Current Algorithms for Detecting Subgraphs of Bounded Treewidth Are Probably Optimal

Karl Bringmann
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Jasper Slusallek
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

The Subgraph Isomorphism problem is of considerable importance in computer science. We examine the problem when the pattern graph H is of bounded treewidth, as occurs in a variety of applications. This problem has a well-known algorithm via color-coding that runs in time $O(n^{\mathrm{tw}(H)+1})$ [Alon, Yuster, Zwick'95], where n is the number of vertices of the host graph G. While there are pattern graphs known for which Subgraph Isomorphism can be solved in an improved running time of $O(n^{\mathrm{tw}(H)+1-\varepsilon})$ or even faster (e.g. for k-cliques), it is not known whether such improvements are possible for all patterns. The only known lower bound rules out time $n^{o(\mathrm{tw}(H)/\log(\mathrm{tw}(H)))}$ for any class of patterns of unbounded treewidth assuming the Exponential Time Hypothesis [Marx'07].

In this paper, we demonstrate the existence of maximally hard pattern graphs H that require time $n^{\mathrm{tw}(H)+1-o(1)}$. Specifically, under the Strong Exponential Time Hypothesis (SETH), a standard assumption from fine-grained complexity theory, we prove the following asymptotic statement for large treewidth t:

> For any $\varepsilon > 0$ there exists $t \geq 3$ and a pattern graph H of treewidth t such that Subgraph Isomorphism on pattern H has no algorithm running in time $O(n^{t+1-\varepsilon})$.

Under the more recent 3-uniform Hyperclique hypothesis, we even obtain tight lower bounds for each specific treewidth $t \geq 3$:

> For any $t \geq 3$ there exists a pattern graph H of treewidth t such that for any $\varepsilon > 0$ Subgraph Isomorphism on pattern H has no algorithm running in time $O(n^{t+1-\varepsilon})$.

In addition to these main results, we explore **(1)** colored and uncolored problem variants (and why they are equivalent for most cases), **(2)** Subgraph Isomorphism for $\mathrm{tw} < 3$, **(3)** Subgraph Isomorphism parameterized by pathwidth instead of treewidth, and **(4)** a weighted variant that we call Exact Weight Subgraph Isomorphism, for which we examine pseudo-polynomial time algorithms. For many of these settings we obtain similarly tight upper and lower bounds.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Computational complexity and cryptography

Keywords and phrases subgraph isomorphism, treewidth, fine-grained complexity, hyperclique

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.40

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2105.05062`

Funding *Karl Bringmann*: This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979).

1 Introduction

The Subgraph Isomorphism problem is commonly defined as follows: Given a graph H on k vertices, and a graph G on n vertices, is there a (not necessarily induced) subgraph of G which is isomorphic to H?

Subgraph Isomorphism generalizes many problems of independent interest, such as the k-path and k-clique problems. The problem is also of considerable interest when H is less structured, with applications to discovering patterns in graphs that, for example, arise from biological processes such as gene transcription or food networks, from social interaction, from electronic circuits, from neural networks [42], from chemical compounds [47] or from control flow in programs [19]. In some fields, the problem is sometimes referred to as the search for "network motifs", i.e. subgraphs that appear more often than would normally be expected.

In its general form, the problem is NP-hard. We are interested in solving the problem when the pattern graph H is "tree-like" or "path-like", i.e. when the treewidth $\mathrm{tw}(H)$ or the pathwidth $\mathrm{pw}(H)$ of H is bounded. Such pattern graphs of low treewidth or pathwidth often arise in practice when considering the structure of chemical compounds, the control flow of programs, syntactic relations in natural language, or many other graphs from practical applications (see e.g. [14, 16]). On the theoretical side, many restricted classes of graphs have bounded treewidth, see also [15]. Restricting NP-hard problems to graphs of bounded tree- and pathwidth often yields polynomial-time algorithms, and Subgraph Isomorphism is no exception. Most notably, the classic Color-Coding algorithm by Alon, Yuster and Zwick [9] solves the problem by a Las Vegas algorithm in expected time $O(n^{\mathrm{tw}(H)+1}g(k))$, or by a deterministic algorithm in time $\widetilde{O}(n^{\mathrm{tw}(H)+1}g(k))$, where g is a computable function (and $\widetilde{O}(\cdot)$ is used to suppress factors that are polylogarithmic in the input size). In other words, if the pattern graph H has treewidth bounded by some constant, the problem is fixed-parameter tractable when parameterized by k. The Color-Coding algorithm is also relevant for practical purposes: Recently, it has received an efficient implementation, which tested well against state-of-the-art programs for Subgraph Isomorphism [39].

Many researchers wondered whether the Color-Coding algorithm can be improved. This question has been studied in many different directions, including the following:

- Marx [40] showed that no algorithm solves the Subgraph Isomorphism problem in time $O(n^{o(\mathrm{tw}(H)/\log(\mathrm{tw}(H)))}g(k))$ unless the Exponential Time Hypothesis (ETH) fails, and this even holds when restricted to any class of pattern graphs of unbounded treewidth.
- A series of work has improved the computable function g, see e.g. [10, 29, 44].
- For many special pattern graphs faster algorithms have been found; the most famous example is the k-Clique problem, which can be solved in time $O(n^{k\omega/3}g(k))$ [43][1].

In this paper, we use a different angle to approach the question whether Color-Coding can be improved. We ask whether there exist "hard" pattern graphs:

Do there exist pattern graphs H for which Subgraph Isomorphism cannot be solved in time $O(n^{\mathrm{tw}(H)+1-\varepsilon})$ for any constant $\varepsilon > 0$?

To the best of our knowledge, this question has not been previously studied. As our main result, we (conditionally) give a positive answer to this question. More precisely, we show that for every $t \geq 3$ there exists a pattern graph H with $\mathrm{tw}(H) = t$ for which Subgraph Isomorphism cannot be solved in time $O(n^{\mathrm{tw}(H)+1-\varepsilon})$ for any constant $\varepsilon > 0$, assuming the 3-uniform k-Hyperclique hypothesis; see Section 1.3 for details on this hypothesis. We also show a slightly weaker statement under the Strong Exponential Time Hypothesis. This conditionally shows that the Color-Coding algorithm by Alon, Yuster and Zwick cannot be significantly improved while still working for all pattern graphs.

[1] This bound assumes that k is divisible by 3; there are similar results for general k [27].

For the case of $\mathrm{tw}(H) = 2$, an algorithm of Curticapean, Dell and Marx [24] can be adapted such that it solves Subgraph Isomorphism in time $\widetilde{O}(n^\omega g(k))$. We unify this with the algorithm of Alon, Yuster and Zwick by showing that both time bounds can be achieved within a simple framework. In particular, we use so-called k-wise matrix products, an operation which was introduced in its general form in [31] and studied further in [38].

We also study the Subgraph Isomorphism problem when the pathwidth of H is bounded, and specialize our framework to show slight improvements in running time compared to the case of bounded treewidth. Here, we use rectangular matrix products, for which faster-than-naive algorithms are known [30].

