \text{TRUE}$ means that vertex x is colored Red, and variable $x = \text{FALSE}$ means that vertex x is colored Blue. Every edge $\{x, y\}$ between $L_i(S)$ and $L_{i+1}(S)$, for each $1 \leq i \leq \lfloor \frac{k}{2} \rfloor - 1$, is encoded by two clauses $(\bar{x} \vee y)$ and $(x \vee \bar{y})$. This forces vertex x and vertex y to receive the same color. Every edge $\{u, v\}$ with both endpoints in $\bigcup_{i \geq \lfloor \frac{k}{2} \rfloor} L_i(S)$ is encoded by two clauses $(u \vee v)$ and $(\bar{u} \vee \bar{v})$. This forces vertex u and vertex v to receive opposite colors. The corresponding 2-SAT formula is satisfiable if and only if S satisfies the conditions of Lemma 2 and there is a homomorphism from G to C_k that can be derived from S.

Consequently, for each given set S, constructing a homomorphism from G to C_k using S or concluding that S cannot be used can be done by solving the corresponding 2-SAT formula, and thus requires linear time (see [1]). There are less than $(n/k)\binom{n}{\lfloor n/k \rfloor}$ different subsets S of size at most n/k. Hence the total running time is $\mathcal{O}^*(\binom{n}{\lfloor n/k \rfloor})$. □

Our next algorithm improves upon the running time of the previous one for $k \in \{5, 7, 9\}$. First, we need the following algorithmic version of the result from [14] which is due to Byskov [5].

Proposition 4 ([5]). *All maximal independent sets in a triangle-free graph on n vertices can be listed in time $\mathcal{O}^*(2^{n/2})$.*

Lemma 5. *For any odd integer $k \geq 5$, $\mathrm{HOM}(\_, C_k)$ can be solved in time $\mathcal{O}^*(2^{n/2}) = \mathcal{O}(1.41422^n)$.*

Proof. We may assume that $G = (V, E)$ is not bipartite. Furthermore C_k is 3-colorable and triangle-free for every odd integer $k \geq 5$. Thus G is homomorphic to C_k implies that G is 3-colorable and triangle-free.

Let $v_1, v_2, v_3, \ldots, v_{k-1}, v_k$ be the vertices of C_k, where v_i is adjacent to v_{i+1} for $1 \leq i \leq k-1$, and v_k is adjacent to v_1. We choose the following maximal independent set of C_k: $U = \{v_2, v_4, \ldots, v_{k-3}, v_{k-1}\}$. Suppose there is a homomorphism $\varphi : G \longrightarrow C_k$. Then $\varphi^{-1}(U)$ is an independent set of G. We claim that in this case there is even a homomorphism $\psi : G \longrightarrow C_k$ such that $\psi^{-1}(U)$ is a maximal independent set of G. Let $x \in V(G) \setminus \varphi^{-1}(U)$ such that $\{x\} \cup \varphi^{-1}(U)$ is an independent set of G, and let y be a neighbor of x in G. Then $\{x, y\} \in E(G)$ implies $\{\varphi(x), \varphi(y)\} = \{v_1, v_k\}$. Thus the following modification of φ is a homomorphism from G to C_k. Let $I' \subseteq V(G) \setminus \varphi^{-1}(U)$ such that $I = I' \cup \varphi^{-1}(U)$ is a maximal independent set of G. We define a homomorphism $\psi : G \longrightarrow C_k$ such that $\psi^{-1}(U) = I'$. For every vertex $v \in V(G) \setminus I'$, we let $\psi(v) = \varphi(v)$. For every vertex $v \in I'$, we let $\psi(v) = v_2$ if $\varphi(v) = v_k$, and we let $\psi(v) = v_{k-1}$ if $\varphi(v) = v_1$.

The goal of our algorithm is to test, for every maximal independent set I of G, whether there is a homomorphism $\psi : G \longrightarrow C_k$ such that $\psi^{-1}(U) = I$. By the above claim, ψ must exist if G is homomorphic to C_k. For every maximal independent set I in G the test is done as follows: First, if $G - I$ is not bipartite, then reject I since a non bipartite graph cannot be homomorphic to $C_k - U$ which consists of a K_2 and $(k-1)/2$ isolated vertices. If $G - I$ is bipartite, let A be the set of isolated vertices of $G - I$, and let J be the set of vertices in connected components of $G - I$ that have at least two vertices. Clearly $V(G) = I \cup A \cup J$. Furthermore, since G is not bipartite, $J \neq \emptyset$.

