THEORY OF COMPUTING, Volume 6 (2010), pp. 85–112
www.theoryofcomputing.org

Can You Beat Treewidth?*

Dániel Marx†

Received: September 3, 2008; published: August 27, 2010.

Abstract: It is well-known that constraint satisfaction problems (CSP) over an unbounded domain can be solved in time $n^{O(k)}$ if the treewidth of the primal graph of the instance is at most k and n is the size of the input. We show that no algorithm can do significantly better than this treewidth-based algorithm, even if we restrict the problem to some special class of primal graphs. Formally, let $\mathbb{A}$ be an algorithm solving binary CSP (i. e., CSP where every constraint involves two variables). We prove that if there is a class $\mathcal{G}$ of graphs with unbounded treewidth such that the running time of algorithm $\mathbb{A}$ is $f(G)n^{o(k/\log k)}$ on instances whose primal graph G is in $\mathcal{G}$, where k is the treewidth of the primal graph G and f is an arbitrary function, then the Exponential Time Hypothesis (ETH) fails. We prove the result also in the more general framework of the homomorphism problem for bounded-arity relational structures. For this problem, the treewidth of the core of the left-hand side structure plays the same role as the treewidth of the primal graph above. Finally, we use the results to obtain corollaries on the complexity of (Colored/Partitioned) Subgraph Isomorphism.

ACM Classification: F.2.2, G.2.2

AMS Classification: 68Q17, 68R10

Key words and phrases: constraint satisfaction, treewidth, homomorphism

1 Introduction

Constraint Satisfaction Problems. Constraint satisfaction is a general framework that includes many standard algorithmic problems such as satisfiability, graph coloring, database queries, etc. A constraint

*A preliminary version of this paper appeared in the Proc. 48th Ann. IEEE Symp. on Foundations of Computer Science (FOCS 2007), pages 169–179.

†Research partially supported by the Magyary Zoltán Felsőoktatási Közalapítvány, Hungarian National Research Fund (OTKA 67651), and ERC Advanced Grant DMMCA.

DOI: 10.4086/toc.2010.v006a005

satisfaction problem (CSP) consists of a set V of variables, a domain D, and a set C of constraints, where each constraint is a relation on a subset of the variables. The task is to assign a value from D to each variable in such a way that every constraint is satisfied (see Definition 2.1 for the formal definition). For example, 3SAT can be interpreted as a CSP instance where the domain is $\{0,1\}$ and the constraints in C correspond to the clauses (thus the arity of each constraint is 3). Another example is vertex coloring, which can be interpreted as a CSP instance where the variables correspond to the vertices, the domain corresponds to the set of colors, and there is a binary disequality constraint corresponding to each edge. Notice that the domain size can be arbitrarily large in the CSP instances arising from vertex coloring (as the coloring problem might involve any number of colors). In the present paper, we think of the domain as a set whose size is not a fixed constant, but can be be arbitrarily large. This viewpoint is natural in the context of various database query and artificial intelligence applications, where in fact that domain size is usually much larger than the number of variables [24, 41].

Due to its generality, solving constraint satisfaction problems is NP-hard if we do not impose any additional restrictions on the possible instances. Therefore, the main goal of the research on CSP is to identify tractable classes and special cases of the general problem. The theoretical literature on CSP investigates two main types of restrictions. The first type is to restrict the *constraint language,* that is, the type of constraints that is allowed. This direction was initiated by the classical work of Schaefer [42] and was subsequently pursued in, e. g., [7, 6, 5, 15, 31]. The second type is to restrict the *structure* induced by the constraints on the variables. The *primal graph* (or *Gaifman graph*) of a CSP instance is defined to be a graph on the variables of the instance such that there is an edge between two variables if and only if they appear together in some constraint. Freuder [21] observed that if the treewidth of the primal graph is k, then CSP can be solved in time $n^{O(k)}$. (Here n is the size of the input; in the cases we are interested in in this paper, the input size is polynomially bounded by the domain size and the number of variables.) The aim of this paper is to investigate whether there exists any other structural property of the primal graph that can be exploited algorithmically to speed up the search for the solution.

Structural complexity of CSP. The first question is to understand what graphs make CSP polynomial-time solvable. We have to be careful with the formalization of this question: if G is a graph with k vertices, then any CSP instance with primal graph G can be solved in time $n^{O(k)}$ by brute force. Therefore, restricting CSP to *any* fixed graph G makes it polynomial-time solvable. The real question is which *classes* of graphs make the problem polynomial-time solvable. Formally, for a class $\mathcal{G}$ of graphs, let CSP($\mathcal{G}$) be the class of all CSP instances where the primal graph of the instance is in $\mathcal{G}$. Note that this definition does not make any restriction on the constraint relations: it is possible that every constraint has a different constraint relation. If $\mathcal{G}$ has bounded treewidth, then CSP($\mathcal{G}$) is polynomial-time solvable. The converse is also true (under standard assumptions).

Theorem 1.1 (Grohe, Schwentick, Segoufin [29]; Grohe [26])**.** *If $\mathcal{G}$ is a recursively enumerable class of graphs, then CSP($\mathcal{G}$) is polynomial-time solvable if and only if $\mathcal{G}$ has bounded treewidth (assuming* FPT $\neq$ W[1]*).*

The results in [29, 26] are actually more general and are stated in terms of the conjunctive query and homomorphism problems (more on this in Section 5), but it is easy to see that those results imply Theorem 1.1. The assumption FPT $\neq$ W[1] is a standard hypothesis of parameterized complexity (cf. [14, 19]). Let us emphasize that the proof of Theorem 1.1 uses in an essential way the fact that the domain size can be arbitrarily large.

Remark 1.2. We have to define what is meant by saying that an algorithm $\mathbb{A}$ solves CSP($\mathcal{G}$) in polynomial time: what does $\mathbb{A}$ do on instances with $G \notin \mathcal{G}$? We can (1) require that $\mathbb{A}$ reject every instance with $G \notin \mathcal{G}$, or (2) require that $\mathbb{A}$ correctly solve every instance with $G \notin \mathcal{G}$, but not necessarily in polynomial-time, or (3) the behavior of $\mathbb{A}$ is not specified on instances with $G \in \mathcal{G}$ (so it can return an incorrect answer or may not even stop). The negative results are strongest if we rule out the possibility of the weakest form, namely (3) (this is the way Theorem 1.1 is stated in [26]). However, in this paper we state the results in a way that rule out the possibility of type (2) algorithms only. The reason why we do this is because negative results for type (3) algorithms seem to require the assumption that $\mathcal{G}$ is recursively enumerable, and if we make this assumption, then statements for type (2) and (3) algorithms become equivalent (see Corollary 4.5 at the end of Section 4). Thus our results apply to type (2) algorithms without any restriction on $\mathcal{G}$ and to type (3) algorithms with the additional assumption that $\mathcal{G}$ is recursively enumerable. In the formal negative statements of this paper, we will explicitely specify what type of algorithms are considered. To make the negative results stronger, we state them for the decision version of the problem.

By Theorem 1.1, bounded treewidth is the only property of the primal graph that can make the problem polynomial-time solvable. However, Theorem 1.1 does not rule out the possibility that there is some structural property that may enable us to solve instances significantly faster than the treewidth-based algorithm of [21], that is, for some class $\mathcal{G}$ of graphs with unbounded treewidth, CSP($\mathcal{G}$) could be solved in time $n^{f(k)}$ where k is the treewidth and f is a slowly growing function such as $\log k$. The main result of this paper is that this is not possible; the $n^{O(k)}$-time algorithm is essentially optimal for every class of graphs, up to an $O(\log k)$ factor in the exponent. Thus, in our specific setting, there is no other structural information beside treewidth that can be exploited algorithmically.

We prove our result under the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [30]: we assume that there is no $2^{o(n)}$-time algorithm for n-variable 3SAT. This assumption is stronger than FPT $\neq$ W[1]. The formal statement of the main result of the paper is the following (we denote by $\mathrm{tw}(G)$ the treewidth of G):

Theorem 1.3. *If there is a class $\mathcal{G}$ of graphs with unbounded treewidth, an algorithm $\mathbb{A}$, and a function f such that $\mathbb{A}$ correctly decides every binary CSP instance and the running time is $f(G)\|I\|^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))}$ for binary CSP($\mathcal{G}$) instances I with primal graph $G \in \mathcal{G}$, then ETH fails.*

Binary CSP($\mathcal{G}$) is the special case of CSP($\mathcal{G}$) where every constraint is binary, i. e., involves two variables. Note that adding this restriction makes the statement of Theorem 1.3 stronger. Similarly, allowing the multiplicative factor $f(G)$ in the running time also makes the result stronger. We do not make any assumption on f, for example, we do not require that f be computable.

The main technical tool of the proof of Theorem 1.1 in [29, 26] is the Excluded Grid Theorem of Robertson and Seymour [40], which states that there is an unbounded function $g(k)$ such that every graph with treewidth at least k contains a $g(k) \times g(k)$ grid as minor. The basic idea of the proof in [26] is to show that CSP($\mathcal{G}$) is not polynomial-time solvable if $\mathcal{G}$ contains every grid and then this result is used to argue that CSP($\mathcal{G}$) is not polynomial for any $\mathcal{G}$ with unbounded treewidth, since in this case $\mathcal{G}$ contains every grid as minor. However, this approach does not work if we want a tighter lower bound, as in Theorem 1.3. The problem is that the function $g(k)$ is very slowly growing, e. g., $o(\log k)$, in the known proofs of the Excluded Grid Theorem [12]. Therefore, if the only property of graphs with treewidth at least k that we use is that they have $g(k) \times g(k)$ grid minors, then we immediately lose a lot: as CSP on the $g(k) \times g(k)$ grid can be solved in time $\|I\|^{O(g(k))}$, no lower bound stronger than $\|I\|^{o(\log \mathrm{tw}(G))}$ can be proved with this approach. Thus we need a characterization of treewidth that is tighter than the Excluded Grid Theorem.

The almost-tight bound of Theorem 1.3 is made possible by a new characterization of treewidth that is tight up to a logarithmic factor (Theorem 3.1). This result may be of independent interest. We generalize the notion of minors the following way. An *embedding* of H into G is a mapping ψ from $V(H)$ to connected subsets of G such that if $u, v \in V(H)$ are adjacent, then either $\psi(u) \cap \psi(v) \neq \emptyset$ or there is an edge connecting a vertex of $\psi(u)$ and a vertex of $\psi(v)$. The *depth* of the embedding is at most q if every vertex of G appears in the images of at most q vertices of H. Thus H has an embedding of depth 1 into G if and only if H is a minor of G.

We characterize treewidth by the "embedding power" of the graph in the following sense. If q is sufficiently large, then H has an embedding of depth q into G. For example, $q = |V(H)| \leq 2|E(H)|$ (assuming H has no isolated vertices) is certainly sufficient. However, we show that if the treewidth of G is at least k, then there is an embedding with depth $q = O(|E(H)| \log k/k)$, i. e., the depth is a factor $O(k/\log k)$ better than in the trivial bound of $2|E(H)|$. We prove this result using the well-known characterization of treewidth by separators and an $O(\log k)$ integrality gap bound for the sparsest cut. The main idea of the proof of Theorem 1.3 is to use the embedding power of a graph with large treewidth to simulate a 3SAT instance efficiently.

We conjecture that Theorem 1.3 can be improved by removing the $\log \mathrm{tw}(G)$ factor from the exponent; this will make the result tight.

Conjecture 1.4. There is no class $\mathcal{G}$ of graphs with unbounded treewidth, no function f, and no algorithm $\mathbb{A}$ such that an algorithm $\mathbb{A}$ correctly decides every binary CSP instance and the running time is $f(G)\|I\|^{o(\mathrm{tw}(G))}$ for binary CSP($\mathcal{G}$) instances I with primal graph $G \in \mathcal{G}$.

This seemingly minor improvement would be very important for classifying the complexity of other CSP variants [35]. However, it seems that a much better understanding of treewidth will be required before Theorem 1.3 can be made tight. At the very least, it should be settled whether there is a polynomial-time constant-factor approximation algorithm for treewidth.

The homomorphism problem. A large part of the theoretical literature on CSP follows the notation introduced by Feder and Vardi [15] and formulates the problem as a homomorphism between relational structures. This more general framework allows a clean algebraic treatment of many issues. In Section 5, we translate the lower bound of Theorem 1.3 into this framework (Theorem 5.1) to obtain a quantitative version of the main result of Grohe in [26]. That is, the left-hand side classes of structures in the homomorphism problem are not only characterized with respect to polynomial-time solvability, but we prove almost-tight lower bounds on the exponent of the running time. As a special case, Theorem 5.1 immediately implies a generalization of Theorem 1.3 from binary CSP to constraints with any fixed finite arity: for every fixed $r \geq 2$, it can be used to give a lower bound on the running time of r-ary CSP when restricted to a family of r-uniform hypergraphs.

As observed by Grohe in [26], the complexity of the homomorphism problem does not depend directly on the treewidth of the left-hand side structure, but rather on the treewidth of its core. Thus the treewidth of the core appears in Theorem 5.1, the analog of Theorem 1.3. The reason why the notion of core is irrelevant in Theorem 1.3 is that the way we defined CSP($\mathcal{G}$) allows the possibility that every constraint relation appearing in the instance is different. In such a case, a nontrivial homomorphism of the primal graph does not provide any apparent shortcut for solving the problem. Similarly to [26], our result applies only if the left-hand side structure has bounded arity. In the unbounded-arity case, issues related to the representation of the structures arise, which change the problem considerably. The homomorphism problem with unbounded arity is far from understood: new classes of tractable structures have recently been identified [27, 36, 37].