In further results, our focus is on the weighted variant Exact Weight Subgraph Isomorphism, where the subgraph must also have total weight equal to zero. In this work, we consider both the node-weighted and the edge-weighted variant of this problem, for both bounded treewidth and bounded pathwidth, allowing the maximum absolute weight W to appear in the running time (i.e. the pseudopolynomial-time setting). We show that our algorithms for the unweighted case can be adapted to the weighted case. We also speed up the weighted algorithms by using the fact that fast convolution (or rather, sumset computation), a folklore technique that lies at the core of many fast algorithms for problems with weights (e.g. [20, 18, 35, 34, 17, 12] and [23, exercise 30.1.7]), can easily be adapted to work with rectangular matrices and tensors. We furthermore show tight conditional lower bounds in many cases. Last but not least, we show that our algorithms can be slightly improved for the case of node-weighted instances for which either the pathwidth of H is bounded, or H is a tree. These algorithms also rely on fast rectangular matrix products.

1.1 Related Work

Additional to the conditional lower bound of $O(n^{o(\mathrm{tw}(H)/\log(\mathrm{tw}(H)))}g(k))$ by Marx [40], there is an unconditional lower bound of $O(n^{\kappa(H)})$ for the size of any AC^0-circuit, for some graph parameter $\kappa(H) = \Omega(\mathrm{tw}(H)/\log(\mathrm{tw}(H)))$, which holds even when considering the average case [37]. Interestingly, the factor of $1/\log(\mathrm{tw}(H))$ does not seem to be an artefact of the proof: There is an AC^0-circuit of size $O(n^{o(\mathrm{tw}(H))}g(k))$ that solves the problem on certain unbounded-treewidth classes in the average case [45].

In a different direction, Dalirrooyfard et al. [25] design various reductions from k-Clique to Subgraph Isomorphism, among other results. They also present results on the detection of induced subgraphs (we focus on non-induced subgraphs).

For the weighted variant of Subgraph Isomorphism, lower bounds under the k-Sum hypothesis for stars, paths, cycles and some other pattern graphs are presented in [5]. Edge-weighted triangle detection has a by-now classic $O(n^{3-\varepsilon})$ lower bound under both the 3Sum hypothesis and the APSP hypothesis [7]. On the other hand, in [6], it is proven that finding node-weighted k-cliques can be done almost as quickly as finding unweighted k-cliques. We are not aware of any results on the Exact Weight Subgraph Isomorphism problem when W may appear in the running time (i.e. a pseudopolynomial-time algorithm), which is what we focus on here.

In our work, we pose no restrictions on the host graph G. For an extensive classification of Subgraph Isomorphism with respect to various parameters of both G and H, see [41].

1.2 Hardness Assumptions

The most standard hypothesis from fine-grained complexity theory is the Strong Exponential Time Hypothesis (SETH) [32], which postulates that for any $\varepsilon > 0$ there exists $k \geq 3$ such that k-Sat on n variables cannot be solved in time $O^*(2^{(1-\varepsilon)n})$.

More recent is the Hyperclique hypothesis. In the h-uniform k-Hyperclique problem, for a given h-uniform hypergraph we want to decide whether there exist a set of k vertices such that every size-h subset of these vertices forms a hyperedge. For any $k > 3$, the 3-uniform k-Hyperclique hypothesis postulates that this problem cannot be solved in time $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$. This hypothesis has also been formulated when replacing 3 with any $h < k$, getting progressively more believable with larger h. For a more in-depth discussion of the believability of this hypothesis we refer to [38, Section 7].

Note that we will also use the h-uniform Hyperclique hypothesis for various h, which is simply the conjecture that the h-uniform k-Hyperclique hypothesis is true for all $k > h$.

Related to this is the k-Clique conjecture, which postulates that the k-Clique problem (which is the 2-uniform k-Hyperclique problem) cannot be solved in time $O(n^{\omega k/3-\varepsilon})$ for any constant $\varepsilon > 0$, where $\omega < 2.373$ [36] is the exponent of matrix multiplication.

1.3 Our Results

Unweighted Subgraph Isomorphism with Bounded Treewidth. First, consider the case of the unweighted Subgraph Isomorphism problem for bounded-treewidth pattern graphs H. As was said, and as we will re-prove with a unified algorithm later, this problem has an algorithm running in time $\widetilde{O}(n^{\mathrm{tw}(H)+1})$ for $\mathrm{tw}(H) \geq 3$. We show tight conditional lower bounds by proving the following obstacles to faster algorithms, which use the k-clique hypothesis and the h-uniform k-hyperclique hypothesis. Note that when we say, for some x, that an algorithm has running time $O(n^{x-\varepsilon})$, what we mean is that the algorithm runs in time $O(n^{x-\varepsilon})$ for some constant $\varepsilon > 0$.

▶ **Theorem 1.** *The following statements are true.*

1. *For each $t \geq 3$ and each $3 \leq h \leq t$, there exists a connected, bipartite pattern graph $\mathcal{H}_{t,h}$ of treewidth t such that there cannot be an algorithm solving the Subgraph Isomorphism problem on pattern graph $\mathcal{H}_{t,h}$ in time $O(n^{t+1-\varepsilon})$ unless the h-uniform $h(t+1)$-hyperclique hypothesis fails.*
2. *For each $t \geq 2$ and each $h \geq 3$, there exists a connected, bipartite pattern graph $\mathcal{H}_{t,h}$ of treewidth t such that there cannot be an algorithm solving the Subgraph Isomorphism problem on pattern graph $\mathcal{H}_{t,h}$ in time $O(n^{t-\varepsilon})$ unless the h-uniform ht-hyperclique hypothesis fails.*
3. *For each $t \geq 2$, there exists a connected, bipartite pattern graph $\mathcal{H}_t$ of treewidth t such that there cannot be an algorithm solving the Subgraph Isomorphism problem on pattern graph $\mathcal{H}_t$ in time $O(n^{(t+1)\omega/3-\varepsilon})$ unless the $(t+1)$-Clique hypothesis fails.*

Indeed, with the very same reduction, we also get an obstacle from SETH. However, the lower bound it provides is not as tight as the above, and in the case of the second part does not work for each target treewidth t.

▶ **Theorem 2.** *Assuming SETH, the following two statements are true.*

1. *For any $t \geq 3$ and any $\varepsilon > 0$ there exists a pattern graph $\mathcal{H}_{t,\varepsilon}$ of treewidth t such that there cannot be an algorithm solving all instances of Subgraph Isomorphism with pattern graph $\mathcal{H}_{t,\varepsilon}$ in time $O(n^{t-\varepsilon})$.*
2. *For any $\varepsilon > 0$ there exists a $t \geq 3$ and a pattern graph $\mathcal{H}_\varepsilon$ of treewidth t such that there cannot be an algorithm solving all instances of Subgraph Isomorphism with pattern graph $\mathcal{H}_\varepsilon$ in time $O(n^{t+1-\varepsilon})$.*

On the algorithmic side, we present an algorithm that achieves matching running times (as listed in Theorem 3 below). As was said, the results in the following theorem are not new. Part 1 was shown via Color-Coding in [9] and part 2 follows from techniques in [24].

