Computation of Hadwiger Number and Related Contraction Problems: Tight Lower Bounds

Fedor V. Fomin
University of Bergen, Norway
fomin@ii.uib.no

Daniel Lokshtanov
University of California, Santa Barbara, CA, USA
daniello@ucsb.edu

Ivan Mihajlin
University of California, San Diego, CA, USA
imikhail@cs.ucsd.edu

Saket Saurabh
Department of Informatics, University of Bergen, Norway
The Institute of Mathematical Sciences, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University of the Negev, Beer-Sheva, Israel
meiravze@bgu.ac.il

Abstract

We prove that the Hadwiger number of an n-vertex graph G (the maximum size of a clique minor in G) cannot be computed in time $n^{o(n)}$, unless the Exponential Time Hypothesis (ETH) fails. This resolves a well-known open question in the area of exact exponential algorithms. The technique developed for resolving the Hadwiger number problem has a wider applicability. We use it to rule out the existence of $n^{o(n)}$-time algorithms (up to ETH) for a large class of computational problems concerning edge contractions in graphs.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Hadwiger Number, Exponential-Time Hypothesis, Exact Algorithms, Edge Contraction Problems

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.49

Category Track A: Algorithms, Complexity and Games

Related Version The full version of this paper is available at `http://arxiv.org/abs/2004.11621`.

Funding *Fedor V. Fomin*: Research Council of Norway via the project MULTIVAL.
Daniel Lokshtanov: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 715744), and United States - Israel Binational Science Foundation grant no. 2018302.
Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.
Meirav Zehavi: Israel Science Foundation grant no. 1176/18, and United States – Israel Binational Science Foundation grant no. 2018302.

1 Introduction

The *Hadwiger number* $h(G)$ of a graph G is the largest number h for which the complete graph K_h is a minor of G. Equivalently, $h(G)$ is the maximum size of the largest complete graph that can be obtained from G by contracting edges. It is named after Hugo Hadwiger, who conjectured in 1943 that the Hadwiger number of G is always at least as large as its chromatic number. According to Bollobás, Catlin, and Erdős, this conjecture remains "one of the deepest unsolved problems in graph theory" [4].

The Hadwiger number of an n-vertex graph G can be easily computed in time $n^{\mathcal{O}(n)}$ by brute-forcing through all possible partitions of the vertex set of G into connected sets, contracting each set into one vertex and checking whether the resulting graph is a complete graph. The question whether the Hadwiger number of a graph can be computed in single-exponential $2^{\mathcal{O}(n)}$ time was previously asked in [1, 6, 13]. Our main result provides a negative answer to this open question.

▶ **Theorem 1.** *Unless the Exponential Time Hypothesis (ETH) is false, there does not exist an algorithm computing the Hadwiger number of an n-vertex graph in time $n^{o(n)}$.*

The interest in the complexity of the Hadwiger number is naturally explained by the recent developments in the area of exact exponential algorithms, that is, algorithms solving intractable problems significantly faster than the trivial exhaustive search, though still in exponential time [8]. Within the last decade, significant progress on upper and lower bounds of exponential algorithms has been achieved. Drastic improvements over brute-force algorithms were obtained for a number of fundamental problems like Graph Coloring [3] and Hamiltonicity [2]. On the other hand, by making use of the ETH, lower bounds could be obtained for 2-CSP [15] or for Subgraph Isomorphism and Graph Homomorphism [6].

Graph Minor (deciding whether a graph G contains a graph H as a minor) is a fundamental problem in graph theory and graph algorithms. Graph Minor could be seen as special case of a general graph embedding problem where one wants to embed a graph H into graph G. In what follows we will use n to denote the number of vertices in G and h to denote the number of vertices in H. By the theorem of Robertson and Seymour [14], there exists a computable function f and an algorithm that, for given graphs G and H, checks in time $f(h) \cdot n^3$ whether H is a minor of G. Thus the problem is fixed-parameter tractable (FPT) being parameterized by H. On the other hand, Cygan et al. [6] proved that unless the ETH fails, this problem cannot be solved in time $n^{o(n)}$ even in the case when $|V(G)| = |V(H)|$. Other interesting embedding problems that are strongly related to Graph Minor include the following problems.

- Subgraph Isomorphism: Given two graphs G and H, decide whether G contains a subgraph isomorphic to H. This problem cannot be solved in time $n^{o(n)}$ when $|V(G)| = |V(H)|$, unless the ETH fails [6]. In the special case called Clique, when H is a clique, a brute-force algorithm checking for every vertex subset of G whether it is a clique of size h solves the problem in time $n^{\mathcal{O}(h)}$. The same algorithm also runs in single-exponential time $\mathcal{O}(2^n n^2)$. It is also known that Clique is W[1]-hard parameterized by h and cannot be solved in time $f(h) \cdot n^{o(h)}$ for any function f unless the ETH fails [7, 5].
- Graph Homomorphism: Given two graphs G and H, decide whether there exists a homomorphism from G to H. (A *homomorphism* $G \to H$ from an undirected graph G to an undirected graph H is a mapping from the vertex set of G to that of H such that the image of every edge of G is an edge of H.) This problem is trivially solvable in time $h^{\mathcal{O}(n)}$, and an algorithm of running time $h^{o(n)}$ for this problem would yield the failure of

the ETH [6]. However, for the special case of H being a clique, Graph Homomorphism is equivalent to h-Coloring (deciding whether the chromatic number of G is at most h), and thus is solvable in single-exponential time $2^n \cdot n^{\mathcal{O}(1)}$ [3, 12]. When the graph G is a complete graph, the problem is equivalent to finding a clique of size n in H, and then is solvable in time $2^h \cdot h^{\mathcal{O}(1)}$.

- Topological Graph Minor: Given two graphs G and H, decide whether G contains H as a topological minor. (We say that a graph H is a subdivision of a graph G if G can be obtained from H by contracting only edges incident with at least one vertex of degree two. A graph H is called a *topological minor* of a graph G if a subdivision of H is isomorphic to a subgraph of G.) This problem is, perhaps, the closest "relative" of Graph Minor. Grohe et al. [11] gave an algorithm of running time $f(h) \cdot n^3$ for this problem for some computable function f. Similar to Graph Minor and Subgraph Isomorphism, this problem cannot be solved in time $n^{o(n)}$ when $|V(G)| = |V(H)|$, unless the ETH fails [6]. However for the special case of the problem with H being a complete graph, Lingas and Wahlen [13] gave a single-exponential algorithm solving the problem in time $2^{\mathcal{O}(n)}$.

Thus all the above graph embedding "relatives" of Graph Minor are solvable in single-exponential time when graph H is a clique. However, from the perspective of exact exponential algorithms, Theorem 1 implies that finding the largest clique minor is the most difficult problem out of them all. This is why we find the lower bound provided by Theorem 1 surprising. Moreover, from the perspective of parameterized complexity, finding a clique minor of size h, which is FPT, is actually easier than finding a clique (as a subgraph) of size h, which is W[1]-hard, as well as from finding an h-coloring of a graph, which is para-NP-hard.