Subgraph problems. Tight lower bounds on the exponent under ETH have previously been obtained in the framework of parameterized complexity. A basic result in this direction is due to Chen et al.:

Theorem 1.5 ([9, 10])**.** *There is no $f(k) \cdot n^{o(k)}$-time algorithm for k-Clique, unless ETH fails.*

For a number of problems parameterized by clique width, tight bounds on the exponent of the running time were given by Fomin et al. [20]. The Closest Substring problem was studied in [34], and it was shown that in two specific settings, there are no algorithms with $o(\log k)$ and $o(\log\log k)$ in the exponents of their respective running times (unless ETH fails), and there are algorithms matching these lower bounds. The class M[1] was introduced as a tool that uses ETH to provide an alternative way of proving hardness in parameterized complexity [13, 18].

Theorem 1.5 can be interpreted as a lower bound for the Subgraph Isomorphism problem (given two graphs G and H, decide if G is a subgraph of H). Using the color coding technique of Alon, Yuster, and Zwick [2], it is possible to solve Subgraph Isomorphism in time $f(|V(G)|) \cdot n^{O(\mathrm{tw}(G))}$. Theorem 1.5 and the fact that the treewidth of the k-clique is $k-1$ shows that it is not possible to improve the dependence on $\mathrm{tw}(G)$ in the exponent to $o(\mathrm{tw}(G))$, since in particular this would imply an $f(k) \cdot n^{o(k)}$-time algorithm for the k-Clique problem. However, this observation does not rule out the possibility that there is a special class of graphs (say, bounded degree graphs or planar graphs) where it is possible to improve the exponent to $o(\mathrm{tw}(G))$. In Section 6, we discuss lower bounds for Subgraph Isomorphism (and its colored version) that follow from our CSP results.

Another important aspect of Theorem 1.5 is that it can be used to obtain lower bounds for other parameterized problems. W[1]-hardness proofs are typically done by parameterized reductions from k-Clique. It is easy to observe that a parameterized reduction implies a lower bound similar to Theorem 1.5 for the target problem, with the exact form of the lower bound depending on the way the reduction changes the parameter. Many of the more involved reductions use edge selection gadgets (see e.g., [17]). As the k-clique has $\Theta(k^2)$ edges, this means that the reduction increases the parameter to $\Theta(k^2)$ and we can conclude that there is no $f(k) \cdot n^{o(\sqrt{k})}$-time algorithm for the target problem (unless ETH fails). If we want to obtain stronger bounds on the exponent, then we have to avoid the quadratic blow-up of the parameter and do the reduction from a different problem. One possibility is to reduce from Subgraph Isomorphism, parameterized by the number of edges. In a reduction from Subgraph Isomorphism, we need $|E(G)|$ edge selection gadgets, which usually implies that the new parameter is $\Theta(|E(G)|)$.

Therefore, the reduction and the following corollary obtained in Section 6 allows us to conclude that there is no $f(k) \cdot n^{o(k/\log k)}$-time algorithm for the target problem:

Corollary 1.6. *If Subgraph Isomorphism can be solved in time $f(k)n^{o(k/\log k)}$, where f is an arbitrary function and $k = |E(G)|$ is the number of* edges *of the smaller graph G, then ETH fails.*

Organization. Section 2 summarizes the notation we use. Section 3 presents the new characterization of treewidth. Section 4 treats binary CSP and proves Theorem 1.3. Section 5 gives an overview of the homomorphism problem and presents the main result in this context. In Section 6, we obtain hardness results for subgraph problems as corollaries to the main result.

2 Preliminaries

Constraint satisfaction problems. We briefly recall some terminology related to CSP. For more background, see e. g., [25, 15].

Definition 2.1. An instance of a *constraint satisfaction problem* is a triple (V,D,C), where:

- V is a set of variables,
- D is a domain of values,
- C is a set of constraints, $\{c_1,c_2,\ldots,c_q\}$. Each constraint $c_i \in C$ is a pair $\langle s_i,R_i\rangle$, where:
  - s_i is a tuple of variables of length m_i, called the *constraint scope,* and
  - R_i is an m_i-ary relation over D, called the *constraint relation.*

For each constraint $\langle s_i,R_i\rangle$ the tuples of R_i indicate the allowed combinations of simultaneous values for the variables in s_i. The length m_i of the tuple s_i is called the *arity* of the constraint. We allow repeated variables in the scope s_i, but this does not make the problem more general and can be usually ignored. A *solution* to a constraint satisfaction problem instance is a function f from the set of variables V to the domain D of values such that for each constraint $\langle s_i,R_i\rangle$ with $s_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_m})$, the tuple $(f(v_{i_1}), f(v_{i_2}), \ldots, f(v_{i_m}))$ is a member of R_i. In the decision version of CSP, we have to decide if a solution for the given instance I exists. We say that an instance is *binary* if each constraint relation is binary, i. e., $m_i = 2$ for every constraint.[1] In this paper, we consider only binary instances. It can be assumed that the instance does not contain two constraints $\langle s_i,R_i\rangle$, $\langle s_j,R_j\rangle$ with $s_i = s_j$, since in this case the two constraints can be replaced by the constraint $\langle s_i,R_i \cap R_j\rangle$.

In the input, the relation in a constraint is represented by listing all the tuples of the constraint. We denote by $\|I\|$ the size of the representation of the instance $I = (V,D,C)$. For binary constraint satisfaction problems, we may assume that $\|I\| = O(V^2D^2)$; by the argument in the previous paragraph, we may assume that there are $O(V^2)$ constraints and each constraint has a representation of length

[1] It is unfortunate that while some communities use the term "binary CSP" in the sense that each constraint is binary (as does this paper), others use it in the sense that the variables are 0-1, i. e., the domain size is 2.

$O(D^2)$. Furthermore, it can be assumed that $|D| \le \|I\|$; elements of D that do not appear in any relation can be removed.

Let $I = (V, D, C)$ be a CSP instance and let $V' \subseteq V$ be a nonempty subset of variables. The instance *induced* by V' is the CSP instance $I[V'] = (V', D, C')$, where $C' \subseteq C$ is the set of constraints whose scope is contained in V'. Clearly, if f is a solution of I, then f restricted to V' is a solution of $I[V']$.

The *primal graph* of a CSP instance $I = (V, D, C)$ is a graph G with vertex set V, where $x, y \in V$ form an edge if and only if there is a constraint $\langle s_i, R_i \rangle \in C$ with $x, y \in s_i$. For a class $\mathcal{G}$ of graphs, we denote by CSP($\mathcal{G}$) the problem restricted to instances where the primal graph is in $\mathcal{G}$.

Graphs. We denote by $V(G)$ and $E(G)$ the set of vertices and the set of edges of the graph G, respectively. Given a graph G, the *line graph* $L(G)$ has one vertex for each edge of G, and two vertices of $L(G)$ are connected if and only if the corresponding edges in G share an endpoint. The line graph $L(K_k)$ of the complete graph K_k will appear repeatedly in the paper. Usually we denote the vertices of $L(K_k)$ by $v_{\{i,j\}}$ ($1 \le i < j \le k$), where $v_{\{i_1,j_1\}}$ and $v_{\{i_2,j_2\}}$ are adjacent if and only if $\{i_1, j_1\} \cap \{i_2, j_2\} \neq \emptyset$.

A *tree decomposition* of a graph G is a tuple $(T, (B_t)_{t \in V(T)})$, where T is a tree and $(B_t)_{t \in V(T)}$ is a family of subsets of $V(G)$ such that for each $e \in E(G)$ there is a node $t \in V(T)$ such that $e \subseteq B_t$, and for each $v \in V(G)$ the set $\{t \in V(T) \mid v \in B_t\}$ is connected in T. That is, we represent the graph G as a subgraph of the intersection graph of subtrees of the tree T. The sets B_t are called the *bags* of the decomposition. The *width* of a tree-decomposition $(T, (B_t)_{t \in V(T)})$ is $\max\left\{|B_t| \mid t \in V(t)\right\} - 1$. The *treewidth* $\operatorname{tw}(G)$ of a graph G is the minimum of the widths of all tree decompositions of G. A class $\mathcal{G}$ of graphs is of *bounded treewidth* if there is a constant c such that $\operatorname{tw}(G) \le c$ for every $G \in \mathcal{G}$. The concept of treewidth was introduced by Robertson and Seymour [39] and played a fundamental role in the theory of graph minors. For more background on treewidth and its applications, the reader is referred to [4, 32, 3].

Minors and embeddings. A graph H is a *minor* of G if H can be obtained from G by a sequence of vertex deletions, edge deletions, and edge contractions. The following alternative definition will be more relevant to our purposes. An *embedding* of H into G is a mapping ψ from $V(H)$ to connected subsets of G such that if $u, v \in V(H)$ are adjacent, then either $\psi(u) \cap \psi(v) \neq \emptyset$ or there is an edge connecting a vertex of $\psi(u)$ and a vertex of $\psi(v)$. The *depth* of a vertex v of G is the size of the set $\{u \in V(H) \mid v \in \psi(u)\}$ and the depth of the embedding is the maximum of the depths of the vertices. It is easy to see that H is a minor of G if and only if H has an embedding of depth 1 into G, i. e., the images are disjoint. To emphasize this connection, we will say that an embedding of depth 1 is a *minor mapping*.

In an equivalent way, we can use minors to define embeddings of a certain depth. Given a graph G and an integer q, we denote by $G^{(q)}$ the graph obtained by replacing every vertex with a clique of size q and replacing every edge with a complete bipartite graph on $q + q$ vertices. It is easy to see that H has an embedding of depth q into G if and only if H is a minor of $G^{(q)}$. The mapping ϕ that maps each vertex of G to the corresponding clique of $G^{(q)}$ will be called the *blow-up* mapping from G to $G^{(q)}$.

3 Embedding in a graph with large treewidth

If H is a graph with n vertices, then obviously H has an embedding of depth n into any (nonempty) G. If G has a clique of size k, then there is an embedding with depth at most n/k. Furthermore, even if G does not have a k-clique subgraph, but it does have a k-clique minor, then there is such an embedding with depth at most n/k. Thus a k-clique minor increases the "embedding power" of a graph by a factor of k. The main result of the section is that large treewidth implies a similar increase in embedding power. The following lemma states this formally:

Theorem 3.1. *There are computable functions $f_1(G)$, $f_2(G)$, and a universal constant c such that for every $k \ge 1$, if G is a graph with $\mathrm{tw}(G) \ge k$ and H is a graph with $|E(H)| = m \ge f_1(G)$ and no isolated vertices, then H has an embedding into G with depth at most $\lceil cm \log k/k \rceil$. Furthermore, such an embedding can be found in time $f_2(G)m^{O(1)}$.*

Using the equivalent characterization by minors, the conclusion of Theorem 3.1 means that H is a minor of $G^{(q)}$ for $q = \lceil cm \log k/k \rceil$. In the rest of the paper, we mostly use this notation.

The value $cm \log k/k$ is optimal up to an $O(\log k)$ factor, i. e., it cannot be improved to $o(m/k)$. To see this, observe first that $\mathrm{tw}(G^{(q)}) = \Theta(q \cdot \mathrm{tw}(G))$ (cf. [28]). We use the fact that the treewidth of a graph H with m edges can be $\Omega(m)$ (e. g., bounded-degree expanders). Therefore, if $\mathrm{tw}(G) = k$, then the treewidth of $G^{(q)}$ for $q = o(m/k)$ is $o(m)$, making it impossible that H is a minor of $G^{(q)}$. Furthermore, Theorem 3.1 does not remain true if m is the number of vertices of H (instead of the number of edges). Let H be a clique on m vertices, and let G be a bounded-degree graph on $O(k)$ vertices with treewidth k. It is easy to see that $G^{(q)}$ has $O(q^2k)$ edges, hence H can be a minor of $G^{(q)}$ only if $q^2k = \Omega(m^2)$, that is, $q = \Omega(m/\sqrt{k})$. Note that it makes no sense to state in this form an analog of Theorem 3.1 where m is the number of vertices of H: the worst case happens if H is an m-clique, and the theorem would become a statement about embedding cliques. The requirement $m \ge f_1(G)$ is a technical detail: some of the arguments in the embedding technique requires H to be large.

The graph $L(K_k)$, i. e., the line graph of the complete graph plays a central role in the proof of Theorem 3.1. The proof consists of two parts. In the first part (Section 3.1), we show that if $\mathrm{tw}(G) \ge k$, then a blow-up of $L(K_k)$ is a minor of an appropriate blow-up of G. This part of the proof is based on the characterization of treewidth by balanced separators and uses a result of Feige et al. [16] on the linear programming formulation of separation problems. Similar ideas were used in [28]; some of the arguments are reproduced here for the convenience of the reader. In the second part (Section 3.2), we show that every graph is a minor of an appropriate blow-up of $L(K_k)$.

3.1 Embedding $L(K_k)$ in G

Given a nonempty set W of vertices, we say that a set S of vertices is a *balanced separator* (with respect to W) if $|W \cap C| \le |W|/2$ for every connected component C of $G \setminus S$. A k-separator is a separator S with $|S| \le k$. The treewidth of a graph is closely connected with the existence of balanced separators:

Lemma 3.2 ([38], [19, Section 11.2])**.**

(a) If the graph G has treewidth greater than $3k$, then there is a set $W \subseteq V(G)$ of size $2k+1$ having no balanced k-separator.