We unify these two results by providing a single, relatively simple algorithmic technique achieving both, based on k-wise matrix products. These techniques are later expanded to also work for the weighted version, where they then achieve new results. In the following, $\omega < 2.373$ [36] is the exponent of matrix multiplication.

▶ **Theorem 3.** *There are algorithms which, given an arbitrary instance $\phi = (H, G)$ of* Subgraph Isomorphism *where H has treewidth $\mathrm{tw}(H)$, solve ϕ in*

1. *time $\widetilde{O}(n^{\mathrm{tw}(H)+1}g(k))$ when $\mathrm{tw}(H) \geq 3$,*
2. *time $\widetilde{O}(n^{\omega}g(k))$ when $\mathrm{tw}(H) = 2$, and*
3. *time $\widetilde{O}(n^{2}g(k))$ when $\mathrm{tw}(H) = 1$,*

where $k := |V(H)|, n := |V(G)|$ and g is a computable function.

Semi-Equivalence of Hyperclique and Subgraph Isomorphism. In the full version of the paper, we also discuss how our results not only show a reduction from Hyperclique to Subgraph Isomorphism with bounded treewidth, but also in the other direction. For this, we show that calculating the boolean k-wise matrix products, which is the bottleneck in our algorithm for bounded-treewidth Subgraph Isomorphism, is actually equivalent to the k-uniform $(k+1)$-Hypergraph problem. Hence we have a reduction in the second direction. This gives an interesting intuition for why the Hyperclique hypothesis is the "correct" conjecture to prove conditional hardness of Subgraph Isomorphism for bounded treewidth.

We remark that this does not lead to a full equivalence of these problems because the uniformity (i.e. the size of hyperedges) of the Hyperclique problem we reduce from in the first reduction is much smaller than the size of the hypercliques we search for. Hence we only have a reduction from a Hyperclique instance with small edge uniformity to Subgraph Isomorphism, and a reduction from Subgraph Isomorphism to Hyperclique instances with large edge uniformity.

Weighted Subgraph Isomorphism with Bounded Treewidth. Now consider the weighted version of Subgraph Isomorphism for bounded-treewidth graphs H. Recall that the weighted version can be either node- or edge-weighted and is defined such that the weights in the solution subgraph must have total weight zero. A trivial dynamic programming algorithm on the tree decomposition achieves a running time of $\widetilde{O}(n^{\mathrm{tw}(H)+1} \cdot W \log W)$ for $\mathrm{tw}(H) \geq 3$.

Note that these results show conditional lower bounds even when the maximum weight is restricted to $W = \Theta(n^{\gamma})$, for any constant $\gamma > 0$.

▶ **Theorem 4.** *For both the node- and edge weighted variant of the problems, the following statements are true.*

1. *For each $t \geq 3$, each $\gamma \in \mathbb{R}^+$ and each $3 \leq h \leq t$, there exists a connected, bipartite graph $\mathcal{H}_{t,h,\gamma}$ of treewidth t such that there cannot be an algorithm solving the Exact Weight Subgraph Isomorphism problem on pattern graph $\mathcal{H}_{t,h,\gamma}$ for instances with maximum weight $W = \Theta(n^{\gamma})$ in time $O(n^{t+1-\varepsilon}W)$, unless the h-uniform Hyperclique hypothesis fails.*
2. *For each $t \geq 1$, each $\gamma \in \mathbb{R}^+$ and each $h \geq 3$, there exists a connected, bipartite graph $\mathcal{H}_{t,h,\gamma}$ of treewidth t such that there cannot be an algorithm solving the Exact Weight Subgraph Isomorphism problem on pattern graph $\mathcal{H}_{t,h,\gamma}$ for instances with maximum weight $W = \Theta(n^{\gamma})$ in time $O(n^{t-\varepsilon}W)$, unless the h-uniform Hyperclique hypothesis fails.*
3. *For each $t \geq 1$ and each $\gamma \in \mathbb{R}^+$, there exists a connected, bipartite graph $\mathcal{H}_{t,\gamma}$ of treewidth t such that there cannot be an algorithm solving the Exact Weight Subgraph Isomorphism problem on pattern graph $\mathcal{H}_{t,\gamma}$ for instances with maximum weight $W = \Theta(n^{\gamma})$ in time $O(n^{(t+1)\omega/3-\varepsilon}W^{\omega/3})$, unless the Clique hypothesis fails.*

Similar lower bounds also hold when trying to reduce the exponent of W instead of n. Meaning there is also no algorithm of running time $O(n^{t+1}W^{1-\varepsilon})$ in part 1, etc.

On the algorithmic side, we present an algorithm that achieves matching running times for $\mathrm{tw}(H) \geq 3$, and almost matching running times for $\mathrm{tw}(H) = 1, 2$. Note that in terms of exponents, the first algorithm below is not better than the naive one with running time $O(n^{\mathrm{tw}+1}W \log W)$. However, it avoids a factor of $\log W$ in the largest term, and instead appends it to a smaller term, so in a way it presents an improvement of $\log W$ in the running time. Specifically, we show

▶ **Theorem 5.** *There are algorithms which, given an arbitrary instance $\phi = (H, G, w)$ of the* Exact Weight Subgraph Isomorphism *problem where H has treewidth $\mathrm{tw}(H)$, solve ϕ in*
1. *time $\widetilde{O}((n^{\mathrm{tw}(H)+1}W + n^{\mathrm{tw}(H)}W \log W)g(k))$ when $\mathrm{tw}(H) \geq 3$,*
2. *time $\widetilde{O}((n^{\omega}W + n^2 W \log W)g(k))$ when $\mathrm{tw}(H) = 2$, or*
3. *time $\widetilde{O}((n^2 W + nW \log W)g(k))$ when $\mathrm{tw}(H) = 1$,*

where $n := |V(G)|, k := |V(H)|$, g is a computable function, and W is the maximum absolute weight in the image of w.

Comparing these upper bounds with the lower bounds from Theorem 5, we have a tight lower bound for the weighted case with $\mathrm{tw}(H) \geq 3$. For weighted $\mathrm{tw}(H) = 2$, we have a lower bound which is tight except for the exponent of $\omega/3$ to W; it is unclear whether this can be strengthened. The lower bound for weighted graphs with $\mathrm{tw}(H) = 1$ is obviously not tight: We have an upper bound of $\widetilde{O}(n^2 W + nW \log W)$, but our lower bounds only states that it requires time $O(n^{1-o(1)}W^{1-o(1)})$ and $O(n^{2\omega/3-o(1)}W^{\omega/3-o(1)})$. Tighter lower bounds for this case remain an important open problem.

Unweighted Subgraph Isomorphism with Bounded Pathwidth. So far we have only looked at the case of bounded treewidth. However, similar results hold for the case of bounded pathwidth. Let us start with the unweighted Subgraph Isomorphism problem.