Theorem 1 also answers another question of Cygan et al. [6], who asked whether deciding if a graph H can be obtained from a graph G only by edge contractions, could be resolved in single-exponential time. By Theorem 1, the existence of such an algorithm is highly unlikely even when the graph H is a complete graph. Moreover, the technique developed to prove Theorem 1, appears to be extremely useful to rule out the existence of $n^{o(n)}$-time algorithms for various contraction problems. We formalize our results with the following $\mathcal{F}$-Contraction problem. Let $\mathcal{F}$ be a graph class. Given a graph G and $t \in \mathbb{N}$, the task is to decide whether there exists a subset $F \subseteq E(G)$ of size at most t such that $G/F \in \mathcal{F}$ (where G/F is the graph obtained from G by contracting the edges in F). We prove that in each of the cases of $\mathcal{F}$-Contraction where $\mathcal{F}$ is the family of chordal graphs, interval graphs, proper interval graphs, threshold graphs, trivially perfect graphs, split graphs, complete split graphs and perfect graphs, unless the ETH fails, $\mathcal{F}$-Contraction is not solvable in time $n^{o(n)}$. For lack of space, some of these results are relegated to the full version of this paper (see [9]).

Technical Details. A summary of the reductions presented in this paper is given in Fig. 1. To prove our lower bounds, we first revisit the proof of Cygan et al. [6] for the ETH-hardness of a problem called List Subgraph Isomorphism. Informally, in this problem we are given two graphs G and H on the same number of vertices, as well as a list of vertices in H for each vertex in G, and we need to find a copy of G in H so that each vertex u in G is mapped to a vertex v in H that belongs to its list (i.e. v belongs to the list of u). We prove that the instances produced by the reduction (after some modification) of [6] have a very useful property that we crucially exploit later. Specifically, we construct a proper coloring of G as well as a proper coloring of H, and show that every vertex v in H that belongs to the list of some vertex u is, in fact, of the same color as u.

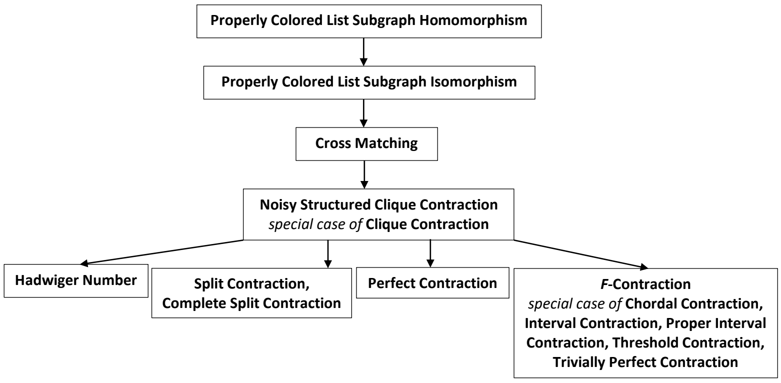

Figure 1 A summary of the problems considered in this paper, and the reductions between them.

Having proved the above, we turn to prove the ETH-hardness of a special case of Clique Contraction where the input graph is highly structured. To this end, we introduce an intermediate problem called Cross Matching. Informally, in this problem we are given a graph L with a partition (A, B) of its vertex set, and need to find a perfect matching between A and B whose contraction gives a clique. To see the connection between this problem and List Subgraph Isomorphism, think of the subgraph of L induced by one side of the partition – say, A – as a representation of the *complement* of G, and the subgraph of L induced by the other side of the partition as a representation of H. Then, the edges that go across A and B in a perfect matching can be thought of as a mapping of the vertices of G to the vertices of H. The crossing edges of L are easily defined such that necessarily a vertex of G can only be matched to a vertex in its list. In particular, we would like to enforce that every "non-edge" of the complement of G (which corresponds to an edge of G) would have to be mapped to an edge of H in order to obtain a clique. However, the troublesome part is that non-edges of the complement of G may also be "filled" (to eventually get a clique) using crossing edges rather than only edges of H. To argue that this critical issue does not arise, we crucially rely on the proper colorings of G and H.

Now, for the connection between Cross Matching and Clique Contraction, note that a solution to an instance of Cross Matching is clearly a solution to the instance of Clique Contraction defined by the same graph, but the other direction is not true. By adding certain vertices and edges to the graph of an instance of Cross Matching, we enforce all solutions to be perfect matchings between A and B. In particular, we construct the instances of Clique Contraction in a highly structured manner that allows us to derive not only the ETH-hardness of Clique Contraction itself, but to build upon them and further derive ETH-hardness for a wide variety of other contraction problems. In particular, we show that the addition of "noise" (that is, extra vertices and edges) to any structured instance of Clique Contraction has very limited effect. Roughly speaking, we show that the edges in the "noise" and the edges going across the "noise" and core of the graph (that is, the original vertices corresponding to the structured instance of Clique Contraction) are not "helpful" when trying to create a clique on the core (i.e. it is not helpful to try to use these edges in order to fill non-edges between vertices in the core). Depending on the contraction problem at hand, the noise is slightly different, but the proof technique stays the same – first showing that the core must yield a clique, and then using the argument above (in fact, in all cases but that of perfect graphs, we are able to invoke the argument as a black box) to show that the noise is, in a sense, irrelevant.

Preliminaries. As we only use standard notations, we present them only in the full version of this paper.

2 Lower Bound: Prop-Colored List Subgraph Isomorphism

In this section we build upon the work of Cygan et al. [6] and show a lower bound for a problem called Properly Colored List Subgraph Isomorphism (Prop-Col LSI). Intuitively, Prop-Col LSI is a variant of Spanning Subgraph Isomorphism where given two graphs G and H, we ask whether G is isomorphic to some spanning subgraph of H. The input to the variant consists also of proper colorings of G and H and an additional labeling of vertices in G by subsets of vertices in H of the same color, so that each vertex in G can be mapped only to vertices in H contained in its list. Formally, it is defined as follows.

Properly Colored List Subgraph Isomorphism (Prop-Col LSI)
Input: Graphs G and H with proper colorings $c_G : V(G) \to \{1,\dots,k\}$ and $c_H : V(H) \to \{1,\dots,k\}$ for some $k \in \mathbb{N}$, respectively, and a function $\ell : V(G) \to 2^{V(H)}$ such that for every $u \in V(G)$ and $v \in \ell(u)$, $c_G(u) = c_H(v)$.
Question: Does there exist a bijective function $\varphi : V(G) \to V(H)$ such that *(i)* for every $\{u,v\} \in E(G)$, $\{\varphi(u),\varphi(v)\} \in E(H)$, and *(ii)* for every $u \in V(G)$, $\varphi(u) \in \ell(u)$?

Notice that as the function φ above is bijective rather than only injective, we seek a spanning subgraph. Our objective is to prove the following statement.

▶ **Lemma 2.** *Unless the ETH is false, there does not exist an algorithm that solves Prop-Col LSI in time $n^{o(n)}$ where $n = |V(G)|$.*

In [6], the authors considered the two problems defined below. Intuitively, the second is defined as Prop-Col LSI when no proper colorings of H and G are given (and hence the labeling of vertices in G is not restricted accordingly); the first is defined as the second when we seek a homomorphism rather than an isomorphism (i.e., the sought function φ may not be injective) and also $|V(G)|$ may not be equal to $|V(H)|$ (thus φ may neither be onto).

List Subgraph Homomorphism (LSH)
Input: Graphs G and H, and a function $\ell : V(G) \to 2^{V(H)}$.
Question: Does there exist a function $\varphi : V(G) \to V(H)$ such that *(i)* for every $\{u,v\} \in E(G)$, $\{\varphi(u),\varphi(v)\} \in E(H)$, and *(ii)* for every $u \in V(G)$, $\varphi(u) \in \ell(u)$?