(b) If the graph G has treewidth at most k, then every $W \subseteq V(G)$ has a balanced $(k+1)$-separator.

A *separation* is a partition of the vertices into three classes (A,B,S) ($S \neq \emptyset$) such that there is no edge between A and B. Note that it is possible that $A = \emptyset$ or $B = \emptyset$. The *sparsity* of the separation (A,B,S) (with respect to W) is defined in [16] as

$$\alpha^W(A,B,S) = \frac{|S|}{|(A \cup S) \cap W| \cdot |(B \cup S) \cap W|}. \tag{3.1}$$

We denote by $\alpha^W(G)$ the minimum of $\alpha^W(A,B,S)$ taken over every separation (A,B,S). It is easy to see that for every G and nonempty W, $1/|W|^2 \leq \alpha^W(G) \leq 1/|W|$ (the second inequality follows from the fact that the separation $(V(G) \setminus W, \emptyset, W)$ has sparsity exactly $1/|W|$). For our applications, we need a set W such that $\alpha^W(G)$ is close to the maximum possible, i. e., $\Omega(1/|W|)$. The following lemma shows that the non-existence of a balanced separator can guarantee the existence of such a set W. The connection between balanced separators and sparse separations is well known, see for example [16, Section 6]. However, in our parameter setting a simpler argument is sufficient.

Lemma 3.3. *If $|W| = 2k+1$ and W has no balanced k-separator in a graph G, then $\alpha^W(G) \geq 1/(4k+1)$.*

Proof. Let (A,B,S) be a separation of sparsity $\alpha^W(G)$; we may assume that $|A \cap W| \geq |B \cap W|$, hence $|B \cap W| \leq k$. If $|S| > k$, then

$$\alpha^W(A,B,S) \geq \frac{k+1}{(2k+1)^2} \geq \frac{1}{4k+1}.$$

If $|S| \geq |(B \cup S) \cap W|$, then

$$\alpha^W(A,B,S) \geq \frac{1}{|(A \cup S) \cap W|} \geq \frac{1}{2k+1}.$$

Assume therefore that $|(B \cup S) \cap W| \geq |S| + 1$. Let S' be a set of $k - |S| \geq 0$ arbitrary vertices of $W \setminus (B \cup S)$. We claim that $S \cup S'$ is a balanced k-separator of W. Suppose that there is a component C of $G \setminus (S \cup S')$ that contains more than k vertices of W. Component C is either a subset of A or a subset of B. However, C cannot be a subset of B, since $|B \cap W| \leq k$. On the other hand,

$$|(A \setminus S') \cap W| \leq 2k+1 - |(B \cup S) \cap W| - |S'| \leq 2k+1 - (|S|+1) - (k - |S|) \leq k.$$

□

Remark 3.4. Lemma 3.3 does not remain true in this form for larger W. For example, let K be a clique of size $3k+1$, let us attach k degree-one vertices to a distinguished vertex x of K, and let us attach a degree-one vertex to every other vertex of K. Let W be the set of these $4k$ degree-one vertices. It is not difficult to see that W has no balanced k-separator. On the other hand, $S = \{x\}$ is a separator with sparsity $1/(k \cdot 3k)$, hence $\alpha^W(G) = O(1/k^2)$.

Let $W = \{w_1, \ldots, w_r\}$ be a set of vertices. A *concurrent vertex flow of value* ε is a collection of $|W|^2$ flows such that for every ordered pair $(u,v) \in W \times W$, there is a flow of value ε between u and v, and the total amount of flow going through each vertex is at most 1. A *flow* between u and v is a weighted collection of $u - v$ paths. A $u - v$ path contributes to the load of vertex u, of vertex v, and of every vertex between u and v on the path. In the degenerate case when $u = v$, vertex $u = v$ is the only vertex where the flow between u and v goes through, that is, the flow contributes to the load of only this vertex.

The maximum concurrent vertex flow can be expressed as a linear program the following way. For $u, v \in W$, let $\mathcal{P}_{uv}$ be the set of all $u - v$ paths in G, and for each $p \in \mathcal{P}_{uv}$, let variable $p^{uv} \geq 0$ denote the amount of flow that is sent from u to v along p. Consider the following linear program:

$$\begin{aligned} &\text{maximize } \varepsilon \quad \text{s. t.} \\ &\sum_{p \in \mathcal{P}_{uv}} p^{uv} \geq \varepsilon && \forall u, v \in W \\ &\sum_{(u,v) \in W \times W} \sum_{p \in \mathcal{P}_{uv} : w \in p} p^{uv} \leq 1 && \forall w \in V \\ &p^{uv} \geq 0 && \forall u, v \in W, p \in \mathcal{P}_{uv} \end{aligned} \tag{LP1}$$

The dual of this linear program can be written with variables $\{\ell_{uv}\}_{u,v \in W}$ and $\{s_v\}_{v \in V}$ the following way:

$$\begin{aligned} &\text{minimize } \sum_{v \in V} s_v \quad \text{s. t.} \\ &\sum_{w \in p} s_w \geq \ell_{uv} && \forall u, v \in W, p \in \mathcal{P}_{uv} \ (*) \\ &\sum_{(u,v) \in W \times W} \ell_{uv} \geq 1 && (**) \\ &\ell_{uv} \geq 0 && \forall u, v \in W \\ &s_w \geq 0 && \forall w \in V \end{aligned} \tag{LP2}$$

We show that, in some sense, (LP2) is the linear programming relaxation of finding a separator with minimum sparsity. If there is a separation (A, B, S) with sparsity $\alpha^W(A, B, S)$, then (LP2) has a solution with value at most $\alpha^W(A, B, S)$. Set $s_v = \alpha^W(A, B, S)/|S|$ if $v \in S$ and $s_v = 0$ otherwise; the value of such a solution is clearly $\alpha^W(A, B, S)$. For every $u, v \in W$, set $\ell_{uv} = \min_{p \in \mathcal{P}_{uv}} \sum_{w \in p} s_w$ to ensure that inequalities (*) hold. To see that (**) holds, notice first that $\ell_{uv} \geq \alpha^W(A, B, S)/|S|$ if $u \in A \cup S$, $v \in B \cup S$, as every $u - v$ path has to go through at least one vertex of S. Furthermore, if $u, v \in S$ and $u \neq v$, then $\ell_{uv} \geq 2\alpha^W(A, B, S)/|S|$ since in this case a $u - v$ paths meets S in at least two vertices. The expression $|(A \cup S) \cap W| \cdot |(B \cup S) \cap W|$ counts the number of ordered pairs (u, v) satisfying $u \in (A \cup S) \cap W$ and $v \in (B \cup S) \cap W$, such that pairs with $u, v \in S \cap W$, $u \neq v$ are counted twice. Therefore,

$$\sum_{(u,v) \in W \times W} \ell_{uv} \geq (|(A \cup S) \cap W| \cdot |(B \cup S) \cap W|) \cdot \frac{\alpha^W(A, B, S)}{|S|} = 1,$$

which means that inequality (**) is satisfied.

The other direction is not true: a solution of (LP2) with value α does not imply that there is a separation with sparsity at most α. However, Feige et al. [16] proved that it is possible to find a separation whose sparsity is greater than that by at most a $O(\log |W|)$ factor (this result appears implicitly already in [33]):

Theorem 3.5 (Feige et al. [16], Leighton and Rao [33]). *If (LP2) has a solution with value α, then there is a separation with sparsity $O(\alpha \log |W|)$.*

We use (the contrapositive of) Theorem 3.5 to obtain a concurrent vertex flow in a graph with large treewidth. This concurrent vertex flow can be used to find an $L(K_k)$ minor in the blow-up of the graph in a natural way: the flow paths correspond to the edges of K_k.

Lemma 3.6. *Let G be a graph with $\mathrm{tw}(G) > 3k$. There are universal constants $c_1, c_2 > 0$ such that $L(K_k)^{(\lceil c_1 \log n \rceil)}$ is a minor of $G^{(\lceil c_2 \log n \cdot k \log k \rceil)}$, where n is the number of vertices of G.*

Proof. Since G has treewidth greater than $3k$, by Lemma 3.2(a), there is a subset W_0 of size $2k+1$ that has no balanced k-separator. By Lemma 3.3, $\alpha^{W_0}(G) \geq 1/(4k+1) \geq 1/(5k)$. Therefore, Theorem 3.5 implies that the dual linear program (LP2) has no solution with value less than $1/(c_0 5k \log(2k+1))$, where c_0 is the constant hidden by the big O notation in Theorem 3.5. By linear programming duality, there is a concurrent flow of value at least $\alpha := 1/(c_0 5k \log(2k+1))$ connecting the vertices of W_0; let p^{uv} be a corresponding solution of (LP1).

Let $W \subseteq W_0$ be a subset of k vertices. For each pair of vertices $(u,v) \in W \times W$, let us randomly and independently choose $\lceil \ln n \rceil$ paths $P_{u,v,1}, \ldots, P_{u,v,\lceil \ln n \rceil}$ of $\mathcal{P}_{uv}$ (here ln denotes the natural logarithm of n), where path p is chosen with probability

$$\frac{p^{uv}}{\sum_{p' \in \mathcal{P}_{uv}} (p')^{uv}} \leq \frac{p^{uv}}{\alpha}. \tag{3.2}$$

That is, we scale the values p^{uv} to obtain a probability distribution. Inequality (3.2) is true because the values p^{uv} satisfy (LP1). The expected number of times a path $p \in \mathcal{P}_{uv}$ is selected is

$$\lceil \ln n \rceil \cdot \frac{p^{uv}}{\sum_{p' \in \mathcal{P}_{uv}} (p')^{uv}} \leq \lceil \ln n \rceil \cdot \frac{p^{uv}}{\alpha}.$$

Thus the expected number of paths selected from $\mathcal{P}_{uv}$ that go through a vertex w is at most $\lceil \ln n \rceil \cdot \sum_{p \in \mathcal{P}_{uv} : w \in p} p^{uv}/\alpha$. Considering that we select $\lceil \ln n \rceil$ paths for every pair $(u,v) \in W \times W$, the expected number μ_w of selected paths containing w is at most

$$\lceil \ln n \rceil \cdot \sum_{(u,v) \in W \times W} \sum_{p \in \mathcal{P}_{uv} : w \in p} p^{uv}/\alpha,$$

which is at most $\lceil \ln n \rceil/\alpha$, since the values p^{uv} satisfy (LP1). We use the following standard Chernoff bound: for every $r > \mu_w$, the probability that more than $\mu_w + r$ of the $k^2 \ln n$ paths contain vertex w is at most $(\mu_w e/r)^r$. Thus the probability that more than $\mu_w + 10\lceil \ln n \rceil/\alpha \leq 11\lceil \ln n \rceil/\alpha$ of the paths contain w is at most

$$\left(\frac{\mu_w e}{10\lceil \ln n \rceil/\alpha}\right)^{10\lceil \ln n \rceil/\alpha} \leq (1/e)^{10 \ln n} = 1/n^{10}$$

(in the exponent, we used $\lceil \ln n \rceil/\alpha \geq \ln n$, since it can be assumed that $c_0 \geq 1$ and $\ln n \geq 1$). Therefore, with probability at least $1 - 1/n$, each vertex w is contained in at most $q := 11\lceil \ln n/\alpha \rceil$ paths. Note that $q \leq \lceil c_2 \log n \cdot k \log k \rceil$, for an appropriate value of c_2.

Let ϕ be the blow-up mapping from G to $G^{(q)}$. For each path $P_{u,v,i}$ in G, we define a path $P'_{u,v,i}$ in $G^{(q)}$. Let $P_{u,v,i} = p_1 p_2 \dots p_r$. The path $P'_{u,v,i}$ we define consists of one vertex of $\phi(p_1)$, followed by one vertex of $\phi(p_2)$, ..., followed by one vertex of $\phi(p_r)$. The vertices are selected arbitrarily from these sets, the only restriction is that we do not select a vertex of $G^{(q)}$ that was already assigned to some other path $P'_{u',v',i'}$. Since each vertex w of G is contained in at most q paths, the q vertices of $\phi(w)$ are sufficient to satisfy all the paths going through w. Therefore, we can ensure that the $k^2\lceil \ln n\rceil$ paths $P'_{u,v,i}$ are pairwise disjoint in $G^{(q)}$.

The minor mapping from $L(K_k)^{(\lceil \ln n\rceil)}$ to $G^{(q)}$ is defined as follows. Let ψ be the blow-up mapping from $L(K_k)$ to $L(K_k)^{(\lceil \ln n\rceil)}$, and let $v_{\{1,2\}}$, $v_{\{1,3\}}$ ..., $v_{\{k-1,k\}}$ be the $\binom{k}{2}$ vertices of $L(K_k)$, where $v_{\{i_1,i_2\}}$ and $v_{\{j_1,j_2\}}$ are adjacent if and only if $\{i_1,i_2\}\cap\{j_1,j_2\}\neq\emptyset$. Let $W=\{w_1,\dots,w_k\}$. The $\lceil \ln n\rceil$ vertices of $\psi(v_{i,j})$ are mapped to the $\lceil \ln n\rceil$ paths

$$P'_{w_i,w_j,1},\dots,P'_{w_i,w_j,\lceil \ln n\rceil}\,.$$

Clearly, the images of the vertices are disjoint and connected. We have to show that this minor mapping maps adjacent vertices to adjacent sets. If $x\in\psi(v_{i_1,i_2})$ and $x'\in\psi(v_{j_1,j_2})$ are connected in $L(K_k)^{(\lceil \ln n\rceil)}$, then there is a $t\in\{i_1,i_2\}\cap\{j_1,j_2\}$. This means that the paths corresponding to x and x' both contain a vertex of the clique $\phi(w_t)$ in $G^{(q)}$, which implies that there is an edge connecting the two paths. □

With the help of the following proposition, we can make a small improvement on Lemma 3.6: the assumption $\mathrm{tw}(G) > 3k$ can be replaced by the assumption $\mathrm{tw}(G)\geq k$. This will make the result more convenient to use.