Note that we do not get any lower bounds for the current setting. This is because we prove all our lower bounds by showing an equivalence of the standard Subgraph Isomorphism problem to a colored variant (see also Section 1.4), and then proving a lower bound for the colored version. We do not know how to prove such an equivalence for the current setting, therefore we do not get lower bounds in this case; we leave this as an open problem.

Since a path decomposition is always also a tree decomposition, we trivially get upper bounds as in Theorem 3 (when replacing treewidth by pathwidth). However, we can do better by using rectangular matrix multiplication to speed up the computation. For $z \in \mathbb{R}^+$, let $\omega(z)$ be the smallest real number such that multiplying a $n \times n$ matrix with a $n \times n^z$ matrix can be done in time $O(n^{\omega(z)})$[2]. We prove the following upper bounds.

▶ **Theorem 6.** *There are algorithms which, given an arbitrary instance $\phi = (H, G)$ of* Subgraph Isomorphism *where H has pathwidth p, solve ϕ in*
1. *time $\widetilde{O}(n^{\omega(p-1)}g(k))$ when $p \geq 2$, and*
2. *time $\widetilde{O}(n^2 g(k))$ when $p = 1$*

where $k := |V(H)|$ and $n := |V(G)|$.

[2] Le Gall [30] has shown that there are fast algorithms for rectangular matrix multiplication based on the Coppersmith-Winograd method [22, 36] used for square matrix multiplication. Among other values, he shows $\omega(0.31) = 2$, $\omega(2) < 3.26$, $\omega(3) < 4.2$, $\omega(4) < 5.18$ and $\omega(5) < 6.16$. See [30] for an extensive table of such values.

We certainly have $p \leq \omega(p-1) < p+1$, so these results represent only a minor improvement, which is nonetheless important because it "beats" the lower bound for treewidth. Hence the lower bound for pathwidth cannot be the same as for treewidth.

Weighted Subgraph Isomorphism with Bounded Pathwidth. We also analyze the bounded-pathwidth pattern graph version of Weighted Subgraph Isomorphism. Specifically, we get the following lower bound.

▶ **Theorem 7** (Theorem 4 for pathwidth)**.** *Parts 2 and 3 of Theorem 4 also hold when replacing the treewidth t by the pathwidth p. Part 1 does not hold.*

And on the algorithmic side, we can again use rectangular matrix multiplication to improve on the algorithms from the case of bounded treewidth. Specifically, we get:

▶ **Theorem 8.** *There are algorithms which, given an arbitrary instance $\phi = (H, G, w)$ of the Exact Weight Subgraph Isomorphism problem, solve ϕ in*

1. *time $\widetilde{O}((n^{\omega(\mathrm{pw}(H)-1)}W + n^{\mathrm{pw}(H)}W \log W)g(k))$ when $\mathrm{pw}(H) \geq 2$,*
2. *time $\widetilde{O}((n^2W + nW \log W)g(k))$ when $\mathrm{pw}(H) = 1$*

where $n := |V(G)|, k := |V(H)|$, and W is the maximum absolute weight in the image of w.

For $\mathrm{pw}(H) \geq 3$, the lower bounds are therefore obviously not tight (at least for current algorithms), unless significant advances in matrix multiplication techniques are made. For $\mathrm{pw}(H) = 1, 2$, the situation is the same as with treewidth, see the discussion of Theorem 5.

Improvements to Special Cases of Weighted Subgraph Isomorphism. It is natural to think that the exponents $\frac{\omega}{3}$ to W in the lower bounds of Theorems 4 and 7 are only artefacts of the reduction, and that with more advanced methods, this exponent can be improved to 1. However, the following two theorems show that this notion is false for $\mathrm{tw}(H) = 1$ and $\mathrm{pw}(H) = 1, 2$, at least when considering the node-weighted case. Indeed, for $\mathrm{tw}(H) = 1$ (or $\mathrm{pw}(H) = 1$) and $W = n$, these bounds are tight, so further general improvements on the exponent are impossible.

Specifically, Theorems 9 and 10 show the following improvements of the algorithms from Theorems 5 and 8 for small tree- or pathwidth. Let $\mathrm{MM}(n, n, x)$ be the time in which a $n \times n$ matrix can be multiplied with with a $n \times x$ matrix.

▶ **Theorem 9.** *There is an algorithm which, given an arbitrary instance $\phi = (H, G, w)$ of the node-weighted Exact Weight Subgraph Isomorphism problem where H is a tree, solves ϕ in time $\widetilde{O}((\mathrm{MM}(n, n, W) + nW \log W)g(k))$.*

▶ **Theorem 10.** *There is an algorithms which, given an arbitrary instance $\phi = (H, G, w)$ of the node-weighted Exact Weight Subgraph Isomorphism problem, solves ϕ in time $\widetilde{O}(\mathrm{MM}(n, n, n^{\mathrm{pw}(H)-1}W)g(k))$.*

For $W = O(n^\gamma)$, the running time of Theorem 9 is $\widetilde{O}(n^{\omega(\gamma)}\,\mathrm{poly}(k))$. This implies several interesting facts. One such fact is that due to the known convergence $\lim_{\gamma\to\infty} \omega(\gamma) - \gamma = 1$ [21], we have that for node-weighted trees, there cannot be a lower bound of $O(n^{(1+\varepsilon-o(1)}W)$ for any $\varepsilon > 0$ that holds for any constant $\gamma > 0$. A similar result holds for bounded-pathwidth graphs. Other implications are discussed in the full version.

1.4 Equivalence of the Colored and Uncolored Problems

All mentioned algorithms and conditional lower bounds are shown for the restricted problem of Colored Subgraph Isomorphism, where the nodes of G and H are colored with $|V(H)|$ colors and the isomorphism must preserve colors, as also studied in [40]. In the full version of the paper, we prove that the standard and the colored variant of Subgraph Isomorphism can be solved in essentially the same running time in almost all cases. Specifically, we show the following lemma.

▶ **Lemma 11.** *Let ρ be any graph parameter.*

1. *If there is a $T(n,k,\rho(H))$ time algorithm for Colored Subgraph Isomorphism, then there is a $\widetilde{O}(T(kn,k,\rho(H))g(k))$ time algorithm for Subgraph Isomorphism, where g is some computable function.*
2. *If there is a $T(n,k,\rho(H),W)$ time algorithm for Exact Weight Colored Subgraph Isomorphism, then there is a $\widetilde{O}(T(kn,k,\rho(H),W)g(k))$ time algorithm for Exact Weight Subgraph Isomorphism, where g is some computable function.*
3. *Let $\mathrm{tw}(H) \geq 2$. If there is a $T(n,k,\mathrm{tw}(H))$ time algorithm for Subgraph Isomorphism, then there is a $O(T(\mathrm{poly}(k)n,\mathrm{poly}(k),\mathrm{tw}(H)) + \mathrm{poly}(k)n^2)$ time algorithm for Colored Subgraph Isomorphism.*
4. *If there is a $T(n,k,\rho(H),W)$ time algorithm for Exact Weight Subgraph Isomorphism, then there is a $O(T(2n,2k,\rho(H),2^kW) + \mathrm{poly}(k)n^2)$ time algorithm for Exact Weight Colored Subgraph Isomorphism.*

This lemma enables us to prove results for (Exact Weight) Subgraph Isomorphism while only talking about the more structured colored variants of the problem.