List Subgraph Isomorphism (LSI)
Input: Graphs G and H where $|V(G)| = |V(H)|$, and a function $\ell : V(G) \to 2^{V(H)}$.
Question: Does there exist a bijective function $\varphi : V(G) \to V(H)$ such that *(i)* for every $\{u,v\} \in E(G)$, $\{\varphi(u),\varphi(v)\} \in E(H)$, and *(ii)* for every $u \in V(G)$, $\varphi(u) \in \ell(u)$?

The proof of hardness of LSI consists of two parts:
- Showing ETH-hardness of LSH.
- Giving a fine-grained reduction from LSH to LSI.

We cannot use the hardness of LSI as a black box because Prop-Col LSI is a special case of LSI. Nevertheless, we will prove that the instances generated by the reduction (with a minor crucial modification) of Cygan et al. [6] have the additional properties required to make them instances of our special case.

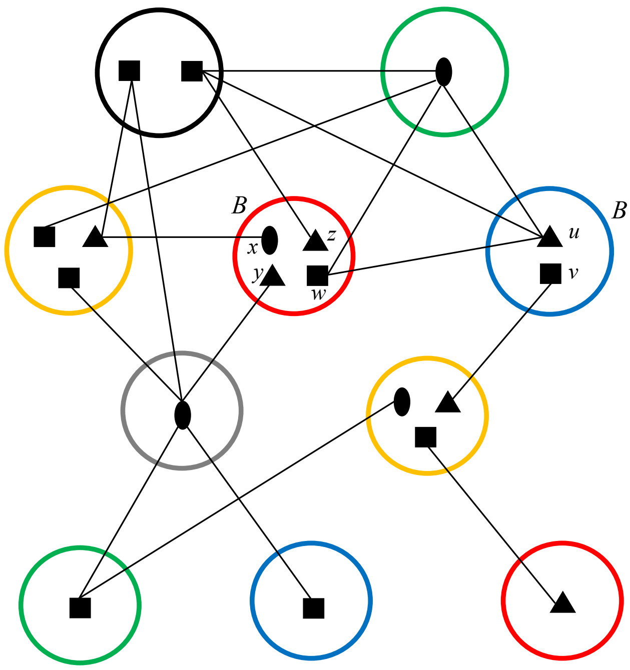

Figure 2 The reduction in Definition 4. The vertices of G are depicted by black shapes, where each distinct shape represents a different color (say, square is 1, rectangle is 2 and oval is 3), and the vertices of $\widetilde{G}$ are depicted by circles enclosing the vertex sets identifies with them, where the color of a vertex is the color of its circle (say, black is 1, green is 2, yellow is 3, red is 4, blue is 5 and grey is 6). Edges (of both graphs) are depicted by black lines. (The graph $\widetilde{H}$ is not shown). Then, the function ϕ_B is defined as follows: $\phi_B(1) = z, \phi_B(2) = \phi_B(5) = w, \phi_B(3) = x, \phi_B(4) = 0$, and $\phi_B(6) = y$. Moreover, the function $\phi_{B'}$ is defined as follows: $\phi_{B'}(1) = \phi_{B'}(2) = \phi_{B'}(4) = u, \phi_{B'}(3) = v$, and $\phi_{B'}(5) = \phi_{B'}(6) = 0$. With respect to B and B', the labeling ℓ is defined as follows: $\ell(B) = \{(R, 4) : R[1] \neq 0, R[2] = R[5] \neq 0, R[3] \neq 0, R[4] = 0, R[6] \neq 0\}$, and $\ell(B') = \{(R, 5) : R[1] = R[2] = R[4] \neq 0, R[3] \neq 0, R[5] = R[6] = 0\}$.

Lower Bound: Properly Colored Subgraph Homomorphism. Adapting the scheme of Cygan et al. [6] to our purpose, we will first show that finding a homomorphism remains hard if it has to preserve a given proper coloring:

> Properly Colored List Subgraph Homomorphism (Prop-Col LSH)
> **Input:** Graphs G and H with proper colorings $c_G : V(G) \to \{1, \ldots, k\}$ and $c_H : V(H) \to \{1, \ldots, k\}$ for some $k \in \mathbb{N}$, respectively, and a function $\ell : V(G) \to 2^{V(H)}$ such that for every $u \in V(G)$ and $v \in \ell(u)$, $c_G(u) = c_H(v)$.
> **Question:** Does there exist a function $\varphi : V(G) \to V(H)$ such that *(i)* for every $\{u, v\} \in E(G)$, $\{\varphi(u), \varphi(v)\} \in E(H)$, and *(ii)* for every $u \in V(G)$, $\varphi(u) \in \ell(u)$?

In [6], the authors gave a reduction from the 3-Coloring problem on n-vertex graphs of degree 4 (which is known not to be solvable in time $2^{o(n)}$ unless the ETH fails), which generates equivalent instances (G', H', ℓ) of LSH where both $|V(G')|$ and $|V(H')|$ are bounded by $\mathcal{O}(\frac{n}{\log n})$. This proves that LSH is not solvable in time $n^{o(n)}$ where $n = \max\{|V(G)|, |V(H)|\}$ unless the ETH fails. For their reduction, Cygan et al. [6] considered the notion of a *grouping* (also known as *quotient graph*) $\widetilde{G}$ of a graph G is a graph with vertex set $V(\widetilde{G}) = \{B_1, B_2, \ldots, B_t\}$ where $(B_1, B_2, \ldots, B_t)$ is a partition of $V(G)$ for some $t \in \mathbb{N}$ and for any distinct $i, j \in \{1, \ldots, t\}$, the vertices B_i and B_j are adjacent in $\widetilde{G}$ if and only if there exist $u \in B_i$ and $v \in B_j$ that are adjacent in G. Specifically, they computed a grouping with a coloring having specific properties as stated in the following lemma (see also Fig. 2.).

▶ **Lemma 3** (Lemma 3.2 in [6]). *For any constant $d \geq 1$, there exist positive integers $\lambda = \lambda(d)$, $n_0 = n_0(d)$ and a polynomial time algorithm that for a given graph G on $n \geq n_0$ vertices of maximum degree d and a positive integer $r \leq \sqrt{\frac{n}{2\lambda}}$, finds a grouping $\widetilde{G}$ of G and a coloring $\widetilde{c} : V(\widetilde{G}) \to [\lambda r]$ with the following properties:*

1. *$|V(\widetilde{G})| \leq |V(G)|/r$;*
2. *The coloring $\widetilde{c}$ is a proper coloring of $\widetilde{G}^2$;*[1]
3. *Each vertex of $\widetilde{G}$ is an independent set in G;*
4. *For any edge $\{B_i, B_j\} \in E(\widetilde{G})$, there exists exactly one pair $(u, v) \in B_i \times B_j$ such that $\{u, v\} \in E(G)$.*

Now, we describe the reduction of [6]. Here, without loss of generality, it is assumed that G has no isolated vertices, else they can be removed. An explanation of the intuition behind this somewhat technical definition is given below it.