Every vertex of J must be mapped to v_1 or v_k since each connected component of $G[J]$ has at least two vertices. Then every vertex of $N(J)$ must be mapped to v_2 or v_{k-1}. Clearly $N(J) \subseteq I$. Following Lemma 2, we map the vertices of G in a breadth first search manner starting from J, with levels $L_0(J)$=J, $L_1(J)$=$N(J)$, $L_2(J)$, ..., $L_{(k-1)/2}(J)$. At any stage we consider only the vertices that have to be mapped due to adjacencies in G to already mapped vertices. Therefore the vertices of $L_2(J)$ must be mapped to v_3 or v_{k-2}. Clearly $L_2(J) \subseteq A$. The vertices of $L_3(J)$ must be mapped to v_4 or v_{k-3}, $\ldots$, the vertices of $L_{(k-3)/2}(J)$ must be mapped to $v_{(k-1)/2}$ or $v_{(k+3)/2}$, and finally the vertices of $L_{(k-1)/2}(J)$ must be mapped to $v_{(k+1)/2}$. Now, there may be some remaining vertices of G that are not assigned to any vertex of C_k by the above procedure. If $(k+1)/2$ is even, then all remaining vertices should be mapped to $v_{(k-1)/2}$ or $v_{(k+3)/2}$ if they belong to A, and to $v_{(k+1)/2}$ if they belong to I. If $(k+1)/2$ is odd, then we should do the reverse: the remaining vertices should be mapped to $v_{(k+1)/2}$ if they belong to A and to $v_{(k-1)/2}$ or $v_{(k+3)/2}$ if they belong to I. Consequently, in the end, vertices of $A \cup J$ are mapped to $V(C_k) \setminus U$, and vertices of I are mapped to U.

To check whether our partial mapping can be transformed into a homomorphism we shall use a 2-SAT formula. For all vertices of G except those mapped to $v_{(k+1)/2}$ there is a choice between two vertices of the host graph C_k. Furthermore adjacent vertices of G must be mapped to adjacent vertices of C_k.

For every vertex x of G with $\varphi(x) \in \{v_i, v_{k-i+1}\}$ we define a boolean variable x such that variable $x =$ TRUE means that vertex x is mapped to v_i with $i = 1, 2, \ldots, (k-1)/2$, and variable $x =$ FALSE means that vertex x is mapped to v_i with $i = (k+3)/2, (k+5)/2, \ldots, k$. For each edge $\{x, y\} \in E(G[J])$, either $\varphi(x) = v_1$ and $\varphi(y) = v_k$, or vice versa. Otherwise, for each edge $\{x, y\} \in E(G)$ with $\{x, y\} \not\subseteq J$, either $\varphi(x) = v_i$ and $\varphi(y) = v_j$ with $i, j \in \{1, 2, \ldots, (k-1)/2\}$, or $\varphi(x) = v_i$ and $\varphi(y) = v_j$ with $i, j \in \{(k+3)/2, \ldots, k\}$. Therefore, for each edge $\{x, y\} \in E(G[J])$, we insert the following two clauses in our 2-SAT formula: $(\bar{x} \vee y)$ and $(x \vee \bar{y})$. For all other edges $\{x, y\} \in E(G)$, i.e., at least one of x and y does not belong to J, we insert the following two clauses in our 2-SAT formula: $(\bar{x} \vee \bar{y})$ and $(x \vee y)$.

The corresponding 2-SAT formula is satisfiable if and only if there is a homomorphism φ from G to C_k such that $\varphi^{-1}(U) = I$. Consequently, for each maximal independent set I of G, constructing a homomorphism from G to C_k using I or concluding that I cannot be used can be done by solving the corresponding 2-SAT formula, and thus requires linear time (see [1]). By Proposition 4, the number of maximal independent sets in a triangle free graph is at most $2^{n/2}$ and all maximal independent sets of a triangle free graph can be enumerated in time $\mathcal{O}^*(2^{n/2})$. Thus the overall running time of our algorithm is $\mathcal{O}^*(2^{n/2})$. □

The algorithm of Lemma 3 has running time $\mathcal{O}(1.64939^n)$ when $k = 5$, $\mathcal{O}(1.50700^n)$ when $k = 7$, and $\mathcal{O}(1.41742^n)$ when $k = 9$, and its running time is $\mathcal{O}(\alpha^n)$ with $\alpha < \sqrt{2}$ for all $k \geq 11$. Hence the algorithm of Lemma 3 is faster for all $k \geq 11$, and the algorithm of Lemma 5 is faster for $k \in \{5, 7, 9\}$. Combining Lemmata 3 and 5 we obtain the following theorem.