2 Technical Overview of Our Main Results

We now give proof sketches of our main lower bound results. We do not give proofs for the upper bound results due to their technicality. The full version of this paper contains all proofs with full detail.

2.1 Lower Bound for Subgraph Isomorphism

Our main result is the existence of the hard pattern graphs for bounded-treewidth Subgraph Isomorphism. We now prove their existence for treewidth at least 3 under the Hyperclique hypothesis, i.e. part 1 of Theorem 1.

The exact statement we prove is that for each $t \geq 3$ and each $3 \leq h \leq t$, there exists a pattern graph of treewidth t such that Subgraph Isomorphism cannot be solved in time $O(n^{t+1-\varepsilon})$ on that pattern graph unless the h-uniform $h(t+1)$-hyperclique hypothesis fails. Note that the proof actually shows this for the colored variant of Subgraph Isomorphism, after which we can use Lemma 11 to transfer the lower bound to the uncolored problem.

Proof sketch. See Figure 1 for a sketch of the reduction. Let $t \geq 3$ and $3 \leq h \leq t$ be given and assume that Subgraph Isomorphism can be solved in time $O(n^{t+1-\varepsilon})$ on pattern graphs of treewidth t. We show that the h-uniform $h(t+1)$-hyperclique hypothesis fails.

Construction of H. We construct a pattern graph H as a bipartite graph with vertex set $A \cup B$ as follows. Writing $[c] := \{1,\dots,c\}$, we set $A := [t+1]$ and $B := \binom{A \times [h]}{h}$. We connect a vertex $b = ((a_1,j_1),..,(a_h,j_h))$ in B to a vertex a in A if $a = a_\ell$ for some ℓ. Set $k := |A| + |B|$.

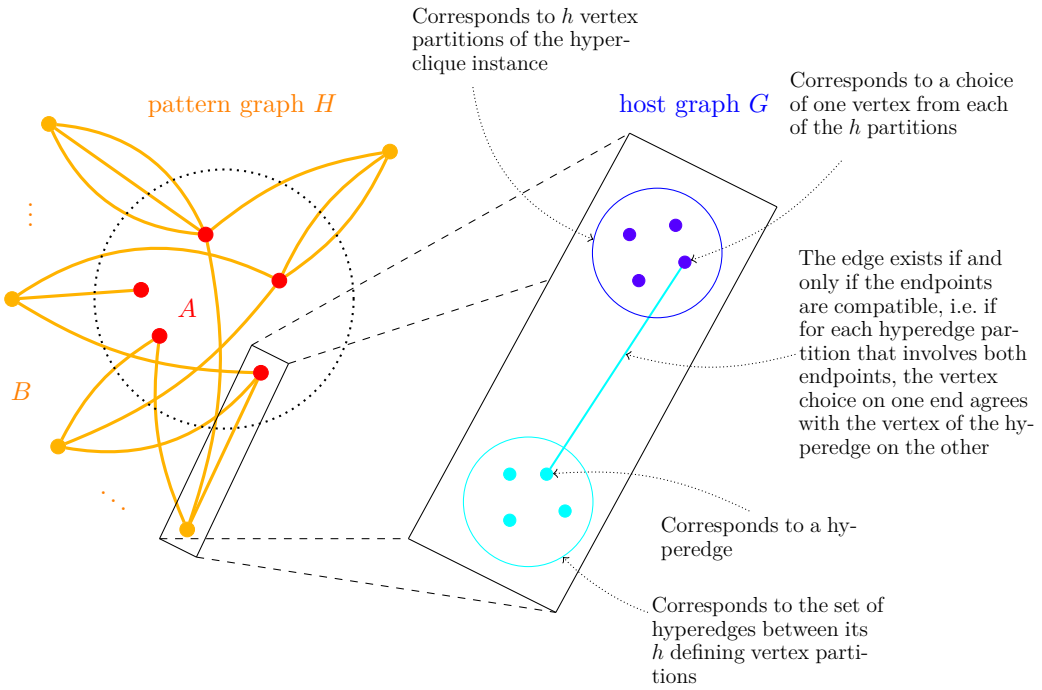

Figure 1 A sketch of the reduction we use to prove part 1 of Theorem 1 where $h = 3$ and $t = 4$. Note that this is only a partial sketch of the pattern graph. We use multiedges to signify that for the endpoints $a \in A$ and $b = ((a_1, j_1), \ldots, (a_h, j_h)) \in B$ there exists more than one ℓ such that $a = a_\ell$.

We show that this pattern has a treewidth of t, via a well-known characterization of treewidth as a graph-theoretic game: A graph F has treewidth $\leq t$ if and only if $t + 1$ cops can catch[3] a robber on F [46]. To show the bound on the treewidth of H, initially place a cop on each vertex of A. No matter on which vertex of B the robber starts, they are surrounded by cops. Since every vertex in B has $h \leq t < t + 1$ neighbors in A, there must exist some cop which is not adjacent to the robber, so this cop can catch the robber in a single step. This concludes the proof that the pattern graph H has treewidth t.

Construction of G. Now given a $h(t + 1)$-partite hypergraph H', i.e. an instance of the h-uniform k'-HYPERCLIQUE problem for $k' := h(t + 1)$, we write the vertex set of H' as $U_{1,1} \cup \ldots \cup U_{1,h} \cup \ldots \cup U_{t+1,1} \cup \ldots \cup U_{t+1,h}$. Let N_H be the number of vertices vertices in each partition and $n_H = O(N_H)$ the number of vertices overall.

We construct a k-partite graph G as follows. For a in A we set $V_a := U_{a,1} \times \ldots \times U_{a,h}$. For $b = ((a_1, j_1), .., (a_h, j_h))$ in B, we set $V_b := E(H') \cap (U_{a_1,j_1} \times \ldots \times U_{a_h,j_h})$. This describes the k parts of the k-partite vertex set $V(G)$. Note that each part has size at most $N_G := N_H^h$. Now we construct the edges. For any a in A and $b = ((a_1, j_1), .., (a_h, j_h))$ in B with (a, b) in $E(H)$, consider an arbitrary $u = (u_1, .., u_h)$ in V_a and $u' = (u'_1, ..., u'_h)$ in V_b. We say that u and u' are "compatible" if for every ℓ with $a_\ell = a$ we have $u'_\ell = u_{j_\ell}$; in this case we connect u and u' by an edge. This finishes the construction of G.