Proposition 3.7. *For every $k\geq 3$, $q\geq 1$, $L(K_{qk})$ is a subgraph of $L(K_k)^{(2q^2)}$.*

Proof. Let ϕ be a mapping from $\{1,\dots,qk\}$ to $\{1,\dots,k\}$ such that exactly q elements of $\{1,\dots,qk\}$ are mapped to each element of $\{1,\dots,k\}$. Let

$$v_{\{i_1,i_2\}} \quad (1\leq i_1 < i_2 \leq qk)$$

be the vertices of $L(K_{qk})$ and

$$u^t_{\{i_1,i_2\}} \quad (1\leq i_1<i_2\leq k, 1\leq t\leq 2q^2)$$

be the vertices of $L(K_k)^{(2q^2)}$, with the usual convention that two vertices are adjacent if and only if the lower indices are not disjoint. Let $U_{\{i_1,i_2\}}$ be the clique $\{u^t_{\{i_1,i_2\}} \mid 1\leq t\leq 2q^2\}$. Let us consider the vertices of $L(K_{qk})$ in some order. If $\phi(i_1)\neq\phi(i_2)$, then vertex $v_{\{i_1,i_2\}}$ is mapped to a vertex of $U_{\{\phi(i_1),\phi(i_2)\}}$ that was not already used for a previous vertex. If $\phi(i_1)=\phi(i_2)$, then $v_{\{i_1,i_2\}}$ is mapped to a vertex $U_{\{\phi(i_1),\phi(i_1)+1\}}$ (where addition is modulo k). It is clear that if two vertices of $L(K_{qk})$ are adjacent, then the corresponding vertices of $L(K_k)^{(2q^2)}$ are adjacent as well. We have to verify that, for a given i_1,i_2, at most $2q^2$ vertices of $L(K_{qk})$ are mapped to the clique $U_{\{i_1,i_2\}}$. As $|\phi^{-1}(i_1)|$ and $|\phi^{-1}(i_2)|$ are both q, there are at most q^2 vertices $v_{\{j_1,j_2\}}$ with $\phi(j_1)=i_1$, $\phi(j_2)=i_2$. Furthermore, if $i_2=i_1+1$, then there are $\binom{q}{2}\leq q^2$ additional vertices $v_{\{j_1,j_2\}}$ with $\phi(j_1)=\phi(j_2)=i_1$ that are also mapped to $U_{\{i_1,i_2\}}$. Thus at most $2q^2$ vertices are mapped to each clique $U_{\{i_1,i_2\}}$. □

Set $k' := 3k+1 \leq 4k$. Using Proposition 3.7 with $q = 4$, we get that $L(K_{k'})^{(\lceil c_1 \log n \rceil/32)}$ is a subgraph of $L(K_k)^{(\lceil c_1 \log n \rceil)}$. Thus if $\mathrm{tw}(G) \geq k'$, then we can not only find a blowup of $L(K_k)$, but even a blowup of $L(K_{k'})$. By replacing k' with k, Lemma 3.6 can be improved the following way:

Lemma 3.8. *Let G be a graph with $\mathrm{tw}(G) \geq k$. There are universal constants $c_1, c_2 > 0$ such that $L(K_k)^{(\lceil c_1 \log n \rceil)}$ is a minor of $G^{(\lceil c_2 \log n \cdot k \log k \rceil)}$, where n is the number of vertices of G.* □

3.2 Embedding H in $L(K_k)$

As the second step of the proof of Theorem 3.1, we show that every (sufficiently large) graph H is a minor of $L(K_k)^{(q)}$ for $q = O(|E(H)|/k^2)$.

Lemma 3.9. *For every $k > 1$ there is a constant $n_k = O(k^4)$ such that for every G with $|E(G)| > n_k$ and no isolated vertices, the graph G is a minor of $L(K_k)^{(q)}$ for $q = \lceil 130|E(G)|/k^2 \rceil$. Furthermore, a minor mapping can be found in time polynomial in q and the size of G.*

Proof. We may assume that $k \geq 5$: otherwise the result is trivial, as $q \geq 2|E(G)| \geq |V(G)|$ and $L(K_k)^{(q)}$ contains a clique of size q. First we construct a graph G' of maximum degree 3 that contains G as a minor. This can be achieved by replacing every vertex v of G with a path on $d(v)$ vertices (where $d(v)$ is the degree of v in G); now we can ensure that the edges incident to v use distinct copies of v from the path. The new graph G' has exactly $2|E(G)|$ vertices.

We show that G', hence G, is a minor of $L(K_k)^{(q)}$. Take an arbitrary partition of $V(G')$ into $\binom{k}{2}$ classes $V_{\{i,j\}}$ ($1 \leq i < j \leq k$) such that $|V_{\{i,j\}}| \leq \lceil |V|/\binom{k}{2} \rceil$ for every i, j. Let $v_{\{i,j\}}$ ($1 \leq i < j \leq k$) be the vertices of $L(K_k)$, and let ϕ be the blow-up mapping from $L(K_k)$ to $L(K_k)^{(q)}$.

The minor mapping ψ from G' to $L(K_k)^{(q)}$ is defined the following way. First, if $u \in V_{\{i,j\}}$, then let $\psi(u)$ contain a vertex $\hat{u}$ from $\phi(v_{\{i,j\}})$. Observe that if edge e connects vertices $u_1 \in V_{\{i_1,j_1\}}$, $u_2 \in V_{\{i_2,j_2\}}$ and $\{i_1, j_1\} \cap \{i_2, j_2\} \neq \emptyset$ holds, then $\hat{u}_1$ and $\hat{u}_2$ are adjacent. In order to ψ be a minor mapping, we extends the sets $\psi(u)$ to ensure that the endpoints of e are mapped to adjacent sets even if $V_{\{i_1,j_1\}}$ and $V_{\{i_2,j_2\}}$ have disjoint indices.

Fix an arbitrary orientation of each edge of G'. For every quadruple (i_1, j_1, i_2, j_2) of distinct values with $i_1 < j_1$, $i_2 < j_2$, let E_{i_1,j_1,i_2,j_2} be the set of edges going from a vertex of $V_{\{i_1,j_1\}}$ to a vertex of $V_{\{i_2,j_2\}}$. Let us partition the set E_{i_1,j_1,i_2,j_2} into $k-4$ classes $E^{\ell}_{i_1,j_1,i_2,j_2}$ ($\ell \in \{1,\dots k\} \setminus \{i_1, j_1, i_2, j_2\}$) in an arbitrary way such that $|E^{\ell}_{i_1,j_1,i_2,j_2}| \leq \lceil |E_{i_1,j_1,i_2,j_2}|/(k-4) \rceil$. For each edge $\overrightarrow{uw} \in E^{\ell}_{i_1,j_1,i_2,j_2}$, we add a vertex of $\phi(v_{\{i_1,\ell\}})$ to $\psi(u)$ and a vertex of $\phi(v_{\{i_2,\ell\}})$ to $\psi(w)$; these two vertices are neighbors with each other and they are adjacent to $\hat{u}$ and $\hat{w}$, respectively. This ensures that $\psi(u)$ and $\psi(v)$ remain connected and there is an edge between $\psi(u)$ and $\psi(w)$. After repeating this step for every edge, ψ is clearly a minor mapping.

What remains to be shown is that the sets $\phi(v_{\{x,y\}})$ are large enough so that we can ensure that no vertex of $L(K_k)^{(q)}$ is assigned to more than one $\psi(u)$. Let us count how many vertices of $\phi(v_{\{x,y\}})$ are used when the minor mapping is constructed as described above. First, the image of each vertex u in $V_{\{x,y\}}$ uses one vertex $\hat{u}$ of $\phi(v_{\{x,y\}})$; together these vertices use at most $|V_{\{x,y\}}| \leq \lceil |V(G')|/\binom{k}{2} \rceil$ vertices from $\phi(v_{\{x,y\}})$. Furthermore, as described in the previous paragraph, for some quadruples (i_1, j_1, i_2, j_2) and integer ℓ, each edge of $E^{\ell}_{i_1,j_1,i_2,j_2}$ requires the use of an additional vertex from $\phi(v_{\{x,y\}})$. More precisely, this can happen only if $\ell = x$ and $y \in \{i_1, j_1, i_2, j_2\}$ or $\ell = y$ and $x \in \{i_1, j_1, i_2, j_2\}$. Thus the total number of vertices used from $\phi(v_{\{x,y\}})$ is at most

$$\begin{aligned}
&\left\lceil \frac{|V(G')|}{\binom{k}{2}} \right\rceil + \sum_{x\in\{i_1,j_1,i_2,j_2\}} |E^{y}_{i_1,j_1,i_2,j_2}| + \sum_{y\in\{i_1,j_1,i_2,j_2\}} |E^{x}_{i_1,j_1,i_2,j_2}| \\
&\qquad \le \frac{|V(G')|}{\binom{k}{2}} + 1 + \sum_{x\in\{i_1,j_1,i_2,j_2\}} \left\lceil \frac{|E_{i_1,j_1,i_2,j_2}|}{k-4} \right\rceil + \sum_{y\in\{i_1,j_1,i_2,j_2\}} \left\lceil \frac{|E_{i_1,j_1,i_2,j_2}|}{k-4} \right\rceil \\
&\qquad \le \frac{|V(G')|}{\binom{k}{2}} + \sum_{x\in\{i_1,j_1,i_2,j_2\}} \frac{|E_{i_1,j_1,i_2,j_2}|}{k-4} + \sum_{y\in\{i_1,j_1,i_2,j_2\}} \frac{|E_{i_1,j_1,i_2,j_2}|}{k-4} + 2k^4 .
\end{aligned}$$

(The term $2k^4$ generously bounds the rounding errors, since it is greater than the number of terms in the sums.) The first sum counts only edges incident to some vertex of $V_{\{i,j\}}$ with $x \in \{i,j\}$ and each edge is counted at most once. Since each vertex has degree at most 3, the number of such edges is at most $3\sum_{x\in\{i,j\}} |V_{\{i,j\}}|$. Thus we can bound the first sum by

$$\frac{3(k-1)\lceil |V(G')|/\binom{k}{2} \rceil}{k-4} \le 12 \left\lceil \frac{|V(G')|}{\binom{k}{2}} \right\rceil$$

(here we use $k \ge 5$). A similar argument applies for the second sum above, hence the number of vertices used from $\phi(v_{\{x,y\}})$ can be bounded as

$$\begin{aligned}
&\frac{|V(G')|}{\binom{k}{2}} + 24 \left\lceil \frac{|V(G')|}{\binom{k}{2}} \right\rceil + 2k^4 \le 25 \frac{|V(G')|}{\binom{k}{2}} + 2k^4 + 24 \\
&\qquad \le 26 \frac{|V(G')|}{\binom{k}{2}} = 52 \frac{|V(G')|}{k(k-1)} \le 65 \frac{|V(G')|}{k^2} = 130 \frac{|E(G)|}{k^2} \le q ,
\end{aligned}$$

what we had to show (in the second inequality, we used that $|V(G')| = 2|E| \ge n_k$ is sufficiently large; in the third inequality, we used that $k \ge 5$ implies $k/(k-1) \le 5/4$). $\square$

Putting together Lemma 3.8 and Lemma 3.9, we can prove the main result of this section.

Proof of Theorem 3.1. Let $k := \mathrm{tw}(G)$, $n := |V(G)|$, and $f_1(G) := n_k + k^2 \lceil c_1 \log n \rceil$, where n_k is the constant from Lemma 3.9 and c_1 is the constant from Lemma 3.8. Assume that $|E(H)| = m \ge f_1(G)$. By Lemma 3.9, H is a minor of $L(K_k)^{(q)}$ for $q := \lceil 130m/k^2 \rceil$ and a minor mapping ψ_1 can be found in polynomial time. Let $q' := \lceil q/\lceil c_1 \log n \rceil \rceil$; clearly, H is a minor of $L(K_k)^{(q' \lceil c_1 \log n \rceil)}$. Observe that m is large enough such that $130m/k^2 \ge 1$ and $q/\lceil c_1 \log n \rceil \ge 1$ holds, hence $q' \le c' \cdot m/(k^2 \cdot \log n)$ for an appropriate constant c'.