Theorem 6. *For any odd integer $k \geq 5$, $\mathrm{HOM}(\_, C_k)$ can be solved in time $\mathcal{O}^*(\min\{\binom{n}{\lfloor n/k \rfloor}, 2^{n/2}\})$.*

To conclude this section, let us consider a natural extensions of cycles. Let W_k be a wheel obtained from C_k by adding a vertex u adjacent to all vertices $v_1, v_2, \ldots, v_k$ of C_k. If k is even then $\chi(W_k) = \omega(W_k) = 3$, and thus $\mathrm{HOM}(\_, W_k)$ is equivalent to the well-studied 3-coloring problem by Lemma 1.

Hence we concentrate on $\mathrm{HOM}(\_, W_k)$ for odd wheels, and we assume that $k \geq 5$ be an odd integer.

There is an easy way to use the algorithms of Lemmata 3 and 5 as building blocks to obtain exponential time algorithms for the $\mathrm{HOM}(\_, W_k)$ problem, where $k \geq 5$ is an odd integer.

Corollary 7. *For any odd integer $k \geq 5$, $\mathrm{HOM}(\_, W_k)$ can be solved in time $\mathcal{O}^*(3^{n/3} \cdot \min\{2^{n/2}, \binom{n}{\lfloor n/k \rfloor}\})$.*

Proof. First we show that if the input graph G is homomorphic to W_k, then there is a homomorphism ψ such that $\psi^{-1}(u)$ is a maximal independent set of G, and then we show how to construct an algorithm using this property. To establish the property, note that $\varphi^{-1}(u)$ is an independent set of G for any homomorphism $\varphi : G \to W_k$. If $\varphi^{-1}(u)$ is not a maximal independent set of G

then take any maximal independent set I of G such that $\varphi^{-1}(u) \subseteq I$. Now we construct a new mapping ψ as follows: $\psi(v) = u$ for all $v \in I$ and $\psi(v) = \varphi(v)$ for all $v \in V \setminus I$, which is a homomorphism since u is adjacent to all other vertices in W_k.

Thus our algorithm simply checks whether there is a maximal independent set I of G such that $G - I$ is homomorphic to C_k and uses an algorithm of Lemmata 3 and 5 to verify whether $G - I$ is homomorphic to C_k. Finally taking into account that a graph on n vertices has at most $3^{n/3}$ maximal independent sets and that these sets can be enumerated in time $O^*(3^{n/3})$ [16], we obtain the claimed upper bound for the running time. □

It is an interesting question whether faster algorithms solving HOM($\_, W_k$) can be established, and whether such algorithms must use an algorithm solving HOM($\_, C_k$) as subroutine.

4 Homomorphisms to graphs of bounded treewidth

A tree decomposition $(\{X_i : i \in I\},\ T)$ of a graph G is said to be *nice* if a root of T can be chosen such that every node $i \in I$ of T has at most two children in the rooted tree T, and

1. if a node $i \in I$ has two children j_1 and j_2 then $X_i = X_{j_1} = X_{j_2}$ (i is called a **join** node),
2. if a node $i \in I$ has one child j, then either $X_i \subset X_j$ and $|X_i| = |X_j| - 1$ (i is called a **forget** node), or $X_j \subset X_i$ and $|X_j| = |X_i| - 1$ (i is called an **introduce** node),
3. if a node $i \in I$ is a leaf of T, then $|X_i| = 1$ (i is called a **leaf** node).

Given a nice tree decomposition $(\{X_i : i \in I\},\ T)$, we denote by T_i the subtree of T rooted at node i, for each $i \in I$. The parent of node i is denoted by $p(i)$.

It is known that every graph G of treewidth at most t has a nice tree decomposition $(\{X_i : i \in I\},\ T)$ of width t such that $|I| = \mathcal{O}(t \cdot n)$. Furthermore, given a tree decomposition of G of width t, a nice tree decomposition of G of width t can be computed in time $\mathcal{O}(n)$ [15].