[3] The game works as follows: The $k + 1$ cops select their starting vertices in the graph. Then the robber may choose their starting vertex. The cops can always see the robber and adapt their strategy accordingly. Similarly, the robber can see the cops. The game now proceeds in steps, where in each step, one of the cops chooses an arbitrary destination vertex and takes off via helicopter in the direction of that vertex. While the cop is travelling, the robber sees where they will land and may now move arbitrarily along edges of the graph, as long as they do not pass through stationary cops. When the robber has finished moving, the cop lands. The cops win if and only if they are guaranteed to catch the robber after a finite number of moves, and lose otherwise.

Correctness. Note that any colored subgraph isomorphism of H in G chooses vertices v_a in V_a for all a in A. This corresponds to choosing vertices $u_{i,j}$ in $U_{i,j}$ for all i in $[t+1]$, Moreover, the edges of a $h(t+1)$-hyperclique are in one-to-one correspondence with the set B. Since for each b in B the colored subgraph isomorphism of H in G needs to choose a vertex v_b in V_b, which corresponds to an edge between certain vertices $u_{i,j}$, we indeed check that the chosen vertices $u_{i,j}$ form an h-uniform $h(t+1)$-hyperclique.

Running Time. Trivially, the construction time and output size are $O(N_H^{2h})$ (actually, it is slightly better, but this is not important in this proof sketch), and G has $n = O(N_G) = O(N_H^h)$ vertices. Now if we can solve Subgraph Isomorphism in time $O(n^{t+1-\varepsilon})$, we solve the h-uniform $h(t+1)$-Hyperclique instance in time $O(N_H^{2h} + (N_H^h)^{t+1-\varepsilon}) = O(N_H^{h(t+1)-h\varepsilon}) = O(N_H^{h(t+1)-\varepsilon'}) = O(n_H^{h(t+1)-\varepsilon'})$. ◀

This shows part 1 of Theorem 1. Part 2 can be shown by the almost the exact same proof, except that now we choose the size of A to be t instead of $t+1$, and we start with an ht-hyperclique instead of an $h(t+1)$-hyperclique. It can be seen that in this case, the pattern graph still has treewidth t. The third part of the theorem can be seen by simply taking an instance of $(t+1)$-Clique and subdividing the edges in the obvious way to make the graph bipartite.

Let us also quickly mention how the proof of the slightly weaker bounds under SETH, i.e. Theorem 2, works. The split-and-list technique from [48] allows one to reduce the Satisfiability problem to Hyperclique. Using this technique, the following result was shown in [38, Lemma 9.1].

▶ **Lemma 12** ([38]). *Assuming SETH, for any $\varepsilon > 0$ there exists $h \geq 3$ such that for all $k > h$, the h-uniform k-Hyperclique problem is not in time $O(n^{k-\varepsilon})$.*

The SETH result now follows by using essentially the same reduction as above, but we prefix it by the reduction from SAT to Hyperclique.

2.2 Lower Bound for Exact Weight Subgraph Isomorphism

We also give lower bounds for the exact weight variant of the Subgraph Isomorphism problem. In particular, we prove the existence of hard pattern graphs for the bounded-treewidth Exact Weight Subgraph Isomorphism problem for any polynomial weight bound. We give this result for any treewidth which is at least 3, and under the Hyperclique hypothesis. This is part 1 of Theorem 4.

The exact statement we prove is that for each $t \geq 3$, $\gamma \in \mathbb{R}^+$ and $3 \leq h \leq t$, there exists a pattern graph of treewidth t such that Exact Weight Subgraph Isomorphism with maximum weight $W = \Theta(n^\gamma)$ cannot be solved in time $O(n^{t+1-\varepsilon}W)$ unless the h-uniform Hyperclique hypothesis fails. Again, we show this statement for the colored problem and transfer the lower bound via Lemma 11.

To do this, we will encode part of a hyperclique instance in the edges of the Exact Weight Subgraph Isomorphism problem, and the rest of the instance in the weights. To do the latter, we need to encode certain equality constraints only via weights. This can be done using so-called k-average free sets[4], which we define below.

[4] These k-average-free sets are a tool which are very useful for weighted problems, especially when they have additive elements. Such problems include k-sum, Subset Sum, Bin Packing, various scheduling problems, Tree Partitioning, Max-Cut, Maximum/Minimum Bisection, a Dominating Set variant with capacities, and similar [3, 4, 6, 11, 28, 33, 26]. Other uses of k-average-free sets in computer science include constructions in extremal graph theory, see e.g. [1, 2, 8].

▶ **Definition 13** (k-average free sets). *A set $S \subseteq \mathbb{Z}$ is called k-average-free if, for any $s_1, \ldots, s_{k'+1} \in S$ with $k' \leq k$, we have $s_1 + \ldots + s_{k'} = k' \cdot s_{k'+1}$ if and only if $s_1 = \ldots = s_{k'+1}$. In other words, the average of $s_1, \ldots, s_{k'} \in S$ is in S if and only if all s_i are equal.*

We use the following construction for k-average free sets, originally proven in [13], modified into a more useful version in [6] and formulated in this form in [3].

▶ **Lemma 14.** *There exists a universal constant $c > 0$ such that, for all constants $\varepsilon \in (0,1)$ and $k \geq 2$, a k-average-free set S of size n with $S \subseteq [0, k^{c/\varepsilon} n^{1+\varepsilon}]$ can be constructed in time* $\mathrm{poly}(n)$.

Let us now prove the statement about Exact Weight Subgraph Isomorphism. We will construct an instance that is node-weighted, however this can easily be converted into an edge-weighted version by moving the weight of each vertex to all of its incident edges.

Proof sketch. Let $t \geq 3$, $3 \leq h \leq t$ and $\gamma \in \mathbb{R}$ be given and assume that Exact Weight Subgraph Isomorphism can be solved in time $O(n^{t+1-\varepsilon}W)$ on instances where the pattern graph has treewidth t and all weights are bounded by $W = \Theta(n^\gamma)$. We show that the h-uniform k-hyperclique hypothesis fails for some large enough k.

Construction of H. We construct a pattern graph H as a graph with vertex set $(A_1 \cup A_2) \cup B$ as follows. We set $A_1 := [t+1]$, $A_2 := [r]$ (for some r large enough) and $B := \binom{(A_1 \cup A_2) \times [h]}{h}$. We connect a vertex $b = ((a_1, j_1), \ldots, (a_h, j_h))$ in B to a vertex a in A_1 (not in A_2) if $a = a_\ell$ for some ℓ. Set $k := |A_1| + |A_2| + |B|$. By almost the same proof as in the unweighted version, it can be shown that this pattern H has treewidth t.

Grouping partitions. Now let an instance of the h-uniform k'-Hyperclique problem be given, and write the vertex set of H' as $U_1 \cup \ldots \cup U_k$. Let N_H be the number of vertices vertices in each partition and $n_H = O(N_H)$ the number of vertices overall.