▶ **Definition 4.** *For any instance G of 3-Coloring where G has degree d and a positive integer $r = o(\sqrt{|V(G)|})$, the instance* `reduce`$(G) = (\widetilde{G}, \widetilde{H}, \ell)$ *of LSH is defined as follows.*

- ***The graph $\widetilde{G}$.*** *Let $\widetilde{G}$ and $\widetilde{c} : V(\widetilde{G}) \to \{1, 2, \ldots, L\}$ be the grouping and coloring given by Lemma 3 where $L = \lambda(d)r$. Additionally, for each $B \in V(\widetilde{G})$, define $\phi_B : \{1, 2, \ldots, L\} \to B \cup \{0\}$ as follows: for any $i \in \{1, 2, \ldots, L\}$, if there exists (u, v, B') such that $u \in B$ and $v \in B'$, $\{u, v\} \in E(G)$ and $\widetilde{c}(B') = i$, then $\phi_B(i) = u$, and otherwise $\phi_B(i) = 0$.*[2]
- ***The graph $\widetilde{H}$.*** *Let $V(\widetilde{H}) = \{(R, l) : R \in \{0, 1, 2, 3\}^L, l \in L\}$,*[3] *and $E(\widetilde{H}) = \{\{(R, l), (R', l')\} : R[l'] \neq R'[l]\}$.*
- ***The labeling ℓ.*** *For any $B \in V(\widetilde{G})$, let $\ell(B)$ contain all vertices $(R, l) \in V(\widetilde{H})$ such that $\widetilde{c}(B) = l$, and there exists $f : B \to \{1, 2, 3\}$ such that for all $i \in \{1, 2, \ldots, L\}$, either $\phi_B(i) = R[i] = 0$ or both $\phi_B(i) \neq 0$ and $f(\phi_B(i)) = R[i]$.*

Intuitively, for every vertex $B \in V(\widetilde{G})$, the function ϕ_B can be interpreted as follows. It is the assignment, for every possible color $i \in \{1, \ldots, L\}$, of the unique vertex u within the vertex set identified with B itself that is adjacent to some vertex in the vertex subset identified with some vertex $B' \in V(\widetilde{G})$ colored i, if such a vertex u exists (else the assignment is of 0). In a sense, B thus stores the information on the identity of each vertex within it that is adjacent (in G) to some vertex outside of it, where each such internal vertex is uniquely accessed by specifying the color of the vertex in $\widetilde{G}$ whose identified vertex set contains the *neighbor*. With respect to the graph $\widetilde{H}$ and labeling ℓ, we interpret each vertex $(R, l) \in V(\widetilde{H})$ as a "placeholder" (i.e. potential assignment of the sought function φ) for any vertex $B \in V(\widetilde{G})$ that "complies with the pattern encoded by the pair (R, l)" as follows. First and straightforwardly, B must be colored l. Here, we remind that the colors of vertices in $\widetilde{G}$ belong to $\{1, \ldots, L\}$, while vertices in G are colored 1, 2 or 3 only. Then, the second requirement is that we can recolor (by f) the vertices in B so that the color of each vertex in B that is adjacent (in G) to some vertex outside B is as encoded by the vector R – that is, for each color $i \in \{1, \ldots, L\}$, if the vertex $\phi_B(i)$ is defined (i.e., $\phi_B(i) \neq 0$), then its color (which is $1, 2$ or 3) must be equal to the i-th entry of R. (More intuition is given in Fig. 2.)

Now, we state the correctness of the reduction.

[1] The square G^2 of a graph G is the graph on vertex set $V(G)$ and edge set $\{\{u, v\} : \{u, v\} \in E(G)$ or there exists $w \in V(G)$ with $\{u, w\}, \{v, w\} \in E(G)\}$.
[2] The uniqueness of u (if it exists), and thus the validity of ϕ_B, follows from Properties 2 and 4 in Lemma 3.
[3] That is, R is a vector with L entries where each entry is 0, 1, 2 or 3.

▶ **Lemma 5** (Lemma 3.3 in [6]). *For any instance G of 3-Coloring where G is an n-vertex graph of degree d, and a positive integer $r = o(\sqrt{|V(G)|})$, the instance* $\texttt{reduce}(G) = (\widetilde{G}, \widetilde{H}, \ell)$ *is computable in time polynomial in the sizes of $G, \widetilde{G}$ and $\widetilde{H}$, and has the following properties.*

- *G is a* Yes*-instance of 3-Coloring if and only if $(\widetilde{G}, \widetilde{H}, \ell)$ is a* Yes*-instance of LSH.*
- *$|V(\widetilde{G})| \leq n/r$, and $|V(\widetilde{H})| \leq \gamma(d)^r$ where γ is some computable function of d.*

We next prove that we can add colorings to the instance $\texttt{reduce}(G) = (\widetilde{G}, \widetilde{H}, \ell)$ of LSH in order to cast it as an instance of Prop-Col LSH while making a minor mandatory modification to the graph $\widetilde{H}$.

▶ **Lemma 6.** *Given an instance* $\texttt{reduce}(G) = (\widetilde{G}, \widetilde{H}, \ell)$ *of LSH, an equivalent instance $(\widetilde{G}, \widetilde{H}', c_{\widetilde{G}}, c_{\widetilde{H}'}, \ell)$ of Prop-Col LSH, where $\widetilde{H}'$ is a subgraph of $\widetilde{H}$, is computable in polynomial time.*

Proof. Define $c_{\widetilde{G}} = \widetilde{c}$ where $\widetilde{c}$ is the coloring of $\widetilde{G}$ in Definition 4. Additionally, let $\widetilde{H}'$ be the subgraph of $\widetilde{H}$ induced by the vertex set $\{(R, l) \in V(\widetilde{H})$: there exists $B \in V(\widetilde{G})$ such that $(R, l) \in \ell(B)\}$. Then, define $c_{\widetilde{H}'} : V(\widetilde{H}') \to \{1, 2, \ldots, L\}$ as follows: for any $(R, l) \in V(\widetilde{H}')$, define $c_{\widetilde{H}'}((R, l)) = l$. Notice that, by the definition of $V(\widetilde{H}')$, every set assigned by ℓ is subset of $V(\widetilde{H}')$.

First, we assert that $(\widetilde{G}, \widetilde{H}', c_{\widetilde{G}}, c_{\widetilde{H}'}, \ell)$ is an instance of Prop-Col LSH. To this end, we need to verify that the three following properties hold.

1. $c_{\widetilde{G}}$ is a proper coloring of $\widetilde{G}$.
2. $c_{\widetilde{H}'}$ is a proper coloring of $\widetilde{H}'$.
3. For every $B \in V(\widetilde{G})$ and $(R, l) \in \ell(B)$, it holds that $c_{\widetilde{G}}(B) = c_{\widetilde{H}'}((R, l))$.

By the definition of $c_{\widetilde{G}}$, it is a proper coloring of $\widetilde{G}^2$, which is a supergraph of $\widetilde{G}$. Thus, $c_{\widetilde{G}}$ is a proper coloring of $\widetilde{G}$.