There is an $\mathcal{O}(1.8899^n)$ time algorithm to compute the treewidth and an optimal tree decomposition for any given graph [10, 18]. There is also a well-known linear-time algorithm to compute the treewidth and an optimal tree decomposition for graphs of bounded treewidth [2].

We now present an algorithm to decide whether for given graphs G and H there is an homomorphism from G to H. The algorithm is based on dynamic programming on a nice optimal tree decomposition of H.

Theorem 8. *There is an $\mathcal{O}^*((\mathbf{tw}(H)+3)^{|V(G)|})$ time algorithm taking as input a graph G, a graph H, and an optimal tree decomposition of H, that decides whether G is homomorphic to H and produces a homomorphism $\varphi : G \longrightarrow H$ if there is one.*

Proof. Let $n = |V(G)|$ and $t = \mathbf{tw}(H)$. First our algorithm transforms the given optimal tree decomposition of H into a nice tree decomposition $(\{Y_i : i \in J\},\ U)$ of width t. Then we modify this nice tree decomposition as follows. For every non root node $i \in J$ of tree U we add a new **nochange** node i' as the parent of i, and we let the old parent of i in tree U become the parent of i' in the new tree. We let $X_{i'} = X_i = Y_i$. In this way we obtain a new tree decomposition $(\{X_i : i \in I\},\ T)$ of H of width t. In the new tree T, the parent of every node of U is a **nochange** node, which is more convenient for our following argumentation. Furthermore, the bags of the parent and of a child of node i in T differ by at most one vertex.

We define two auxiliary subsets of vertices of H for each node $i \in I$ of T: $V_i = \bigcup_{j \in V(T_i)} X_j$, and $\tilde{X}_i = X_i \cap X_{p(i)}$. Notice that $\tilde{X}_i = X_i$ if $p(i)$ is an introduce, join, or nochange node, and that $\tilde{X}_i = X_i \setminus \{u\}$ if $p(i)$ is a forget node with $X_{p(i)} = X_i \setminus \{u\}$. For r, the root of T, we define $\tilde{X}_r = X_r$.

The following simple observation is essential for our algorithm.

Observation. *Let $\varphi : G \longrightarrow H$ be a homomorphism, and let $X \subseteq V(H)$ be a separator of H such that for each connected component D of $H - X$, $V(G) \neq \varphi^{-1}(D \cup X)$. Then $\varphi^{-1}(X)$ is a separator of G. Furthermore, for each connected component D of $H - X$, the set $\varphi^{-1}(D)$ is a connected component of $G - \varphi^{-1}(X)$.*

Proof. Vertices mapped to different components D and D' of $H - X$ cannot be adjacent, and thus there are no edges of G with one extremity in $\varphi^{-1}(D)$ and the other in $\varphi^{-1}(D')$. Consequently, $\varphi^{-1}(D) \neq \emptyset$ and $\varphi^{-1}(D') \neq \emptyset$ implies that $G - \varphi^{-1}(X)$ is disconnected, and its components are all of the form $\varphi^{-1}(D)$, where D is a component of $H - X$. $\square$

Our algorithm computes for each node $i \in I$ of T in a bottom-up fashion all characteristics of i, defined as follows.

Definition. *A tuple $(S; (v_1, S_1), (v_2, S_2), \ldots, (v_{l_i}, S_{l_i}); Q; i)$ is a characteristic of node $i \in I$ of T if the following conditions are satisfied.*

1. *$S \subseteq V(G)$, the sets $S_1, S_2, \ldots, S_{l_i}$ (some of which might be empty) are disjoint subsets of S and $\bigcup_{j=1}^{l_i} S_j = S$.*
2. *$\tilde{X}_i = \{v_1, v_2, \ldots, v_{l_i}\}$*
3. *If $\tilde{X}_i$ is a separator of H then $Q \subseteq V(G) \setminus S$ is a (possibly empty) union of connected components of $G - S$. Otherwise either $Q = \emptyset$, or $Q = V(G) \setminus S$.*
4. *There is a homomorphism φ from $G[S \cup Q]$ to $H[V_i]$ such that $\varphi^{-1}(\tilde{X}_i) = S$ and for every $j \in \{1, 2, \ldots, l_i\}$, $\varphi^{-1}(v_j) = S_j$.*

Notice that characteristics are defined in such a way that G is homomorphic to H if and only if there is at least one characteristic for the root r of T satisfying $S \cup Q = V(G)$. In fact, any characteristic of a node i satisfying $S \cup Q = V(G)$ corresponds to a homomorphism $\varphi : G \longrightarrow H$, and thus the following algorithm may terminate after discovering such a characteristic.