We construct the k-partite graph G with at most some number N_G of vertices in each partition as follows. We will encode a β-fraction of the Hyperclique instance in the weights of the final Exact Weight Subgraph Isomorphism instance, and a $(1-\beta)$-fraction in the edges, for some β chosen appropriately. To do this, we will choose β such that $\frac{\beta k'}{hr}, \frac{(1-\beta)k'}{h(t+1)} \in \mathbb{N}$ and then group the sets $U_1, \ldots, U_{\beta k'}$ into hr groups and the sets $U_{\beta k'+1}, \ldots, U_{k'}$ into $h(t+1)$ groups. Specifically, for each $(x,y) \in [r] \times [h]$, we create the set $U^1_{x,y} = U_{(xr+y-1)\frac{\beta k'}{hr}+1} \times \ldots \times U_{(xr+y)\frac{\beta k'}{hr}}$, and for each $(x,y) \in [t+1] \times [h]$, we create the set $U^2_{x,y} = U_{\beta k' + (x(t+1)+y-1)\frac{(1-\beta)k'}{h(t+1)}+1} \times \ldots \times U_{\beta k' + (x(t+1)+y)\frac{(1-\beta)k'}{h(t+1)}}$.

Vertices of G. Now for each a in A_1 we set $V_a := U^1_{a,1} \times \ldots \times U^1_{a,h}$ and for each a in A_2 we set $V_a := U^2_{a,1} \times \ldots \times U^2_{a,h}$. Finally, for each $b = ((a_1, j_1), \ldots, (a_h, j_h))$ in B, where for each ℓ we have $a_\ell \in A_{i_\ell}$, we set $V_b := E(H') \cap (U^{i_1}_{a_1,j_1} \times \ldots \times U^{i_h}_{a_h,j_h})$. This describes the k parts of the k-partite vertex set $V(G)$. We choose r large enough so that the maximum size of each part is $N_G := N_H^{(1-\beta)k'/(t+1)}$.

Edges of G Now we construct the edges and weights of the graph. Let us start with the edges. The construction here is basically the same as the construction of the edges in the unweighted proof in the last section. For each a in A_2 and $b = ((a_1, j_1), \ldots, (a_h, j_h))$ in B with (a,b) in $E(H)$, consider an arbitrary $u = (u_1, \ldots, u_h)$ in V_a and $u' = (u'_1, \ldots, u'_h)$ in V_b. We say that u and u' are "compatible" if for every ℓ with $a_\ell = a$ we have $u'_\ell = u_{j_\ell}$; in this case we connect u and u' by an edge. This finishes the construction of the edges of G.

Weights of G. Now we construct the weights. We want to encode the same edge constraints as we just encoded for A_2, but now for A_1, and we have to use weights instead of edges. To do this, we use $|B|$-average free sets via the construction of Lemma 14. We simplify the usage in this shortened proof to avoid dealing with too many variables. We use the lemma to obtain in polynomial time (which we will treat as negligible here) a $|B|$-average free set S of size $N_H^{\beta k'/(hr)}$ such that $S \subseteq [0, C]$, where $C \approx O(N_H^{\beta k'/(hr)})$ (up to a factor of $(1+\varepsilon)$ in the exponent, but we will ignore this here for simplicity). From this, we can construct an arbitrary bijection $\varrho_S : [N_H^{\beta k'/(hr)}] \to S$.

To simplify our construction, we specify a target weight T (instead of the default target zero). We can easily get rid of this again later by subtracting T from the weights of all vertices of some set of the partition. The binary representation of T consists of hr blocks of $\lceil 2|B|C \rceil$ bits, indexed by pairs $(i,j) \in [r] \times [h]$, each containing the binary representation of $|B|C$. The block (i,j) represents the group $U^1_{i,j}$. The size of the blocks is large enough to prevent overflow between the blocks. Note that the maximum weight W now satisfies $\log_2(W) = \Theta(hr \log_2(2|B|C))$ and hence $W \approx O(N_H^{\beta k'})$.

Let us now actually specify the weights of the vertices, beginning with the vertices in V_a for $a \in A_1$. For each $(i,j) \in [r] \times [h]$, we relabel the elements of each $U^1_{i,j}$ as $\{1, \dots, N_H^{\beta k'/(hr)}\}$. Now we define the weight of the vertex $V_a \ni u = (u_1, \dots, u_h)$ to have, for each $i \in [h]$, the value $|B|C - |N(a)| \cdot \varrho_S(u_i)$ in the block (a,i) of its binary representation. Now we move on to the vertices in V_b for $b = ((a_1, j_1), \dots, (a_h, j_h))$. We define the weight of the vertex $V_b \ni u' = (u'_1, \dots, u'_h)$ to have, for each $i \in [h]$ such that $a_i \in A_1$, the value $\varrho_S(u'_i)$ in the block (a_i, j_i) of its binary representation.

All blocks and vertices which have not been assigned a weight yet are assigned a value of zero. This concludes the construction of G.

Correctness. Note that any colored subgraph isomorphism of H in G chooses vertices v_a in V_a for all a in $A_1 \cup A_2$. This corresponds to choosing vertices u_i in U_i for each $i \in [k']$. Moreover, the edges of a k'-hyperclique are in one-to-one correspondence with the set B. We simply need to show that the choice of hyperclique vertices induced by the choice of vertices in $A_1 \cup A_2$ agrees with the choice of hyperclique edges induced by the choice of vertices in B. For the vertices in A_2, this is easily seen to be ensured by the edges. For the vertices in A_1, we need to prove that the weights encode the same constraint. This, however, is simply the definition of a $|B|$-average free set: Consider the block (i,j) (where $(i,j) \in [r] \times [h]$) in the binary representation of the total weight of the subgraph. Suppose that for $i \in A_1$ the vertex $V_i \ni u = (u_1, \dots, u_h)$ was selected, and that for each $B \ni b = ((a_1, j_1), \dots, (a_h, j_h))$ with $\exists \ell : (a_\ell, j_\ell) = (i,j)$ the vertex $V_b \ni u' = (u'_1, \dots, u'_h)$ was selected. Then by construction, the total value in the block (i,j) is the value $|B|C - |N(i)|\varrho_S(u_j)$ (where $N(i)$ is the neighbourhood of $i \in A_1$), plus the value $\varrho_S(u'_\ell)$ for all b as above. Note that the latter term has exactly $|N(i)|$ summands, hence in order for the value in the block to be equal to $|B|C$ as specified by the target weight, we must have that the value of $\varrho_S(u_j)$ is equal to the value of each of the $\varrho_S(u'_\ell)$ by the definition of $|B|$-average free sets. Since ϱ_S is a bijection, this ensures that the choice of hyperclique vertices in the sets appearing in the Cartesian product defining $U^1_{i,j}$ – i.e. $U_{(ir+j-1)\frac{betak'}{hr}+1}, \dots, U_{(ir+j)\frac{\beta k'}{hr}}$ – agree with the choice of hyperclique edges. This is true for all i,j and hence for each U_ℓ for $\ell \in [k']$.

The other direction is easy to see via a similar, simpler argument. This concludes the correctness proof.