By Lemma 3.8, $L(K_k)^{(\lceil c_1 \log n \rceil)}$ is a minor of $G^{(\lceil c_2 \log n \cdot k \log k \rceil)}$ and a minor mapping ψ_2 can be found in time $f_2(G)$ by brute force, for some function $f_2(G)$. Therefore, $L(K_k)^{(q' \lceil c_1 \log n \rceil)}$ is a minor of $G^{(q' \lceil c_2 \log n \cdot k \log k \rceil)}$ and it is straightforward to obtain the corresponding minor mapping ψ_3 from ψ_2. We may assume $c_2 \log n \cdot k \log k \ge 1$, otherwise the theorem automatically holds if we set c sufficiently large. Since $q' \lceil c_2 \log n \cdot k \log k \rceil \le c' \cdot m/(k^2 \cdot \log n) \cdot (2c_2 \log n \cdot k \log k) \le cm \log k/k$ for an appropriate constant

c, we have that H is a minor of $G^{\lceil cm\log k/k \rceil}$. The corresponding minor mapping is the composition $\psi_3 \circ \psi_1$. Observe that each step can be done in polynomial time, except the application of Lemma 3.8, which takes $f_2(G)$ time. Thus the total running time can be bounded by $f_2(G)m^{O(1)}$. □

4 Complexity of binary CSP

In this section, we prove our main result for binary CSP (Theorem 1.3). The proof relies in an essential way on the so-called Sparsification Lemma for 3SAT:

Theorem 4.1 (Impagliazzo, Paturi, and Zane [30])**.** *If there is a $2^{o(m)}$-time algorithm for m-clause 3SAT, then there is a $2^{o(n)}$-time algorithm for n-variable 3SAT.*

The main strategy of the proof of Theorem 1.3 is the following. First we show that a 3SAT formula ϕ with m clauses can be turned into an equivalent binary CSP instance I of size $O(m)$ (Lemma 4.2). Here "equivalent" means that ϕ is satisfiable if and only if I has a solution. By the embedding result of Theorem 3.1, for every $G \in \mathcal{G}$, the primal graph of I is a minor of $G^{(q)}$ for an appropriate q. This implies that we can simulate I with a CSP instance I' whose primal graph is G (Lemma 4.3 and Lemma 4.4). Now we can use the assumed algorithm for CSP($\mathcal{G}$) to solve instance I', and thus decide the satisfiability of formula ϕ. If the treewidth of G is sufficiently large, then the assumed algorithm is much better than the treewidth-based algorithm. This translates into a $2^{o(m)}$ algorithm for the 3SAT instance. By Theorem 4.1, this means that n-variable 3SAT can be solved in time $2^{o(n)}$, i. e., ETH fails.

Lemma 4.2. *Given an instance of 3SAT with n variables and m clauses, it is possible to construct in polynomial time an equivalent CSP instance with $n+m$ variables, $3m$ binary constraints, and domain size 3.*

Proof. Let ϕ be a 3SAT formula with n variables and m clauses. We construct an instance of CSP as follows. The CSP instance contains a variable x_i ($1 \le i \le n$) corresponding to the i-th variable of ϕ and a variable y_j ($1 \le j \le m$) corresponding to the j-th clause of ϕ. Let $D = \{1,2,3\}$ be the domain. We try to describe a satisfying assignment of ϕ with these $n+m$ variables. The intended meaning of the variables is the following. If the value of variable x_i is 1 (or 2), then this represents that the i-th variable of ϕ is true (or false, respectively). If the value of variable y_j is ℓ, then this represents that the j-th clause of ϕ is satisfied by its ℓ-th literal. To ensure consistency, we add $3m$ constraints. Let $1 \le j \le m$ and $1 \le \ell \le 3$, and assume that the ℓ-th literal of the j-th clause is a positive occurrence of the i-th variable. In this case, we add the binary constraint $(x_i = 1 \vee y_j \ne \ell)$: either x_i is true or some other literal satisfies the clause. Similarly, if the ℓ-th literal of the j-th clause is a negated occurrence of the i-th variable, then we add the binary constraint $(x_i = 2 \vee y_j \ne \ell)$. It is easy to verify that if ϕ is satisfiable, then we can assign values to the variables of the CSP instance such that every constraint is satisfied, and conversely, if the CSP instance has a solution, then ϕ is satisfiable. □

If G_1 is a minor of G_2, then an instance with primal graph G_1 can be easily simulated by an instance with primal graph G_2: each variable of G_1 is simulated by a connected set of variables in G_2 that are forced to be equal.

Lemma 4.3. *Assume that G_1 is a minor of G_2. Given a binary CSP instance I_1 with primal graph G_1 and a minor mapping ψ from G_1 to G_2, it is possible to construct in polynomial time an equivalent instance I_2 with primal graph G_2 and the same domain.*

Proof. For simplicity, we assume that both G_1 and G_2 are connected; the proof can be easily extended to the general case. If G_2 is connected, then we may assume that ψ is onto. For each pair (x,y) such that xy is and edge of G_2, we add a constraint as follows. If $\psi^{-1}(x) = \psi^{-1}(y)$, then the new constraint is $\langle(x,y),\{(t,t) \mid t \in D\}\rangle$. If $\psi^{-1}(x) \neq \psi^{-1}(y)$ and there is a constraint $\langle(\psi^{-1}(x),\psi^{-1}(y)),R\rangle$, then the new constraint is $\langle(x,y),R\}\rangle$. Otherwise, the new constraint is $\langle(x,y),D\times D\}\rangle$. Clearly, the primal graph of I_2 is G_2.

Assume that I_1 has a solution $f_1 : V_1 \to D$. Then $f_2(v) = f_1(\psi^{-1}(v))$ is a solution of I_2. On the other hand, if I_2 has a solution $f_2 : V_2 \to D$, then we claim that $f_2(x) = f_2(y)$ holds if $\psi^{-1}(x) = \psi^{-1}(y)$. This follows from the way we defined the constraints of I_2 and from the fact that $\psi(x)$ is connected. Therefore, we can define $f_1 : V_1 \to D$ as $f_1(v) = f_2(v')$, where v' is an arbitrary member of $\psi(v)$. To see that a constraint $c_i = \langle(u,v),R_i\rangle$ of I_1 is satisfied, observe that there is a constraint $\langle(u',v'),R_i\rangle$ in I_2 for some $u' \in \psi(u)$, $v' \in \psi(v)$. This means that $(f_1(u), f_1(v)) = (f_2(u'), f_2(v')) \in R_i$, hence the constraint is satisfied. □

An instance with primal graph $G^{(q)}$ can be simulated by an instance with primal graph G if we set the domain to be the q-tuples of the original domain.

Lemma 4.4. *Given a binary CSP instance $I_1 = (V_1,D_1,C_1)$ with primal graph $G^{(q)}$ (where G has no isolated vertices), it is possible to construct (in time polynomial in the size of the* output*) an equivalent instance $I_2 = (V_2,D_2,C_2)$ with primal graph G and $|D_2| = |D_1|^q$.*

Proof. Let ψ be the blow-up mapping from G to $G^{(q)}$ and let $D_2 = D_1^q$, i. e., D_2 is the set of q-tuples of D_1. For every $v \in V_2$, there is a natural bijection between the elements of D_2 and the $|D_1|^q$ possible assignments $f : \psi(v) \to D_1$. For each edge v_1v_2 of G, we add a constraint $c_{v_1,v_2} = \langle(v_1,v_2),R_{v_1,v_2}\rangle$ to I_2 as follows. Let $(x_1,x_2) \in D_2 \times D_2$. For $i = 1,2$, let g_i be the assignment of $\psi(v_i)$ corresponding to $x_i \in D_2$. The two assignment together define an assignment $g : \psi(v_1) \cup \psi(v_2) \to D$ on the union of their domains. We define the relation R_{v_1,v_2} such that (x_1,x_2) is a member of R_{v_1,v_2} if and only if the corresponding assignment g is a solution of the induced instance $I[\psi(v_1) \cup \psi(v_2)]$.

Assume that I_1 has a solution $f_1 : V_1 \to D_1$. For every $v \in V_2$, let us define $f_2(v)$ to be the member of D_2 corresponding to the assignment f_1 restricted to $\psi(v)$. It is easy to see that f_2 is a solution of I_2: this follows from the trivial fact that for every edge v_1v_2 in G, assignment f_1 restricted to $\psi(v_1) \cup \psi(v_2)$ is a solution of $I_1[\psi(v_1) \cup \psi(v_2)]$.

Assume now that I_2 has a solution $f_2 : V_2 \to D_2$. For every $v \in V_2$, there is an assignment $f_v : \psi(v) \to D_1$ corresponding to $f_2(v)$. These assignments together define an assignment $f_1 : V_1 \to D_1$. We claim that f_1 is a solution of I_1. Let $c_{u,v} = \langle(u,v),R\rangle$ be an arbitrary constraint of I_1. Assume that $u \in \psi(u')$ and $v \in \psi(v')$. If $u' \neq v'$, then $u'v'$ is an edge of G, hence there is a corresponding constraint $c_{u',v'}$ in I_2. The way $c_{u',v'}$ is defined ensures that f_1 restricted to $\psi(u') \cup \psi(v')$ is a solution of $I_1[\psi(u') \cup \psi(v')]$. In particular, this means that $c_{u,v}$ is satisfied in f_1. If $u' = v'$, then there is an edge $u'w$ in G (since G has no isolated vertices), and the corresponding constraint $c_{u',w}$ ensures that f_1 satisfies $c_{u,v}$. □

Now we are ready to prove the main result:

Proof of Theorem 1.3. Assume that there is an algorithm $\mathbb{A}$ that correctly decides every CSP instance and whose running time can be bounded by $f(G)\|I\|^{\mathrm{tw}(G)/(\log \mathrm{tw}(G)\cdot\iota(\mathrm{tw}(G)))}$ for instances with $G \in \mathcal{G}$, where ι is an unbounded function. We may assume that ι is nondecreasing and $\iota(1) \geq 1$. We present a reduction from 3SAT to CSP($\mathcal{G}$) such that this reduction, together with the assumed algorithm $\mathbb{A}$ for CSP($\mathcal{G}$), gives an algorithm $\mathbb{B}$ that is able to solve m-clause 3SAT in time $2^{o(m)}$. Lemma 4.2, Theorem 3.1, and Lemmas 4.3 and 4.4 show a way of solving a 3SAT instance by reducing it to a CSP instance having a particular primal graph G. A crucial point of the reduction is how to select an appropriate G from $\mathcal{G}$. The higher the treewidth of G, the more we gain in the running time. However, G has to be sufficiently small such that some additional factors (such as the time spent on finding G) are not too large.

Given an m-clause 3SAT formula ϕ and a graph $G \in \mathcal{G}$, algorithm $\mathbb{A}$ can be used to decide the satisfiability of ϕ in the following way. By Lemma 4.2, ϕ can be turned into a binary CSP instance I_1 with $O(m)$ constraints and domain size 3. Let H be the primal graph of I_1. For simplicity, we assume that G has no isolated vertices as they can be handled in a straightforward way. By Theorem 3.1, H is a minor of $G^{(q)}$ for $q = O(m\log k/k)$ and we can find a minor mapping ψ in time $f_2(G)m^{O(1)}$. Therefore, by Lemma 4.3, I_1 can be turned into an instance I_2 with primal graph $G^{(q)}$, which, by Lemma 4.4, can be turned into an instance I_3 with primal graph G and domain size 3^q. Now we can use algorithm $\mathbb{A}$ to solve instance I_3.

We shall refer to this way of solving the 3SAT instance ϕ as "running algorithm $\mathbb{A}[\phi,G]$." Let us determine the running time of $\mathbb{A}[\phi,G]$. The two dominating terms are the time required to find the minor mapping from H to $G^{(q)}$ and the time required to run $\mathbb{A}$ on I_3. Note that $\|I_3\| = O(|E(G)|3^{2q})$: there are $|E(G)|$ constraints and each binary constraint contains at most $3^q \cdot 3^q$ pairs. Let k be the treewidth of G. The total running time of $\mathbb{A}[\phi,G]$ can be bounded by

$$\begin{aligned} f_2(G)m^{O(1)} + f(G)\|I_3\|^{k/(\log k\cdot\iota(k))} &= f_2(G)m^{O(1)} + f(G)|E(G)|^{k/(\log k\cdot\iota(k))}\cdot 3^{2qk/(\log k\cdot\iota(k))} \\ &= \hat{f}(G)m^{O(1)}\cdot 2^{O(qk/(\log k\cdot\iota(k)))} = \hat{f}(G)m^{O(1)}\cdot 2^{O(m/\iota(k))} \end{aligned}$$

for an appropriate function $\hat{f}(G)$.

Let us fix an arbitrary easy-to-compute enumeration G_1, G_2, ... of all graphs. Given an m-clause 3SAT formula ϕ, we first spend m steps to enumerate graphs from $\mathcal{G}$; let G_ℓ (for some $\ell \leq m$) be the last graph enumerated (we assume that m is sufficiently large such that $\ell \geq 1$). Next we start simulating the algorithms $\mathbb{A}[\phi,G_1]$, $\mathbb{A}[\phi,G_2]$, ..., $\mathbb{A}[\phi,G_\ell]$ in *parallel*. When one of the simulations stops and returns an answer, then we stop all the simulations and return the answer. It is clear that this algorithm will correctly decide the satisfiability of ϕ.

We claim that there is a universal constant C such that for every s, there is an m_s such that for every $m > m_s$, the running time of $\mathbb{B}$ is $(m\cdot 2^{m/s})^C$ on an m-clause formula. Clearly, this means that the running time of $\mathbb{B}$ is $2^{o(m)}$.