For each forget, introduce, nochange, and join node $i \in I$ of T, our algorithm computes by dynamic programming all characteristics $(S; (v_1, S_1), \ldots, (v_{l_i}, S_{l_i}); Q; i)$ of node i using for each of i's children the set of all its characteristics. Included is a cleaning step to be carried out after all characteristics of a node i have been computed, based on the following observation. If $(S; (v_1, S_1), \ldots, (v_{l_i}, S_{l_i}); Q'; i)$ and $(S; (v_1, S_1), \ldots, (v_{l_i}, S_{l_i}); Q''; i)$ are characteristics of i then $(S; (v_1, S_1), \ldots, (v_{l_i}, S_{l_i}); Q' \cup Q''; i)$ is also a characteristic of i, and we may discard the first two characteristics. Thus in the cleaning step for all tuples $(S; (v_1, S_1), \ldots, (v_{l_i}, S_{l_i}))$ the largest set Q for which $(S; (v_1, S_1), \ldots, (v_{l_i}, S_{l_i}); Q; i)$ is a characteristic of i is stored and all other such tuples are deleted. Consequently, after the cleaning step the set of all characteristics of node i contains for each $(S; (v_1, S_1), \ldots, (v_{l_i}, S_{l_i}))$ at most one characteristic.

Consequently the number of characteristics of a node i of T (after cleaning) is at most $\sum_{\ell=0}^{n} \binom{n}{\ell} \cdot (t+1)^{\ell} = (t+2)^n$ since $|\tilde{X}_\ell| \leq t+1$.

Now we describe how the set of all characteristics can be computed from the characteristics of the children for the different types of nodes in T.

Leaf node:
Let i be a leaf node, thus $X_i = \{u\}$ for some vertex u of H. For a subset S of $V(G)$, there is a homomorphism φ from $G[S]$ to $H[V_i]$ with $V_i = \{u\}$ if and only if $\varphi^{-1}(\{u\}) = S$, and hence S is an independent set. Furthermore $(S; (u, S); \emptyset; i)$ is a characteristic of the leaf node i if and only if S is an independent set of G.

Introduce node:
Let i be an introduce node with child j. Thus $X_i = X_j \cup \{u\}$ for some vertex $u \in V(H) \setminus V_j$, and consequently $\tilde{X}_j = X_j$. Notice that the parent of i is a nochange node, and thus $X_i = X_{p(i)}$ and $\tilde{X}_i = \tilde{X}_j \cup \{u\}$.

All characteristics of node i can be obtained by extending characteristics $(S; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}); Q; j)$ of node j. Since $\tilde{X}_i = \tilde{X}_j \cup \{u\}$, each characteristic of i obtained from $(S; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}); Q; j)$ is of the form $(S \cup S'; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}), (u, S'); Q; i)$ where $S' \subseteq V(G) \setminus (S \cup Q)$ is an independent set in G, and for all $x \in N_{G[S]}(S')$, $\varphi(x) \in N_H(u)$. These conditions can be checked in polynomial time. Finally one characteristic of j extends to at most $2^{n-|S|}$ characteristics of i, since S' must be an independent set of $G - S$. Therefore we compute at most $\sum_{\ell=0}^{n} \binom{n}{\ell} \cdot (t+1)^{\ell} \cdot 2^{n-\ell} = (t+3)^n$ characteristics to obtain the set of all characteristics of an introduce node.

Forget node:
Let i be a forget node with child j. Thus $X_i = X_j \setminus \{u\}$ for some vertex $u \in X_j$, and consequently $\tilde{X}_j = X_i$. The parent of i is a nochange node, and thus $X_i = X_{p(i)}$ and $\tilde{X}_i = \tilde{X}_j$.