Now, we argue that $c_{\widetilde{H}'}$ is a proper coloring of $\widetilde{H}'$. To this end, consider some edge $\{(R, l), (R', l')\} \in E(\widetilde{H}')$. We need to show that $c_{\widetilde{H}'}((R, l)) \neq c_{\widetilde{H}'}((R', l'))$. By the definition of $c_{\widetilde{H}'}$, we have that $c_{\widetilde{H}'}((R, l)) = l$ and $c_{\widetilde{H}'}((R', l')) = l'$, and therefore it suffices to show that $l \neq l'$. By the definition of $E(\widetilde{H})$ (which is a superset of $E(\widetilde{H}')$), we have that $R[l'] \neq R'[l]$. Thus, necessarily at least one among $R[l']$ and $R'[l]$ is not 0, and so we suppose w.l.o.g. that $R[l']$ is not 0. Furthermore, since $(R, l) \in V(\widetilde{H}')$, we have that there exists $B \in E(\widetilde{G})$ such that $(R, l) \in \ell(B)$. Thus,

- $\widetilde{c}(B) = l$.
- There exists $f : B \to \{1, 2, 3\}$ such that for all $i \in \{1, 2, \ldots, L\}$, either $\phi_B(i) = R[i] = 0$ or both $\phi_B(i) \neq 0$ and $f(\phi_B(i)) = R[i]$.

From the second property, and because $R[l'] \neq 0$, we necessarily have that both $\phi_B(l') \neq 0$ and $f(\phi_B(l')) = R[l']$. In particular, by the definition of ϕ_B, having $\phi_B(l') \neq 0$ means that there exists (u, v, B') such that $u \in B$, $v \in B'$, $\{u, v\} \in E(G)$ and $\widetilde{c}(B') = l'$. By the definition of $\widetilde{G}$ as a grouping of G, having $u \in B$, $v \in B'$ and $\{u, v\} \in E(G)$ implies that $\{B, B'\} \in E(\widetilde{G})$. Because $\widetilde{c}$ is a proper coloring of $\widetilde{G}$, this means that $\widetilde{c}(B) \neq \widetilde{c}(B')$. Since $\widetilde{c}(B) = l$ and $\widetilde{c}(B') = l'$, we derive that $l \neq l'$. Hence, $c_{\widetilde{H}'}$ is indeed a proper coloring of $\widetilde{H}'$.

To conclude that $(\widetilde{G}, \widetilde{H}', c_{\widetilde{G}}, c_{\widetilde{H}'}, \ell)$ is indeed an instance of Prop-Col LSH, it remains to assert that for every $B \in V(\widetilde{G})$ and $(R, l) \in \ell(B)$, it holds that $c_{\widetilde{G}}(B) = c_{\widetilde{H}'}((R, l))$. To this end, consider some $B \in V(\widetilde{G})$ and $(R, l) \in \ell(B)$. By the definition of ℓ (recall Definition 4), $(R, l) \in \ell(B)$ implies that $\widetilde{c}(B) = l$. As $c_{\widetilde{G}} = \widetilde{c}$, we have that $c_{\widetilde{G}}(B) = l$. Moreover, the definition of $c_{\widetilde{H}'}$ directly implies that $c_{\widetilde{H}'}((R, l)) = l$. Thus, $c_{\widetilde{G}}(B) = c_{\widetilde{H}'}((R, l))$.

Finally, we argue that $(\widetilde{G}, \widetilde{H}, \ell)$ is a Yes-instance of LSH if and only if $(\widetilde{G}, \widetilde{H}', c_{\widetilde{G}}, c_{\widetilde{H}'}, \ell)$ is a Yes-instance of Prop-Col LSH. In one direction, because $\widetilde{H}'$ is a subgraph of $\widetilde{H}$, it is immediate that if $(\widetilde{G}, \widetilde{H}', c_{\widetilde{G}}, c_{\widetilde{H}'}, \ell)$ is a Yes-instance of Prop-Col LSH, then so is $(\widetilde{G}, \widetilde{H}, \ell)$. For the other direction, suppose that $(\widetilde{G}, \widetilde{H}, \ell)$ is a Yes-instance of LSH. Thus, there exists a function $\varphi : V(\widetilde{G}) \to V(\widetilde{H})$ such that *(i)* for every $\{B, B'\} \in E(\widetilde{G})$, $\{\varphi(B), \varphi(B')\} \in E(\widetilde{H})$, and *(ii)* for every $B \in V(\widetilde{G})$, $\varphi(B) \in \ell(B)$. In particular, directly by the definition of $V(\widetilde{H}')$, the second condition implies that for every $B \in V(\widetilde{G})$, it holds that $\varphi(B) \in V(\widetilde{H}')$. Thus, because $\widetilde{H}'$ is an induced subgraph of $\widetilde{H}$, it holds that for every $\{B, B'\} \in E(\widetilde{G})$, $\{\varphi(B), \varphi(B')\} \in E(\widetilde{H}')$. Therefore, φ witnesses that $(\widetilde{G}, \widetilde{H}', c_{\widetilde{G}}, c_{\widetilde{H}'}, \ell)$ is a Yes-instance of Prop-Col LSH. ◀

We are now ready to assert the hardness of Prop-Col LSH. The proof, based on Lemmas 3, 5 and 6, can be found in the full version of this paper.

▶ **Lemma 7.** *Unless the ETH is false, there does not exist an algorithm that solves Prop-Col LSH in time $n^{o(n)}$ where $n = \max(|V(G)|, |V(H)|)$.*

From Graph Homomorphism to Subgraph Isomorphism. In this part, we observe that the reduction of [6] from LSH to LSI can be essentially used as is to serve as a reduction from Prop-Col LSH to Prop-Col LSI. For the sake of completeness, we give the full details (and the conclusion of the proof of Lemma 2) in the full version of this paper.

3 Lower Bound for the Cross Matching Problem

In this section, towards the proof of a lower bound for Clique Contraction, we prove a lower bound for an intermediate problem called Cross Matching that somewhat resembles Clique Contraction, and which is defined as follows.

Cross Matching
Input: A graph G with a partition (A, B) of $V(G)$ where $|A| = |B|$.
Question: Does there exist a perfect matching M in G such that every edge in M has one endpoint in A and the other in B, and G/M is a clique?

Our objective is to prove the following statement.

▶ **Lemma 8.** *Unless the ETH is false, there does not exist an algorithm that solves Cross Matching in time $n^{o(n)}$ where $n = |A|$.*

Proof. Towards a contradiction, suppose that there exists an algorithm, denoted by MatchingAlg, that solves Cross Matching in time $n^{o(n)}$ where n is the number of vertices in the set A in the input. We will show that this implies the existence of an algorithm, denoted by LSIAlg, that solves Prop-Col LSI in time $n^{o(n)}$ where n is the number of vertices in the input graph G, thereby contradicting Lemma 2 and hence completing the proof.

We define the execution of LSIAlg as follows. Given an instance (G, H, c_G, c_H, ℓ) of Prop-Col LSI, LSIAlg constructs an instance (L, A, B) of Cross Matching as follows (see Fig. 3):

- $V(L) = V(\overline{G}) \cup V(H)$.
- $E(L) = E(\overline{G}) \cup E(H) \cup \{\{u, v\} : u \in V(G), v \in L(u)\}$.
- $A = V(\overline{G})$ and $B = V(H)$.

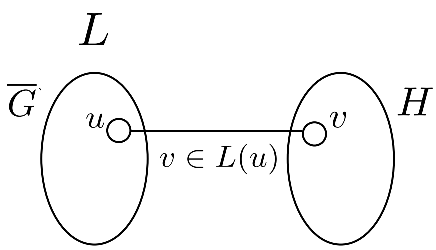

Figure 3 The construction of an instance of Cross Matching in the proof of Lemma 8.