Let k_s be the smallest positive integer such that $\iota(k_s) \geq s$ (as ι is unbounded, this is well defined). Let i_s be the smallest positive integer such that $G_{i_s} \in \mathcal{G}$ and $\mathrm{tw}(G_{i_s}) \geq k_s$ (as $\mathcal{G}$ has unbounded treewidth, this is also well defined). Set m_s sufficiently large such that $m_s \geq \hat{f}(G_{i_s})$ and the enumeration of all graphs reaches G_{i_s} in less then m_s steps. This means that if we run $\mathbb{B}$ on a 3SAT formula ϕ with $m \geq m_s$ clauses, then $\mathbb{A}[\phi,G_{i_s}]$ will be one of the ℓ simulations started by $\mathbb{B}$. The simulation of $\mathbb{A}[\phi,G_{i_s}]$ terminates in

$$\hat{f}(G_{i_s})m^{O(1)}\cdot 2^{O(m/\iota(\mathrm{tw}(G_{i_s})))} = m\cdot m^{O(1)}\cdot 2^{O(m/s)}$$

steps. Taking into account that we simulate $\ell \le m$ algorithms in parallel and all the simulations are stopped not later than the termination of $\mathbb{A}[\phi, G_{i_s}]$, the running time of $\mathbb{B}$ can be bounded polynomially by the running time of $\mathbb{A}[\phi, G_{i_s}]$. Therefore, there is a constant C such that the running time of $\mathbb{B}$ is $(m \cdot 2^{m/s})^C$, as required. □

We close this section by proving a variant of Theorem 1.3, where the output of the algorithm is unspecified for instances whose primal graph is not in $\mathcal{G}$.

Corollary 4.5. *If there is a recursively enumerable class $\mathcal{G}$ of graphs with unbounded treewidth, an algorithm $\mathbb{A}$, and a function f such that $\mathbb{A}$ decides every binary CSP($\mathcal{G}$) instance I with primal graph $G \in \mathcal{G}$ in time $f(G)\|I\|^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))}$, then ETH fails.*

Proof. To show that Theorem 1.3 implies Corollary 4.5, it is sufficient to show that an algorithm $\mathbb{A}$ satisfying the requirements of Corollary 4.5 implies that there exists an algorithm $\mathbb{A}'$ satisfying the requirements of Theorem 1.3. We construct $\mathbb{A}'$ as follows. Given a CSP instance I with primal graph G (not necessarily in $\mathcal{G}$), we start running in parallel a brute force algorithm to decide I and an enumeration of the members of the recursively enumerable class $\mathcal{G}$. If the brute force algorithm stops before reaching G in the enumeration, then we return its answer (which is correct). If the enumeration of $\mathcal{G}$ reaches G before the brute force algorithms stops, we start simulating algorithm $\mathbb{A}$ on I, and return its answer (which is correct, as we know that $G \in \mathcal{G}$ in this case). The overhead for instances with $G \in \mathcal{G}$ is just a constant depending on G, so we get an algorithm that is correct for every G and for instances with $G \in \mathcal{G}$, the running time can be bounded by $f'(G)\|I\|^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))}$ for some function f'. □

5 Complexity of homomorphism

The aim of this section is to extend Theorem 1.3 to the framework of the homomorphism problem for relational structures, which is the standard setting in which CSP has been studied in the theory literature. As we shall see, in this formulation the complexity of the problem depends on the treewidth of the core of the left-hand side. Furthermore, as in the result of Grohe [26], we limit ourselves to relational structures of bounded arity.

Let us recall the standard definitions of the homomorphism problem (see [15, 26]). A *vocabulary* τ is a finite set of relation symbols of specified arities. The *arity* of τ is the maximum of the arities of all relational symbols it contains. A τ-structure $\mathbf{A}$ consists of a finite set A called the *universe* of $\mathbf{A}$ and for each relation symbol $R \in \tau$, say, of arity k, a k-ary relation $R^{\mathbf{A}} \subseteq A^k$. We say that a class $\mathcal{C}$ of structures is of *bounded arity* if there is a constant r such that the arity of the vocabulary of every structure in $\mathcal{C}$ is at most r. A *homomorphism* from a τ-structure $\mathbf{A}$ to a τ-structure $\mathbf{B}$ is a mapping $h : A \to B$ from the universe of $\mathbf{A}$ to the universe of $\mathbf{B}$ that preserves all relations, that is, for all $R \in \tau$, say, of arity k, and all tuples $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$ it holds that $(h(a_1), \ldots, h(a_k)) \in R^{\mathbf{B}}$. Let $\|\mathbf{A}\|$ denote the length of the representation of $\mathbf{A}$. We assume that $\|\mathbf{A}\| = O(|\tau| + |A| + \sum_{R \in \tau} |R^A| \cdot \mathrm{arity}(R))$ for a τ-structure $\mathbf{A}$ with universe A.

A *substructure* of a relational structure $\mathbf{A}$ is a relational structure $\mathbf{B}$ over the same vocabulary τ as $\mathbf{A}$ where $B \subseteq A$ and $R^{\mathbf{B}} \subseteq R^{\mathbf{A}}$ for all $R \in \tau$. If $\mathbf{B}$ is a substructure of $\mathbf{A}$, but $\mathbf{A} \neq \mathbf{B}$, then $\mathbf{B}$ is a *proper substructure* of $\mathbf{A}$.

The notion of treewidth can be defined for relational structures the following way. A *tree decomposition* of a τ-structure $\mathbf{A}$ is a pair (T,X), where $T=(I,F)$ is a tree, and $X=(X_i)_{i\in I}$ is a family of subsets of A such that for each $R\in\tau$, say, of arity k, and each $(a_1,\ldots,a_k)\in R^{\mathbf{A}}$ there is a node $i\in I$ such that $\{a_1,\ldots,a_k\}\subseteq X_i$, and for each $a\in A$ the set $\{i\in I \mid a\in X_i\}$ is connected in T. The *width* of the decomposition (T,X) is $\max\{|X_i| \mid i\in I\}-1$, and the *treewidth* of $\mathbf{A}$, denoted by $\mathrm{tw}(\mathbf{A})$, is the minimum of the widths of all tree decompositions of $\mathbf{A}$.

The *primal graph* of a structure $\mathbf{A}$ with vocabulary τ is a graph with vertex set A where two elements $a',a''\in A$ are connected if and only if there is a relational symbol $R\in\tau$, say, of arity k, such that R has a tuple $(a_1,\ldots,a_k)\in R$ with $a',a''\in\{a_1,\ldots,a_k\}$. It can be shown that the treewidth of the primal graph of $\mathbf{A}$ equals the treewidth of $\mathbf{A}$ (cf. [19, Proposition 11.27]).

A *core* of a relational structure $\mathbf{A}$ is a substructure $\mathbf{A}'$ of $\mathbf{A}$ such that there is a homomorphism from $\mathbf{A}$ to $\mathbf{A}'$, but there is no homomorphism from $\mathbf{A}$ to a proper substructure of $\mathbf{A}'$. We say that a relational structure $\mathbf{A}$ is a *core* if it is its own core. It is well-known that the every relational structure $\mathbf{A}$ has a core and the cores of $\mathbf{A}$ are isomorphic with each other. Let us denote by $\mathrm{ctw}(\mathbf{A})$ the treewidth of the core of $\mathbf{A}$.

Given a CSP instance $I=(V,D,C)$, one can construct in polynomial time two relational structures $\mathbf{A}$ and $\mathbf{B}$ with universe V and D, respectively, such that the solutions of I correspond to the homomorphisms from $\mathbf{A}$ to $\mathbf{B}$. Thus the homomorphism problem of relational structures generalizes constraint satisfaction. Formally, in the homomorphism problem the input is a pair $(\mathbf{A},\mathbf{B})$ of relational structures and the task is to decide whether there is a homomorphism from $\mathbf{A}$ (the *left-hand side structure*) to $\mathbf{B}$ (the *right-hand side structure*). If $\mathcal{A}$ and $\mathcal{B}$ are two classes of relational structures, then we denote by $\mathrm{HOM}(\mathcal{A},\mathcal{B})$ the restriction of the homomorphism problem where $\mathbf{A}\in\mathcal{A}$ and $\mathbf{B}\in\mathcal{B}$. We denote by the symbol $-$ the class of all relational structures. Thus $\mathrm{HOM}(\mathcal{A},-)$ restricts the structure of the constraints, while $\mathrm{HOM}(-,\mathcal{B})$ restricts the constraint language.

If $\mathrm{ctw}(\mathbf{A})\le k$, then the decision version of the homomorphism problem $(\mathbf{A},\mathbf{B})$ can be solved by the k-consistency algorithm in time $n^{O(k)}$ [26, 11] (where n is the length of the input, which is $O(\|A\|+\|B\|)$). The main result of this section is that there is no class $\mathcal{A}$ of structures such that $\mathrm{HOM}(\mathcal{A},-)$ can be solved significantly faster:

Theorem 5.1. *Let $\mathcal{A}$ be a class of bounded-arity relational structures such that the treewidth of the core is unbounded. If there is a function f and an algorithm $\mathbb{A}$ that correctly decides every instance of HOM$(-,-)$ and for instances of HOM$(\mathcal{A},-)$ the running time is $f(\mathbf{A})\|\mathbf{B}\|^{o(\mathrm{ctw}(\mathbf{A})/\log\mathrm{ctw}(\mathbf{A}))}$, then ETH fails.*

For the proof, we need the following lemma:

Lemma 5.2. *The following is true for every fixed $r_{max}\ge 1$. Given a CSP instance I, a relational structure $\mathbf{A}$ of arity at most r_{max}, and an isomorphism between the primal graph of I and the primal graph of the core of $\mathbf{A}$, it is possible to construct in polynomial time a relational structure $\mathbf{B}$ such that I has a solution if and only if there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*

Proof. Let $I=(V,D,C)$ be an instance of binary CSP with primal graph G. Let $\mathbf{A}$ be a structure whose core $\mathbf{A}_0$ has a primal graph isomorphic to G. For ease of notation, from now on we use V both for the set of variables of instance I and for the universe of $\mathbf{A}_0$. Let τ be the vocabulary of $\mathbf{A}$. We construct a

τ-structure $\mathbf{B}$ as follows. The universe B of $\mathbf{B}$ is $V \times D$. Let $R \in \tau$ be a relation symbol of arity r and let $R^{\mathbf{A}_0}$ be the corresponding relation in $\mathbf{A}_0$. To construct the relation $R^{\mathbf{B}}$, let us enumerate the r-tuples of $R^{\mathbf{A}_0}$, and for each $(v_1,\dots,v_r) \in R^{\mathbf{A}_0} \subseteq V^r$, let us enumerate the solutions of the induced instance $I[\{v_1,\dots,v_r\}]$. If $(v_1,\dots,v_r) \in R^{\mathbf{A}_0}$ and f is a solution of $I[(v_1,\dots,v_r\}]$, then let us add the r-tuple $((v_1, f(v_1)),\dots,(v_r, f(v_r))$ to $R^{\mathbf{B}}$. This completes the description of the relation $R^{\mathbf{B}}$ and the structure $\mathbf{B}$. Observe that the size of $R^{\mathbf{B}}$ is at most $D^{r_{\max}}$ times the size of $R^{\mathbf{A}_0}$. Therefore, the size of $\mathbf{B}$ is $(\|\mathbf{A}_0\||D|)^{O(r_{\max})}$ and can be constructed in time polynomial in its size (for fixed $r_{\max}$).

We show that $\mathbf{A}_0 \to \mathbf{B}$ if and only if I has a solution. Since $\mathbf{A}_0$ is the core of $\mathbf{A}$, it follows that $\mathbf{A} \to \mathbf{B}$ if and only if $\mathbf{A}_0 \to \mathbf{B}$.

Assume first that I has a solution $f : V \to D$. We claim that $\phi(v) = (v, f(v))$ is a homomorphism from $\mathbf{A}_0$ to $\mathbf{B}$. Indeed, if $(v_1,\dots,v_r) \in R^{\mathbf{A}_0}$, then f restricted to $\{v_1,\dots,v_r\}$ is obviously a solution of $I[\{v_1,\dots,v_r\}]$, hence $((v_1, f(v_1)),\dots,(v_r, f(v_r))) \in R^{\mathbf{B}}$ by the definition of $R^{\mathbf{B}}$.

Assume now that ϕ is a homomorphism from $\mathbf{A}_0$ to $\mathbf{B}$. Let ψ be the projection $\psi((v,d)) = v$ from $V \times D$ to V. Observe that ψ is a homomorphism from $\mathbf{B}$ to $\mathbf{A}_0$. Therefore, $\psi \circ \phi$ is a homomorphism from $\mathbf{A}_0$ to itself. Since $\mathbf{A}_0$ is core, $\psi \circ \phi$ is an isomorphism of $\mathbf{A}_0$. Thus we may assume that $\psi \circ \phi$ is identity: otherwise let us replace ϕ by $\phi \circ (\psi \circ \phi)^{-1}$. If $\psi \circ \phi$ is the identity, then for every $v \in V$, $\phi(v) = (v, f(v))$ for some $f(v) \in D$. We claim that this function $f : V \to D$ is a solution of I. Let $c_i = \langle (u,v), R_i \rangle$ be an arbitrary constraint of I. Since uv is an edge of the primal graph G, there is an $R \in \tau$ such that $R^{\mathbf{A}_0}$ has a tuple $(v_1,\dots,v_r)$ containing both u and v. Therefore, $(\phi(v_1),\dots,\phi(v_r)) = ((v_1, f(v_1)),\dots,(v_r, f(v_r))) \in R^{\mathbf{B}}$. By the definition of $R^{\mathbf{B}}$, this means that f restricted to $\{v_1,\dots,v_r\}$ is a solution of $I[\{v_1,\dots,v_r\}]$. In particular, this means that f satisfies c_i. $\square$

Proof of Theorem 5.1. Let $\mathcal{A}$ be a class of relational structures of maximum arity $r_{\max}$. Let us fix an enumeration of *all* relational structures. Let $\mathcal{G}$ be the class of graphs containing the primal graph of the core of every structure $\mathbf{A} \in \mathcal{A}$. Clearly, $\mathcal{G}$ has unbounded treewidth. We use algorithm $\mathbb{A}$ to construct an algorithm for CSP($\mathcal{G}$) that contradicts Theorem 1.3.