$\tilde{X}_i = \tilde{X}_j$ implies that each characteristic of i can be obtained directly from a characteristic of j by simply replacing j with i. Hence $(S; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}); Q; j)$ is a characteristic of j if and only if $(S; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}); Q; i)$ is a characteristic of i.

Nochange node:
Let i be nochange node with child j. Thus $X_i = X_j$. If the parent of i is an introduce or join node, then $\tilde{X}_i = \tilde{X}_j$. In this case $(S; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}); Q; j)$ is a characteristic of j if and only if $(S; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}); Q; i)$ is a characteristic of i. Thus each characteristic of i can be obtained by one characteristic of j.

If the parent of i is a forget node then $X_{p(i)} = X_i \setminus \{u\}$ for some vertex $u \in X_i$, and thus $\tilde{X}_i = \tilde{X}_j \setminus \{u\}$. Now $\tilde{X}_i = \tilde{X}_j \setminus \{u\}$ implies that each characteristic of i can be obtained from a characteristic of j, say $(S; (v_1, S_1), \ldots, (v_{l_j}, S_{l_j}); Q; j)$, by removing the pair (v_q, S_q) where $u = v_q$ and replacing Q by $Q' = Q \cup S_q$, obtaining $(S; (v_1, S_1), \ldots, (v_{q-1}, S_{q-1}), (v_{q+1}, S_{q+1}), \ldots, (v_{l_i}, S_{l_i}); Q'; i)$.

Thus again each characteristic of j extends into one characteristic of i.

Join node:
Let i be a join node with children j_1 and j_2; thus $X_i = X_{j_1} = X_{j_2}$. The parent of i is a nochange node, thus $\tilde{X}_i = \tilde{X}_{j_1} = \tilde{X}_{j_2} = X_i$.

Let $(S'; (v_1, S'_1), \ldots, (v_{l_{j_1}}, S'_{l_{j_1}}); Q'; j_1)$ be a characteristic of j_1, and let $(S''; (v_1, S''_1), \ldots, (v_{l_{j_2}}, S''_{l_{j_2}}); Q''; j_2)$ be a characteristic of j_2. Both extend into a characteristic of node i if $l_i = l_{j_1} = l_{j_2}$ and $S'_k = S''_k$ for all $k = 1, 2, \ldots, j_1$, and thus $S' = S''$. In this case $(S'; (v_1, S'_1), \ldots, (v_{l_i}, S'_{l_i}); Q; i)$ is a characteristic of i if $Q = Q' \cup Q''$ and $Q' \cap Q'' = \emptyset$, and there is no edge between a vertex of Q' and a vertex of Q'' in G.

Note that by the cleaning step for each tuple $(S'; (v_1, S'_1), \ldots, (v_{l_i}, S'_{l_i}))$ there is at most one such characteristic for j_1, and thus Q' is unique, and there is at most one such characteristic for j_2, and thus Q'' is unique. Hence at most one characteristic of i will be obtained for each choice of the set $S \subseteq V(G)$ and each partition of S into at most $t+1$ subsets. Therefore we compute at most $\sum_{\ell=0}^{n} \binom{n}{\ell} \cdot (t+1)^\ell = (t+2)^n$ characteristics to obtain the set of all characteristics of a join node.

Finally, notice that the number of nodes in the decomposition is a polynomial in $|V(H)|$, and that suitable standard data structures, like binary search trees, guarantee that the characteristics of a node can be stored such that find and insert operations can be done in polynomial (logarithmic in the size of the stored data) time. Thus the overall running time of our algorithm is $\mathcal{O}^*(\mathbf{tw}(H) + 3)^{|V(G)|})$. □

5 Concluding remarks and open questions

Consider the problem HOM$(_,_)$. For input graphs graphs G and H, within which time bound can be decided whether G is homomorphic to H? The trivial solution brings us a $\mathcal{O}^*(|V(H)|^{|V(G)|})$ time algorithm. There is a faster algorithm through recent results on the constraint satisfaction problem with at most 2 variables per constraint (2-CSP). Williams [19] obtained an $\mathcal{O}(d^{n\cdot\omega/3})$-time algorithm for 2-CSP, where $\omega < 2.376$ is the matrix product exponent over a ring, d is the domain size, and n is the number of variables. As a consequence of Williams' result, we have that HOM($\_,\_$) can be solved in time $\mathcal{O}(|V(H)|^{0.793|V(G)|})$.