Then, LSIAlg calls MatchingAlg with (L, A, B) as input, and returns the answer of this call.

Denote $n = |V(G)|$, and notice that $|A| = |B| = n$. Thus, because MatchingAlg runs in time $|A|^{o(|A|)} = n^{o(n)}$, so does LSIAlg.

For the correctness of the algorithm, first suppose that (G, H, c_G, c_H, ℓ) is a Yes-instance of Prop-Col LSI. This means that there exists a bijective function $\varphi : V(G) \to V(H)$ such that *(i)* for every $\{u, v\} \in E(G)$, $\{\varphi(u), \varphi(v)\} \in E(H)$, and *(ii)* for every $u \in V(G)$, $\varphi(u) \in L(u)$. Having φ at hand, we will show that (L, A, B) is a Yes-instance, which will imply that the call to MatchingAlg with (L, A, B) as input returns Yes, and hence LSIAlg returns Yes.

Based on φ, we define a subset $M \subseteq E(L)$ as follows: $M = \{\{u, \varphi(u)\} : u \in A\}$. Notice that the containment of M in $E(L)$ follows from the definition of $E(L)$ and Condition *(ii)* above. Moreover, by the definition of A, B and because φ is bijective, it further follows that M is a perfect matching in L such that every edge in M has one endpoint in A and the other in B. Thus, to conclude that (L, A, B) is a Yes-instance, it remains to argue that L/M is a clique. To this end, we consider two arbitrary vertices x and y of L/M, and prove that they are adjacent in L/M. Necessarily x is a vertex that replaced two vertices $u \in A$ and $u' \in B$ such that $\{u, u'\} \in M$, and y is a vertex that replaced two vertices $v \in A \setminus \{u\}$ and $v' \in B \setminus \{u'\}$ such that $\{v, v'\} \in M$. By the definition of contraction, to show that x and y are adjacent in L/M, it suffices to show that u and v are adjacent in L or u' and v' are adjacent in L (or both). To this end, suppose that u and v are not adjacent in L, else we are done. By the definition of $E(L)$, this means that $\{u, v\} \notin E(\overline{G})$ and hence $\{u, v\} \in E(G)$. By Condition *(i)* above, we derive that $\{\varphi(u), \varphi(v)\} \in E(H)$. By the definition of M, we know that $u' = \varphi(u)$ and $v' = \varphi(v)$, therefore $\{u', v'\} \in E(H)$. In turn, by the definition of $E(L)$, we get that $\{u', v'\} \in E(L)$. Thus, the proof of the forward direction is complete.

Now, suppose that LSIAlg returns Yes, which means that the call to MatchingAlg with (L, A, B) returns Yes. Thus, (L, A, B) is a Yes-instance, which means that there exists a perfect matching M in G such that every edge in M has one endpoint in A and the other in B, and G/M is a clique. We define a function $\varphi : A \to B$ as follows. For every $u \in V(G)$, let $\varphi(u) = v$ where v is the unique vertex in B such that $\{u, v\} \in M$; the existence and uniqueness of v follows from the supposition that M is a perfect matching such that every edge in M has one endpoint in A and the other in B. Furthermore, by the definition of A, B and the edges in $E(L)$ with one endpoint in A and the other in B, it directly follows that φ is a bijective mapping between $V(G)$ and $V(H)$ such that for every $u \in V(G)$, it holds that $\varphi(u) \in L(u)$. Thus, it remains to argue that for every edge $\{u, v\} \in E(G)$, it holds that $\{\varphi(u), \varphi(v)\} \in E(H)$. To this end, consider some arbitrary edge $\{u, v\} \in E(G)$, and denote $u' = \varphi(u)$ and $v' = \varphi(v)$. Because L/M is a clique and M is a matching that, by the definition of φ, necessarily contains both $\{u, u'\}$ and $\{v, v'\}$, we derive that at least one of the following four conditions must be satisfied: *(i)* $\{u, v\} \in E(L)$; *(ii)* $\{u', v'\} \in E(L)$; *(iii)*

$\{u, v'\} \in E(L)$; *(iv)* $\{v, u'\} \in E(L)$. Because $\{u, v\} \in E(G)$, we have that $\{u, v\} \notin E(\overline{G})$ and therefore $\{u, v\} \notin E(L)$. Thus, we are left with Conditions *(ii)*, *(iii)* and *(iv)*. Now, we will crucially rely on the proper colorings of G and H to rule out the satisfaction of Conditions *(iii)* and *(iv)*.

▷ Claim 9. For any two edges $\{x, x'\}, \{y, y'\} \in E(L)$ such that $\{x, y\} \in E(G)$ and $x', y' \in V(H)$, it holds that neither $\{x, y'\}$ nor $\{y, x'\}$ belongs to $E(L)$.

Proof of Claim 9. Because c_G is a proper coloring of G and $\{x, y\} \in E(G)$, it holds that $c_G(x) \neq x_G(y)$. Because $\{x, x'\}, \{y, y'\} \in E(L)$, $x, y \in V(G)$ and $x', y' \in V(H)$, and by the definition of $E(L)$, it holds that $x' \in L(x)$ and $y' \in L(y)$, and therefore $c_G(x) = c_H(x')$ and $c_G(y) = c_H(y')$. Thus, $c_G(x) \neq c_H(y')$ and $c_G(y) \neq c_H(x')$, implying that $y' \notin L(x)$ and $x' \notin L(y)$. In turn, by the definition of $E(L)$, this means that neither $\{x, y'\}$ nor $\{y, x'\}$ belongs to $E(L)$. This completes the proof of the claim. ◁

We now return to the proof of the lemma. By Claim 9, we are only left with Condition *(ii)*, that is, $\{u', v'\} \in E(L)$. However, by the definition of $E(L)$, this means that $\{u', v'\} \in E(H)$. As argued earlier, this completes the proof of the reverse direction. ◀

4 Lower Bounds: Clique Contraction and Hadwiger Number

In this section, we prove a lower bound for Clique Contraction and consequently for Hadwiger Number, defined as follows.

> Clique Contraction
> **Input:** A graph G and $t \in \mathbb{N}$.
> **Question:** Is there a subset $F \subseteq E(G)$ of size at most t such that G/F is a clique?

> Hadwiger Number
> **Input:** A graph G and $h \in \mathbb{N}$.
> **Question:** Is the Hadwiger number of G at least as large as h?

Our objective is to prove the following statement, where the analogous statement for Hadwiger Number (called Theorem 1 in the introduction) will follow as a corollary.

▶ **Theorem 10.** *Unless the ETH is false, there does not exist an algorithm that solves Clique Contraction in time $n^{o(n)}$ where $n = |V(G)|$.*

To make our approach adaptable to extract analogous statements for other contraction problems, we will first define a new problem called Noisy Structured Clique Contraction (which will arise in Section 5) along with a special case of it that is also a special case of Clique Contraction. Then, we will prove a crucial property of instances of Noisy Structured Clique Contraction, and afterwards we will use this property to prove Theorem 10 and its corollary. The definition of the new problem is as follows (see Fig. 4).

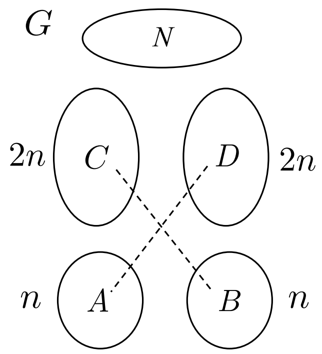

Figure 4 An instance of Noisy Structured Clique Contraction where dashed lines represent non-edges.