Given an instance $I = (V, D, C)$ of binary CSP with primal graph G, we proceed as follows. Let $n = \|I\|$. We start enumerating relational structures and for each structure we determine the core and try to find an isomorphism between the primal graph of the core and G. We spend n steps on this task. If after n steps no relational structure is found for which the primal graph of the core is isomorphic to G, then we solve instance I by brute force. Otherwise, let $\mathbf{A}_1, \dots, \mathbf{A}_t$ be the structures satisfying this requirement and let $\phi_1, \dots, \phi_t$ be the corresponding isomorphisms. We use Lemma 5.2 to compute structures $\mathbf{B}_1, \dots, \mathbf{B}_t$; observe that the size of each $\mathbf{B}_i$ is polynomial in n. Then we start simulating algorithm $\mathbb{A}$ in parallel on inputs $(\mathbf{A}_1, \mathbf{B}_1), \dots, (\mathbf{A}_t, \mathbf{B}_t)$. Whenever one of the simulation stops and returns an answer, we stop all the simulations and return this answer.

It is clear that this way of deciding I always return a correct answer. We bound the running time as follows. For every graph $G \in \mathcal{G}$, let us define $\mathbf{A}_G \in \mathcal{A}$ to be the first structure in the enumeration that is in $\mathcal{A}$ and for which the primal graph of the core is isomorphic to G, and let $n(G)$ be the smallest n such that the above algorithm reaches $\mathbf{A}_G$ in the enumeration and constructs the corresponding isomorphism. Note that $\mathbf{A}_G$ and $n(G)$ are well defined for every $G \in \mathcal{G}$: by the definition of $\mathcal{G}$, there has to be such a structure $\mathbf{A}_G$. For instances I with length $n < n(G)$, the running time can be bounded by a function of G. For instances I with length $n \ge n(G)$, some number $t \le n$ of simulations of $\mathbb{A}$ on $(\mathbf{A}_1, \mathbf{B}_1), \dots, (\mathbf{A}_t, \mathbf{B}_t)$ are started. For some $1 \le i \le t$, we have $\mathbf{A}_i = \mathbf{A}_G \in \mathcal{A}$. By the assumption on $\mathbb{A}$, the simulation of $(\mathbf{A}_i, \mathbf{B}_i)$ terminates in time

$$f(\mathbf{A}_i)\|\mathbf{B}_i\|^{o(\mathrm{ctw}(\mathbf{A}_i)/\log \mathrm{ctw}(\mathbf{A}_i))} = \hat{f}(G)n^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))},$$

for some function $\hat{f}(G)$. At most n simulations are started and at most n steps are spent on enumeration before the simulations. Thus the total running time is $\hat{f}(G)n^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))}$, contradicting Theorem 1.3. □

As in Section 4 for CSP, we can obtain a corollary dealing with algorithms whose output is unspecified for instances where the left-hand side structure is not in $\mathcal{A}$. The proof is similar.

Corollary 5.3. *Let $\mathcal{A}$ be a recursively enumerable class of bounded-arity relational structures such that the treewidth of the core is unbounded. If there is a function f and an algorithm $\mathbb{A}$ that decides every instance of HOM$(\mathcal{A}, -)$ in time $f(\mathbf{A})\|\mathbf{B}\|^{o(\mathrm{ctw}(\mathbf{A})/\log \mathrm{ctw}(\mathbf{A}))}$, then ETH fails.*

6 Complexity of subgraph problems

Subgraph Isomorphism is a basic graph-theoretic problem: given graphs G and H, we have to decide if G is a subgraph of H. That is, we have to find an injective mapping $\phi : V(G) \to V(H)$ such that if u and v are adjacent in the smaller graph G, then $\phi(u)$ and $\phi(v)$ are adjacent in the larger graph H. In the Colored Subgraph Isomorphism problem, the input contains a (not necessarily proper) coloring of the vertices of H and G. The task is to find a subgraph mapping ϕ that satisfies the additional constraint that for every $v \in V(G)$, the color of $\phi(v)$ has to be the same as the color v. Partitioned Subgraph Isomorphism is a special case of the colored version where every vertex of the smaller graph G has a distinct color (i. e., we can assume that $V(G)$ is the set of colors). In other words, the vertices of H are partitioned into $|V(G)|$ classes, and the image of each $v \in V(G)$ is restricted to a distinct class of the partition. Clearly, Partitioned Subgraph Isomorphism is a special case of Colored Subgraph Isomorphism, thus any lower bound for the former problem is a lower bound for the latter as well.

It is not hard to observe that Partitioned Subgraph Isomorphism is essentially the same as binary CSP. We can reduce an instance $I = (V, D, C)$ of binary CSP to Partitioned Subgraph Isomorphism the following way. Let G be the primal graph of I. We construct a graph H, whose vertex set is $V(G) \times D$, and the color of $(v, d) \in V(G) \times D$ is v. For every constraint $\langle (u, v), R_{uv} \rangle \in C$ and every pair $(d_u, d_v) \in R_{uv}$, we add an edge connecting (u, d_u) and (v, d_v) to H. Note that this construction is very similar to the proof of Lemma 5.2.

Suppose that $f : V \to D$ is a satisfying assignment of I and consider the mapping $\phi(v) = (v, f(v))$ for every $v \in V(G)$. It is clear that ϕ respects the colors and it is a subgraph mapping: if u and v are adjacent in G, then there is a corresponding constraint $\langle (u, v), R_{uv} \rangle \in C$, and the fact that $(f(u), f(v)) \in R_{uv}$ implies that $\phi(u)$ and $\phi(v)$ are adjacent. On the other hand, suppose that ϕ is a subgraph mapping respecting the colors. This means the first coordinate of $\phi(v)$ is v; let $f(v)$ be the second coordinate of $\phi(v)$. It is straightforward to verify that f is a satisfying assignment: for every constraint $\langle (u, v), R_{uv} \rangle \in C$, vertices u and v are adjacent in G by the definition of the primal graph, and hence the fact that $(u, f(u))$ and $(v, f(v))$ are adjacent implies that $(f(u), f(v)) \in R_{uv}$.

The reduction from binary CSP to Partitioned Subgraph Isomorphism implies that any lower bound for the former problem can be transfered to the latter. Thus Theorem 1.3 implies the following result:

Corollary 6.1. *If there is a class $\mathcal{G}$ of graphs with unbounded treewidth, an algorithm $\mathbb{A}$, and an arbitrary function f such that $\mathbb{A}$ correctly decides every instance of Partitioned Subgraph Isomorphism and the running time is $f(G)n^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))}$ for instances with the smaller graph G in $\mathcal{G}$, then ETH fails.*

Similarly, Corollary 4.5 can be translated the following way:

Corollary 6.2. *If there is a recursively enumerable class $\mathcal{G}$ of graphs with unbounded treewidth, an algorithm $\mathbb{A}$, and an arbitrary function f such that $\mathbb{A}$ correctly decides every instance of Partitioned Subgraph Isomorphism with the smaller graph G in $\mathcal{G}$ in time $f(G)n^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))}$, then ETH fails.*

It is known that there are infinite recursively enumerable classes $\mathcal{G}$ of graphs such that for every $G \in \mathcal{G}$, both the treewidth and the number of edges are $\Theta(|V(G)|)$: for example, explicit constructions of bounded-degree expanders give such classes (cf. [28]). Using this class $\mathcal{G}$ in Corollary 6.1, we get

Corollary 6.3. *If Partitioned Subgraph Isomorphism can be solved in time $f(G)n^{o(k/\log k)}$, where f is an arbitrary function and k is the number of* edges *of the smaller graph G, then ETH fails.*

Can we prove similar lower bounds for the more natural Subgraph Isomorphism problem (without colors)? Unfortunately, the situation for Subgraph Isomorphism is much less understood. For example, it is a major open question of parameterized complexity whether the k-Biclique problem (given a graph H and an integer k, decide if H contains a $K_{k,k}$ complete bipartite subgraph) is fixed-parameter tractable, i. e., can be solved in time $f(k) \cdot n^{O(1)}$ for some function f depending only on k. Without answering this question, we cannot prove the analog of Corollary 6.1 for Subgraph Isomorphism.

However, there is a special where we can prove lower bounds. Recall that a graph G is a *core* if it has no homomorphism to any of its proper induced subgraphs, that is, if a mapping $\phi : V(G) \to V(G)$ satisfies that $\phi(u)$ and $\phi(v)$ are adjacent for every adjacent $u, v \in V(G)$, then ϕ is bijective. We show that if G is a core, then the colored and uncolored versions are equivalent (essentially, we use the same argument as in the proof of Theorem 5.1). Consider an instance of Partitioned Subgraph Isomorphism with smaller graph G and larger graph H. We may assume that if $u, v \in V(G)$ are not adjacent, then H has no edge whose endpoints are colored u and v, as such an edge could not be used in a solution. We claim that if G is a core and there is a subgraph mapping ϕ from G to H, then there is a subgraph mapping that respects the colors. Let $\psi(v)$ be the color of $\phi(v)$. If $u, v \in V(G)$ are adjacent, then $\phi(u)\phi(v)$ is an edge whose endpoints have colors $\psi(u)$ and $\psi(v)$, which means by our assumption that $\psi(u)$ and $\psi(v)$ are adjacent in H. As H is a core, ψ is an isomorphism of H. Now $\phi(\psi^{-1}(v))$ is a subgraph mapping that respects the colors. Therefore, the lower bounds for Partitioned Subgraph Isomorphism can be transfered to the uncolored problem:

Corollary 6.4. *Let $\mathcal{G}$ be a class of graphs with unbounded treewidth such that every graph in $\mathcal{G}$ is a core. If there is an algorithm $\mathbb{A}$ and an arbitrary function f such that $\mathbb{A}$ correctly decides every instance of Subgraph Isomorphism and the running time is $f(G)n^{o(\mathrm{tw}(G)/\log \mathrm{tw}(G))}$ for instances with the smaller graph G in $\mathcal{G}$, then ETH fails.*

The analog of Corollary 6.2 for Subgraph Isomorphism can be obtained in a similar way. To prove the analog of Corollary 6.3 for Subgraph Isomorphism, we need a family of graphs that are cores, sparse, and treewidth is linear in the number of vertices. The following lemma provides such a family:

Lemma 6.5. *There is a recursively enumerable family of graph $\mathcal{G}$ such that every $G \in \mathcal{G}$ is a core, and both the treewidth and the number of edges of G are $\Theta(|V(G)|)$.*

Proof. Let $\mathcal{G}_0$ be a family of bounded-degree expanders, such as the one given by Gabber and Galil [22]. We will use the known result that the treewidth of such graphs is linear in the number of vertices (cf. [28]). We may assume that the graphs in $\mathcal{G}_0$ are bipartite: subdividing every edge does not decrease treewidth and increases the number of vertices only by a constant factor (as the graph has bounded degree).

We will need the following auxiliary graphs. The graph T_n has $n+2(n-1)$ vertices v_i ($1 \le i \le n$) and $u_{i,1}$, $u_{i,2}$ ($2 \le i \le n$), edges $u_{i,1}u_{i,2}$, $v_iu_{i,1}$, $v_iu_{i,2}$, $v_{i+1}u_{i,1}$, $v_{i+1}u_{i,2}$ for every $1 \le i \le n-1$, and the edges v_1v_n, $v_nu_{n-1,1}$. Graph T_n is not 3-colorable: in any 3-coloring, vertices v_i and v_{i+1} would get the same color, which is impossible, as v_1 and v_n are adjacent. Furthermore, it is easy to see that deleting any vertex makes T_n 3-colorable. This immediately implies that T_n is a core: a homomorphism from T_n to a proper induced subgraph of T_n would map a non-3-colorable graph to a 3-colorable graph, which is impossible. Thus every homomorphism of T_n is an isomorphism. Moreover, it can be verified that any such isomorphism maps v_i to v_i for every $1 \le i \le n$. Note that v_n having degree 4 cannot be mapped to v_1 having degree 3.

For every (bipartite) graph $G_0 \in \mathcal{G}_0$, we construct a graph G as follows. Let $w_1, \dots, w_n$ be the vertices of G_0. We attach a copy of the graph T_{2n+1} to G_0 by making w_i and v_{2i} adjacent for every $1 \le i \le n$. It is clear that the graph G obtained this way is sparse and has treewidth linear in the number of vertices. Thus the only thing we have to verify is that G is a core. Note that every vertex of T_{2n+1} appears in a triangle, and no other vertex of G appears in a triangle (since we assumed that G is bipartite). Therefore, any homomorphism ϕ has to map the vertices of T_{2n+1} to vertices of T_{2n+1}. Therefore, ϕ induces a homomorphism of T_{2n+1}, which means that $\phi(v_i) = v_i$ for every $1 \le i \le 2n$. We claim that $\phi(w_i) = w_i$ for every $1 \le i \le n$. Let w_j be an arbitrary neighbor of w_i. There is a path $v_{2i}w_iw_jv_{2j}$ of length 3 between v_{2i} and v_{2j} in G. Applying ϕ on this path gives a walk of length 3 between v_{2i} and v_{2j}. As the distance of v_{2i} and v_{2j} is greater than 3 in T_{2n+1}, this is only possible if the walk leaves T_{2n+1}, implying $\phi(w_i) = w_i$ and $\psi(w_j) = w_j$. □

Putting together Corollary 6.4 and Lemma 6.5 immediately gives Corollary 1.6 stated in the introduction.