In this paper we observed that if the right side graph H is of bounded treewidth, then HOM($\_, H$) can be solved in time $c^{|V(G)|} \cdot |V(H)|^{\mathcal{O}(1)}$ for some constant c (depending on $\mathbf{tw}(H)$). Can it be that the problem HOM($\_,\_$) is solvable within running times

1. $f(|V(H)|) \cdot |V(G)|^{\mathcal{O}(1)}$, or
2. $f(|V(G)|) \cdot |V(H)|^{\mathcal{O}(1)}$

for some computable function $f\colon N \to N$? (Unfortunately) the answer to both questions is negative up to some widely believed assumptions in complexity theory.

In fact, for question 1, an $f(|V(H)|) \cdot |V(G)|^{\mathcal{O}(1)}$ time algorithm is also a polynomial-time algorithm for the NP-complete 3-coloring problem implying that P = NP. To answer question 2, we use the widely believed assumption from parameterized complexity [9] that the p-clique problem is not fixed parameter tractable, or in other words, that there is no algorithm for finding a clique of size p in a graph on n vertices in time $f(p) \cdot n^{\mathcal{O}(1)}$ unless FPT = W[1], a collapse of a parameterized hierarchy which is considered to be very unlikely. Since K_p is homomorphic to H if only if H has a clique of size at least p, the problem HOM($K_p,\_$) problem is equivalent to finding a p-clique in H. Therefore, the existence of an $f(|V(G)|)\cdot|V(H)|^{\mathcal{O}(1)}$ time algorithm for HOM($\_,\_$) would imply that the p-clique problem is fixed parameter tractable, thus FPT = W[1].

Now our question is whether a running time of $\mathcal{O}(|V(H)|^{\varepsilon|V(G)|})$ for some constant $\varepsilon > 0$ is the best that we can hope for? Can the problem HOM($\_,\_$) be solved, say by a $c^{|V(G)|+|V(H)|}$-time algorithm for some constant c?

References

[1] B. Aspvall, M. Plass, and R.E. Tarjan, A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters* **8** (1979), 121–123.

[2] H.L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* **25** (1996), 1305–1317.

[3] H. L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* **209** (1998), 1–45.

[4] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. *Journal of Algorithms* **54** (2005), 444–453.

[5] J. M. Byskov, Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters* **32** (2004), 547–556.

[6] J. M. Byskov and D. Eppstein, An algorithm for enumerating maximal bipartite subgraphs. Unpublished.

[7] V. Dalmau, P. G. Kolaitis, and M. Y. Vardi, Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Principles and Practice of Constraint Programming (CP 2002)*, Springer-Verlag, LNCS 2470, 310–326, 2002.

[8] J. Diaz, M. Serna, and D.M. Thilikos, Counting H-colorings of partial k-trees. *Theoretical Computer Science* **281** (2002), 291–309.

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

[10] F. Fomin, D. Kratsch, and I. Todinca, Exact (exponential) algorithms for treewidth and min fill-in. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, Springer-Verlag, LNCS 3124, 568–580, 2004.

[11] M. Grohe, The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, 552–561, 2003.

[12] P. Hell and J. Nešetřil, On the complexity of H-coloring. *Journal of Combinatorial Theory Series B* **48** (1990), 92–110.

[13] P. Hell and J. Nešetřil, Graphs and Homomorphisms. Oxford University Press, 2004.

[14] M. Hujter and Z. Tuza, The number of maximal independent sets in triangle-free graphs. *SIAM Journal on Discrete Mathematics* **6** (1993), 284–288.

[15] T. Kloks, *Treewidth. Computations and approximation.* Springer-Verlag, LNCS 842, Berlin, 1994.

[16] J.W. Moon and L. Moser, On cliques in graphs. *Israel J. Math.* **3** (1965), 23–28.

[17] N. Robertson and P.D. Seymour, Graph Minors. II. Algorithmic Aspects of Tree-Width. *Journal of Algorithms* **7** (1986), 309–322.

[18] Y. Villanger, Improved exponential-time algorithms for treewidth and minimum fill-in. In *Proceedings of the 7th Latin American Symposium (LATIN 2006)*, Springer-Verlag, LNCS 3887, 800–811, 2006.

[19] R. Williams, A new algorithm for optimal constraint satisfaction and its implications. *Theoretical Computer Science* **348** (2005), 357–365.