Noisy Structured Clique Contraction
Input: A graph G on at least $6n$ vertices for some $n \in \mathbb{N}$, and a partition (A, B, C, D, N) of $V(G)$ such that $|A| = |B| = n$, $|C| = |D| = 2n$, no vertex in A is adjacent to any vertex in D, and no vertex in B is adjacent to any vertex in C.
Question: Does there exist a subset $F \subseteq E(G)$ of size at most n such that $G[A \cup B \cup C \cup D \cup X]/F$ is a clique,[a] where $X = \{u \in N :$ there exists a vertex $v \in A \cup B \cup C \cup D$ such that u and v belong to the same connected component of $G[F]\}$?

[a] Note that F might contain edges outside $G[A \cup B \cup C \cup D \cup X]$. Then, we slightly abuse notation so that $G[A \cup B \cup C \cup D \cup X]/F$ refers to $G[A \cup B \cup C \cup D \cup X]/(F \cap E(G[A \cup B \cup C \cup D \cup X]))$.

Intuitively, the vertex set X consists of the noise (represented by N) that "interacts" with non-noise (represented by $V(G) \setminus N$) through contracted edges (in F), i.e. the vertices in N that lie together with at least one vertex in $V(G) \setminus N$ in a component that will be contracted and thereby replaced by a single vertex. We refer to the special case of Noisy Structured Clique Contraction where $N = \emptyset$ as Structured Clique Contraction. Note that Structured Clique Contraction is also a special case of Clique Contraction.

Solutions to instances of Noisy Structured Clique Contraction exhibit the following property, which will be crucial in the proof of Theorem 10 as well as results in Section 5.

▶ **Lemma 11.** *Let F be a solution to an instance (G, A, B, C, D, N, n) of Noisy Structured Clique Contraction. Then, F is a matching of size n in G such that each edge in F has one endpoint in A and the other in B.*

Proof. We first argue that every vertex in $A \cup B$ is incident to at least one edge in F. Targeting a contradiction, suppose that there exists a vertex $u \in A \cup B$ that is not incident to any edge in F. Because $|A \cup B \cup C \cup D| = 6n$, $|F| \leq n$ and $G[A \cup B \cup C \cup D \cup X]/F$ is a clique (where the last two properties follow from the supposition that F is a solution), it holds that $G[A \cup B \cup C \cup D \cup X]/F$ is a clique on at least $5n + |X|$ vertices. Hence, the degree of every vertex in $G[A \cup B \cup C \cup D \cup X]/F$, and in particular of u, should be $5n - 1 + |X|$ in $G[A \cup B \cup C \cup D \cup X]/F$. However, because no vertex in A is adjacent to any vertex in D and no vertex in B is adjacent to any vertex in C, the degree of any vertex in $A \cup B$, and in particular of u, is at most $|A \cup B| - 1 + |C \cup D|/2 + |X| = 4n - 1 + |X|$ in $G[A \cup B \cup C \cup D \cup X]$. Because u is not incident to any edge in F, its degree in $G[A \cup B \cup C \cup D \cup X]/F$ is at most its degree in $G[A \cup B \cup C \cup D \cup X]$. This is a contradiction, thus we get that indeed every vertex in $A \cup B$ is incident to at least one edge in F. From this, because $|F| \leq n$ and $|A \cup B| = 2n$, we derive that F is a perfect matching in $G[A \cup B]$.

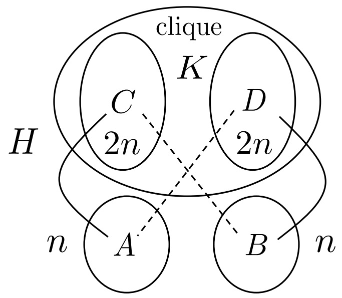

Figure 5 The construction of an instance of STRUCTURED CLIQUE CONTRACTION in the proof of Lemma 12 where dashed lines represent non-edges.

It remains to argue that every edge in F has one endpoint in A and the other in B. Targeting a contradiction, suppose that this is false. Because F is a perfect matching in $G[A \cup B]$, this means that there exist two vertices $a, a' \in A$ such that $\{a, a'\} \in F$. By the definition of NOISY STRUCTURED CLIQUE CONTRACTION, neither a nor a' is adjacent to any vertex in D. Moreover, note that $D \subseteq V(G[A \cup B \cup C \cup D \cup X]/F)$. In particular, the vertex of $G[A \cup B \cup C \cup D \cup X]/F$ yielded by the contraction of $\{a, a'\}$ is not adjacent to any vertex of D in $G[A \cup B \cup C \cup D \cup X]/F$. However, this is a contradiction because $G[A \cup B \cup C \cup D \cup X]/F$ is a clique. ◀

We now prove a lower bound for STRUCTURED CLIQUE CONTRACTION. Because it is a special case of CLIQUE CONTRACTION, this will directly yield the correctness of Theorem 10.

▶ **Lemma 12.** *Unless the ETH is false, there does not exist an algorithm that solves* STRUCTURED CLIQUE CONTRACTION *in time $n^{o(n)}$ where $n = |V(G)|$.*

Proof. Targeting a contradiction, suppose that there exists an algorithm, denoted by CliConAlg, that solves STRUCTURED CLIQUE CONTRACTION in time $n^{o(n)}$ where n is the number of vertices in the input graph. We will show that this implies the existence of an algorithm, denoted by MatchingAlg, that solves CROSS MATCHING in time $n^{o(n)}$ where n is the size of the set A in the input, thereby contradicting Lemma 8 and hence completing the proof.

We define the execution of MatchingAlg as follows. Given an instance (G, A, B) of CROSS MATCHING, MatchingAlg constructs an instance (H, A, B, C, D, n) of STRUCTURED CLIQUE CONTRACTION as follows (see Fig. 5):

- Let $n = |A|$, and K be a clique on $4n$ new vertices. Let (C, D) be a partition of $V(K)$ such that $|C| = |D|$.
- $V(H) = V(G) \cup V(K)$.
- $E(H) = E(G) \cup E(K) \cup \{\{a, c\} : a \in A, c \in C\} \cup \{\{b, d\} : b \in B, d \in D\}$.

Then, MatchingAlg calls CliConAlg with (H, A, B, C, D, n) as input, and returns the answer.

First, note that by construction, $|V(H)| = 6n$. Thus, because CliConAlg runs in time $|V(H)|^{o(|V(H)|)} \leq n^{o(n)}$, it follows that MatchingAlg runs in time $n^{o(n)}$.

For the correctness of the algorithm, first suppose that (G, A, B) is a Yes-instance of CROSS MATCHING. This means that there exists a perfect matching M in G such that every edge in M has one endpoint in A and the other in B, and G/M is a clique. By the definition of $E(H)$, $M \subseteq E(H)$. We will show that H/M is a clique. As $|M| = n$, this will mean that (H, A, B, C, D, n) is a Yes-instance of STRUCTURED CLIQUE CONTRACTION, which will mean, in turn, that the call to CliConAlg with (H, A, B, C, D, n) as input returns Yes, and hence MatchingAlg returns Yes.