7 Conclusions

We have proved that for binary CSP and for the homomorphism problem for relational structures of bounded arity, the algorithms based on treewidth are almost optimal, in the sense that at most a logarithmic factor improvement is possible in the exponent of the running time. This improves the main result of Grohe [26] by making it quantitative: [26] explored only whether there exists a polynomial-time algorithm for a given a class of problems and no effort was made to determine the best possible super-polynomial running time. The main technical tool in our paper is converting a 3SAT formula to a CSP instance by embedding a graph into the blowup of another graph. To obtain this embedding, we use characterizations of treewidth by separators and a dual characterization of separators. We avoid the use of the Excluded Grid Theorem (the main combinatorial tool in [26]), as it is not suitable for obtaining tight results.

The results of this paper suggest two obvious directions for future work. First, one could try to make Theorem 1.3 tight by removing the logarithmic factor from the exponent. We conjecture that this is actually possible (Conjecture 1.4). An obvious approach to prove Conjecture 1.4 would be to prove Theorem 3.1 without the logarithmic factor in the exponent; an inspection of our proof shows that if Theorem 3.1 is true without the logarithmic factor, then Theorem 1.3 is true without the logarithmic factor. More specifically, if we can get rid of $\log k$ in Theorem 3.1 for every $G \in \mathcal{G}$ for some class $\mathcal{G}$ of graphs, then we can get rid of the logarithmic factor in Theorem 1.3 for the problem CSP($\mathcal{G}$). For example, Theorem 3.1 is certainly true without $\log k$ if G is a clique, which implies that Theorem 1.3 is true without the logarithmic factor if $\mathcal{G}$ is the class of all cliques. However, as shown very recently in [1], Theorem 3.1 is tight: there are classes of graphs for which the logarithmic factor is needed. This does not invalidate Conjecture 1.4, but it shows that its proof would require substantially different techniques than the embedding method of this paper. Moreover, probably one should first settle the question of whether there is a polynomial-time constant-factor approximation algorithm for treewidth.

The second direction would be to generalize the results to constraints with higher arities. Theorem 1.3 is stated for binary CSP($\mathcal{G}$), but this means that the negative result also holds for the more general problem where we do not assume that the instance is binary. However, for higher arity CSPs, we can define the hypergraph of the instance the obvious way, and try to understand the complexity in terms of this hypergraph instead of the primal graph. If the arities of the constraints are bounded by a constant, then Theorem 5.1 characterizes the tractable hypergraph classes, as a hypergraph can be expressed by a relational structure (where there is a distinct relation symbol for each hyperedge, to allow every constraint relation to be different). The problem changes considerably if the arities of the constraints are unbounded [27, 24, 23, 8, 35, 37] due to issues related to the representation of constraints. The notions of hypertree width and fractional hypertree width were introduced to obtain tractable classes not covered by bounded treewidth. However, the situation is still far from understood.

Acknowledgments

I'm grateful to Martin Grohe for the many useful discussions that were crucial for obtaining the results. The comments of the anonymous referees and the editor Laci Babai were very helpful in improving the quality of the paper.

References

[1] NOGA ALON AND DÁNIEL MARX: Sparse balanced partitions and the complexity of subgraph problems. Manuscript, 2010. 108

[2] NOGA ALON, RAPHAEL YUSTER, AND URI ZWICK: Color-coding. *J. ACM*, 42(4):844–856, 1995. [doi:10.1145/210332.210337]. 89

[3] HANS L. BODLAENDER: A tourist guide through treewidth. *Acta Cybernet.*, 11(1-2):1–21, 1993. 91

[4] HANS L. BODLAENDER: A partial *k*-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, 209(1-2):1–45, 1998. [doi:10.1016/S0304-3975(97)00228-4]. 91

[5] ANDREI BULATOV, ANDREI KROKHIN, AND PETER JEAVONS: The complexity of maximal constraint languages. In *Proc. 33rd STOC*, pp. 667–674. ACM Press, 2001. [doi:10.1145/380752.380868]. 86

[6] ANDREI A. BULATOV: Tractable conservative constraint satisfaction problems. In *Proc. 18th Ann. IEEE Symp. Logic in Computer Science (LICS)*, p. 321, Los Alamitos, CA, USA, 2003. IEEE Comp. Soc. Press. [doi:10.1109/LICS.2003.1210072]. 86

[7] ANDREI A. BULATOV: A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006. [doi:10.1145/1120582.1120584]. 86

[8] HUBIE CHEN AND MARTIN GROHE: Constraint satisfaction with succinctly specified relations. *J. Comput. System Sci.*, 76(8):847–860, 2010. [doi:10.1016/j.jcss.2010.04.003]. 108

[9] JIANER CHEN, BENNY CHOR, MIKE FELLOWS, XIUZHEN HUANG, DAVID W. JUEDES, IYAD A. KANJ, AND GE XIA: Tight lower bounds for certain parameterized NP-hard problems. In *Proc. 19th Ann. IEEE Conf. Comput. Complexity*, pp. 150–160. IEEE Comp. Soc. Press, 2004. [doi:10.1109/CCC.2004.1313826]. 89

[10] JIANER CHEN, XIUZHEN HUANG, IYAD A. KANJ, AND GE XIA: Linear FPT reductions and computational lower bounds. In *Proc. 36th STOC*, pp. 212–221, New York, 2004. ACM Press. [doi:10.1145/1007352.1007391]. 89

[11] VÍCTOR DALMAU, PHOKION G. KOLAITIS, AND MOSHE Y. VARDI: Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proc. 8th Intern. Conf. Principles and Practice of Constraint Programming (CP)*, pp. 310–326, London, UK, 2002. Springer-Verlag. [doi:10.1007/3-540-46135-3_21]. 103

[12] REINHARD DIESTEL: *Graph theory*. Volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005. 88

[13] RODNEY G. DOWNEY, VLADIMIR ESTIVILL-CASTRO, MICHAEL R. FELLOWS, ELENA PRIETO, AND FRANCES ROSAMOND: Cutting up is hard to do. In JAMES HARLAND, editor, *Electron. Notes Theor. Comput. Sci.*, volume 78. Elsevier, 2003. 89

[14] RODNEY G. DOWNEY AND MICHAEL R. FELLOWS: *Parameterized Complexity*. Monogr. Comput. Sci. Springer, New York, 1999. 87

[15] TOMÁS FEDER AND MOSHE Y. VARDI: The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1999. [doi:10.1137/S0097539794266766]. 86, 88, 90, 102

[16] URIEL FEIGE, MOHAMMADTAGHI HAJIAGHAYI, AND JAMES R. LEE: Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008. [doi:10.1137/05064299X]. 92, 93, 94, 95

[17] MICHAEL R. FELLOWS, DANNY HERMELIN, FRANCES A. ROSAMOND, AND STÉPHANE VIALETTE: On the parameterized complexity of multiple-interval graph problems. *Theoret. Comput. Sci.*, 410(1):53–61, 2009. [doi:10.1016/j.tcs.2008.09.065]. 89

[18] JÖRG FLUM AND MARTIN GROHE: Parameterized complexity and subexponential time. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, (84):71–100, 2004. 89

[19] JÖRG FLUM AND MARTIN GROHE: *Parameterized Complexity Theory*. Texts Theoret. Comput. Sci. EATCS Ser. Springer, Berlin, 2006. 87, 92, 103

[20] FEDOR FOMIN, PETR GOLOVACH, DANIEL LOKSHTANOV, AND SAKET SAURABH: Algorithmic lower bounds for problems parameterized by clique-width. In *Proc. 21st Ann. ACM-SIAM Symp. Discrete Algorithms (SODA 2010)*, pp. 493–502. ACM Press, 2006. 89

[21] EUGENE C. FREUDER: Complexity of *k*-tree structured constraint satisfaction problems. In *Proc. 8th National Conf. Artif. Intell. (AAAI)*, pp. 4–9, Boston, MA, 1990. AAAI Press. 86, 87

[22] OFER GABBER AND ZVI GALIL: Explicit constructions of linear-sized superconcentrators. *J. Comput. System Sci.*, 22(3):407–420, 1981. Special issued dedicated to Michael Machtey. [doi:10.1016/0022-0000(81)90040-4]. 107

[23] GEORG GOTTLOB, MARTIN GROHE, NYSRET MUSLIU, MARKO SAMER, AND FRANCESCO SCARCELLO: Hypertree decompositions: Structure, algorithms, and applications. In *Graph-Theoretic Concepts in Computer Science*, volume 3787 of *Lecture Notes in Comput. Sci.*, pp. 1–15. Springer, Berlin, 2005. [doi:10.1007/11604686_1]. 108

[24] GEORG GOTTLOB AND STEFAN SZEIDER: Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. *Comput. J.*, 51(3):303–325, 2008. [doi:10.1093/comjnl/bxm056]. 86, 108

[25] MARTIN GROHE: The structure of tractable constraint satisfaction problems. In *Proc. Mathematical Found. of Computer Science*, volume 4162 of *Lecture Notes in Comput. Sci.*, pp. 58–72. Springer, 2006. [doi:10.1007/11821069_5]. 90

[26] MARTIN GROHE: The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1, 2007. [doi:10.1145/1206035.1206036]. 86, 87, 88, 89, 102, 103, 107, 108

[27] MARTIN GROHE AND DÁNIEL MARX: Constraint solving via fractional edge covers. In *Proc. 17th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA 2006)*, pp. 289–298, New York, NY, USA, 2006. ACM Press. [doi:10.1145/1109557.1109590]. 89, 108

[28] MARTIN GROHE AND DÁNIEL MARX: On tree width, bramble size, and expansion. *J. Combin. Theory Ser. B*, 99(1):218–228, 2009. [doi:10.1016/j.jctb.2008.06.004]. 92, 106, 107

[29] MARTIN GROHE, THOMAS SCHWENTICK, AND LUC SEGOUFIN: When is the evaluation of conjunctive queries tractable? In *Proc.33rd STOC*, pp. 657–666, New York, NY, USA, 2001. ACM Press. [doi:10.1145/380752.380867]. 86, 87

[30] RUSSELL IMPAGLIAZZO, RAMAMOHAN PATURI, AND FRANCIS ZANE: Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001. [doi:10.1109/SFCS.1998.743516]. 87, 99

[31] PETER JEAVONS, DAVID A. COHEN, AND MARC GYSSENS: Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997. [doi:10.1145/263867.263489]. 86

[32] TON KLOKS: *Treewidth.* Volume 842 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 1994. 91

[33] TOM LEIGHTON AND SATISH RAO: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. [doi:10.1145/331524.331526]. 94, 95

[34] DÁNIEL MARX: Closest substring problems with small distances. *SIAM J. Comput.*, 38(4):1382–1410, 2008. [doi:10.1137/060673898]. 89

[35] DÁNIEL MARX: Tractable structures for constraint satisfaction with truth tables. In SUSANNE ALBERS AND JEAN-YVES MARION, editors, *Proc. 26th Intern. Symp. Theoretical Aspects of Computer Science (STACS)*, pp. 649–660. Schloß Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. [doi:10.4230/LIPIcs.STACS.2009.1807]. 88, 108

[36] DÁNIEL MARX: Approximating fractional hypertree width. *ACM Trans. Algorithms*, 6(2), 2010. [doi:10.1145/1721837.1721845]. 89

[37] DÁNIEL MARX: Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *Proc. 42nd STOC*, pp. 735–744. ACM Press, 2010. [doi:10.1145/1806689.1806790]. 89, 108

[38] B. A. REED: Tree width and tangles: A new connectivity measure and some applications. In R.A. BAILEY, editor, *Surveys in Combinatorics*, volume 241 of *LMS Lecture Note Series*, pp. 87–162. Cambridge University Press, 1997. [doi:10.1017/CBO9780511662119.006]. 92

[39] NEIL ROBERTSON AND P. D. SEYMOUR: Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. [doi:10.1016/0196-6774(86)90023-4]. 91

[40] NEIL ROBERTSON AND P. D. SEYMOUR: Graph minors. V. Excluding a planar graph. *J. Combin. Theory Ser. B*, 41(1):92–114, 1986. [doi:10.1016/0095-8956(86)90030-4]. 87

[41] FRANCESCO SCARCELLO, GEORG GOTTLOB, AND GIANLUIGI GRECO: Uniform constraint satisfaction problems and database theory. In *Complexity of Constraints*, pp. 156–195, 2008. [doi:10.1007/978-3-540-92800-3_7]. 86

[42] THOMAS J. SCHAEFER: The complexity of satisfiability problems. In *Proc. 10th STOC*, pp. 216–226. ACM Press, New York, 1978. [doi:10.1145/800133.804350]. 86

AUTHOR

Dániel Marx
School of Computer Science
Tel Aviv University
Tel Aviv, Israel
http://www.cs.bme.hu/~dmarx

ABOUT THE AUTHOR

DÁNIEL MARX obtained his Ph. D. in 2005 from the Budapest University of Technology and Economics under the guidance of Katalin Friedl. Since then, he has held postdoc positions at Humboldt-Universität zu Berlin, Budapest University of Technology and Economics, and Tel Aviv University. One of his ambitions is to coauthor a paper with someone whose name is Engels.