Running Time. It can be seen that the running time of this reduction is $O(N_H^{2h})$, up to the running time of the algorithm for the construction of the $|B|$-average free set, which we will ignore here for sake of simplicity. Now suppose we can solve Exact Weight

Subgraph Isomorphism in time $O(n^{t+1-\varepsilon}W)$. We use the reduction above to convert a Hyperclique instance with $n_H = O(N_H)$ nodes to an Exact Weight Subgraph Isomorphism instance where $W \approx \Theta(N_H^{\beta k'})$ and $n = O(N_H^{(1-\beta)k'/(t+1)})$. Choosing β carefully, we get $W = \Theta(N_H^{\gamma})$; note that we are ignoring some intricacies in the choice of β that arise when you consider the running time of the algorithm that constructs the $|B|$-average free set – the details are available in the full version of this paper. Now via the algorithm for Exact Weight Subgraph Isomorphism, we can solve this instance and hence the original Hyperclique problem in time $O(N_H^{2h}+N_H^{((1-\beta)k'/(t+1))(t+1-\varepsilon)}\cdot N_H^{\beta k'}) = O(N_H^{k'-\varepsilon'}) = O(n_H^{k'-\varepsilon'})$. ◀

It is easy to see that the same proof also rules out algorithms running in time $O(n^{t+1}W^{1-\varepsilon})$. Similar as with the proof for the unweighted problem, basically the same techniques can be used to prove the other parts of Theorem 4.

3 Open Problems

In this paper we discussed many different variants of the Subgraph Isomorphism problem. For some of these variants we leave gaps, which gives rise to several open problems:

1. Can the algorithms for weighted trees be improved? We have shown that some improvements can be made for node-weighted trees (see Theorem 9), but are these optimal? What about edge-weighted trees?
2. Are there fast algorithms for unweighted Subgraph Isomorphism on graphs of bounded pathwidth that do not use rectangular matrix multiplication? Can the gap between exponent $\omega(p-1)$ and exponent p be closed? Similar questions apply to the weighted case; see Theorems 6 and 8.
3. Relatedly, are there good lower bounds for unweighted Subgraph Isomorphism on graphs of bounded pathwidth? Recall that Lemma 11 does not allow us to transfer our lower bounds for the colored case to the uncolored case.

We conclude with some more general open problems:

1. Do our algorithms and lower bounds also work for other types of graph homomorphisms, and for counting the number of solutions? Techniques from [24] seem applicable.
2. In this work we demonstrated the existence of maximally hard patterns for which Subgraph Isomorphism requires time $n^{\mathrm{tw}(H)+1-o(1)}$. Can we classify which (classes of) patterns are maximally hard?
3. Changing our focus from hard patterns to easy patterns, we can ask: do classes of patterns of unbounded treewidth exist for which Subgraph Isomorphism can be solved in time $n^{o(\mathrm{tw}(H))}$? Recall that a conditional lower bound rules out $n^{o(\mathrm{tw}(H)/\log \mathrm{tw}(H))}$ [40].

References

1 Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *Journal of the ACM (JACM)*, 64(4):1–20, 2017.

2 Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *SIAM Journal on Computing*, 47(6):2203–2236, 2018.

3 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based lower bounds for Subset Sum and Bicriteria Path. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 41–57. SIAM, 2019.

4 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Scheduling lower bounds via AND Subset Sum. *arXiv preprint*, 2020. `arXiv:2003.07113`.

5 Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-SUM conjecture. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2013.

6 Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *European Symposium on Algorithms*, pages 1–12. Springer, 2014.

7 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM Journal on Computing*, 47(3):1098–1122, 2018.

8 Noga Alon. Testing subgraphs in large graphs. *Random Structures & Algorithms*, 21(3-4):359–370, 2002.

9 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995.

10 Omid Amini, Fedor V Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. In *International Colloquium on Automata, Languages, and Programming*, pages 71–82. Springer, 2009.

11 Zhao An, Qilong Feng, Iyad Kanj, and Ge Xia. The complexity of tree partitioning. In *Workshop on Algorithms and Data Structures*, pages 37–48. Springer, 2017.

12 MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Saeed Seddighin, and Cliff Stein. Fast algorithms for knapsack via convolution and prediction. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1269–1282, 2018.

13 Felix A Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences of the United States of America*, 32(12):331, 1946.

14 Hans L Bodlaender. A tourist guide through treewidth. *Acta cybernetica*, 11(1-2):1, 1994.

15 Hans L Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1-2):1–45, 1998.

16 Hans L Bodlaender. Discovering treewidth. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 1–16. Springer, 2005.

17 David Bremner, Timothy M Chan, Erik D Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, and Perouz Taslakian. Necklaces, convolutions, and x+ y. In *European Symposium on Algorithms*, pages 160–171. Springer, 2006.

18 Karl Bringmann. A near-linear pseudopolynomial time algorithm for Subset Sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1073–1084. SIAM, 2017.

19 Danilo Bruschi, Lorenzo Martignoni, and Mattia Monga. Detecting self-mutating malware using control-flow graph matching. In *International conference on detection of intrusions and malware, and vulnerability assessment*, pages 129–143. Springer, 2006.

20 Timothy M Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 31–40, 2015.

21 Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM Journal on Computing*, 11(3):467–471, 1982.

22 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, 1987.

23 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

24 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 210–223, 2017.

25 Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: Hardness for all induced patterns and faster non-induced cycles. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1167–1178, 2019.

26 Bartłomiej Dudek, Paweł Gawrychowski, and Tatiana Starikovskaya. All non-trivial variants of 3-LDT are equivalent. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 974–981, 2020.

27 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, 2004.

28 Fedor V Fomin, Petr A Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM Journal on Computing*, 43(5):1541–1563, 2014.

29 Fedor V Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and BV Raghavendra Rao. Faster algorithms for finding and counting subgraphs. *Journal of Computer and System Sciences*, 78(3):698–706, 2012.

30 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018.

31 Edinah K Gnang, Ahmed Elgammal, and Vladimir Retakh. A spectral theory for tensors. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 20, pages 801–841, 2011.

32 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

33 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

34 Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for Subset Sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1062–1072. SIAM, 2017.

35 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

36 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303, 2014.

37 Yuan Li, Alexander Razborov, and Benjamin Rossman. On the AC^0 complexity of Subgraph Isomorphism. *SIAM Journal on Computing*, 46(3):936–971, 2017.

38 Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1236–1252. SIAM, 2018.

39 Josef Malík, Ondřej Suchỳ, and Tomáš Valla. Efficient implementation of Color Coding algorithm for Subgraph Isomorphism problem. In *International Symposium on Experimental Algorithms*, pages 283–299. Springer, 2019.

40 Dániel Marx. Can you beat treewidth? In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 169–179. IEEE, 2007.

41 Dániel Marx and Michal Pilipczuk. Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask). In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 542–553, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2014.542.

42 Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

43 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

44 Kevin Pratt. Waring rank, parameterized and exact algorithms. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 806–823. IEEE, 2019.

45 Gregory Rosenthal. Beating treewidth for average-case subgraph isomorphism. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

46 Paul D Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993.

47 Edward H Sussenguth. A graph-theoretic algorithm for matching chemical structures. *Journal of Chemical Documentation*, 5(1):36–43, 1965.

48 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.