Note that $V(H/M) = V(K) \cup V(G/M)$. To show that H/M is a clique, we consider two arbitrary vertices $u, v \in V(H/M)$, and show that they are adjacent in H/M. If $u, v \in V(K)$, then because K is a clique, it is clear that $\{u, v\} \in E(H/M)$. Moreover, if $u, v \in G/M$, then because G/M is a clique, it is clear that $\{u, v\} \in E(H/M)$. Thus, one of the vertices u and v belongs to $V(G/M)$ and the other belongs to $V(K)$. We suppose w.l.o.g. that $u \notin V(K)$. Because M is a perfect matching in G such that every edge in M has one endpoint in A and the other in B, it follows that u resulted from the contraction of the edge between some $a \in A$ and some $b \in B$. If $v \in C$, then $\{a, v\} \in E(H)$, and otherwise $v \in D$ and so $\{b, v\} \in E(H)$. Thus, by the definition of contraction, we conclude that $\{u, v\} \in E(H/M)$. This completes the proof of the forward direction.

Now, suppose that MatchingAlg returns Yes, which means that the call to CliConAlg with (H, A, B, C, D, n) returns Yes. Thus, (H, A, B, C, D, n) is a Yes-instance, which means that there exists a subset $F \subseteq E(H)$ of size at most n such that H/F is a clique. We will show that F is a perfect matching in G such that every edge in F has one endpoint in A and the other in B. Because H/F is a clique, this will imply that G/F is a clique and thus that (G, A, B) is a Yes-instance of Cross Matching. To achieve this, notice that by Lemma 11, F is a matching of size n in H such that each edge in F has one endpoint in A and the other in B. Because $G = H[A \cup B]$, we have that F is a perfect matching in G. Thus, the proof of the reverse direction is complete. ◀

▶ **Corollary 13.** *Unless the ETH is false, there does not exist an algorithm that solves Hadwiger Number in time $n^{o(n)}$ where $n = |V(G)|$.*

Proof. Targeting a contradiction, suppose that there exists an algorithm, denoted by HadwigerAlg, that solves Hadwiger Number in time $n^{o(n)}$ where n is the number of vertices in the input graph. We will show that this implies the existence of an algorithm, denoted by CliConAlg, that solves Clique Contraction in time $n^{o(n)}$ where n is the number of vertices in the input graph, thereby contradicting Theorem 10 and hence completing the proof.

We define the execution of CliConAlg as follows. Given an instance (G, t) of Clique Contraction, if G is not connected, then CliConAlg returns No, and otherwise it returns Yes if and only if HadwigerAlg returns Yes when called with $(G, |V(G)| - t)$ as input. Because the call to HadwigerAlg with input $(G, |V(G)| - t)$ runs in time $n^{o(n)}$ where $n = |V(G)|$, we have that CliConAlg runs in time $n^{o(n)}$ as well.

For the correctness of the algorithm, first observe that if G is not connected, then no sequence of edge contractions can yield a clique, and hence it is correct to return No. Thus, now assume that G is connected. First, suppose that (G, t) is a Yes-instance of Clique Contraction. This means that there exists a sequence of at most t edge contractions that transforms G into a clique. In particular, this clique must have at least $|V(G)| - t$ vertices, and therefore the Hadwiger number of G is at least as large as $|V(G)| - t$. By the correctness of HadwigerAlg, its call with $(G, |V(G)| - t)$ returns Yes, and therefore CliConAlg returns Yes.

Now, suppose that CliConAlg returns Yes, which means that the call to HadwigerAlg with $(G, |V(G)| - t)$ returns Yes. By the correctness of HadwigerAlg, the clique K_h for $h = |V(G)| - t$ is a minor of G. This means that there is a sequence of vertex deletions, edge deletions and edge contractions that transforms G into K_h. In particular, this sequence can contain at most t vertex deletions and edge contractions in total. Furthermore, by replacing each vertex deletion for a vertex v by an edge contraction for some edge e incident to v (which exists because G is connected) and dropping all edge deletions, we obtain another sequence that transforms G into K_h. Because this sequence contains only edge contractions, and at most t of them, we conclude that (G, t) is a Yes-instance of Clique Contraction. ◀

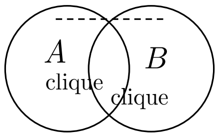

Figure 6 A two-cliques graph (see Definition 15).

5 Lower Bounds for Contraction to Graph Classes Problems

In this section, we prove lower bounds for several cases of the $\mathcal{F}$-Contraction problem, defined as follows. Here, $\mathcal{F}$ is a (possibly infinite) family of graphs.

$\mathcal{F}$-Contraction
Input: A graph G and $t \in \mathbb{N}$.
Question: Does there exist a subset $F \subseteq E(G)$ of size at most t such that $G/F \in \mathcal{F}$?

Notice that Clique Contraction is the case of $\mathcal{F}$-Contraction where $\mathcal{F}$ is the family of cliques. In this section, we consider the cases of $\mathcal{F}$-Contraction where $\mathcal{F}$ is the family of chordal graphs, interval graphs, proper interval graphs, threshold graphs, trivially perfect graphs, split graphs, complete split graphs and perfect graphs, also called Chordal Contraction, Interval Contraction, Proper Interval Contraction, Threshold Contraction, Trivially Perfect Contraction, Split Contraction, Complete Split Contraction and Perfect Contraction, respectively. Before we define these classes formally, it will be more enlightening to first define only the class of chordal graphs as well as somewhat artificial classes of graphs that will help us prove lower bounds for many of the classes above in a unified manner.

▶ **Definition 14** (Chordal Graphs). *A graph is* chordal *if it does not contain C_ℓ for all $\ell \geq 4$ as an induced subgraph.*

Our first class of graphs is defined as follows (see Fig. 6).

▶ **Definition 15** (Two-Cliques Graphs). *A* two-cliques graph *is a graph G such that there exist $A, B \subseteq V(G)$ such that $A \cup B = V(G)$, $G[A]$ and $G[B]$ are cliques, and there do not exist vertices $a \in A \setminus B$ and $b \in B \setminus A$ such that $\{a, b\} \in E(G)$. The* two-cliques class *is the class of all two-cliques graphs.*

It should be clear that the two-cliques class is a subclass of the class of chordal graphs. Now, we further define a family of classes of graphs as follows.

▶ **Definition 16** (Non-Trivial Chordal Class). *We say that a class of graphs $\mathcal{F}$ is* non-trivial chordal *if it is a subclass of the class of chordal graphs, and a superclass of the two-cliques class.*

Clearly, the class of cliques is not a non-trivial chordal class, and the class of chordal graphs is a non-trivial chordal class. The rest of this section is divided as follows. First, in Section 5, we prove a lower bound for any non-trivial chordal class. Then, in Section 5, we prove a lower bound for some graph classes that are not non-trivial chordal.

Non-Trivial Chordal Graph Classes. Here, our objective is to prove the following theorem. Afterwards, we will derive lower bounds for several known graph classes as corollaries.

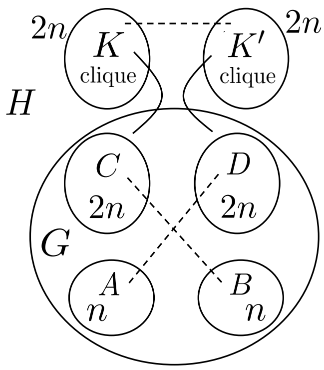

Figure 7 The construction of an instance of $\mathcal{F}$-Contraction in the proof of Theorem 17 where dashed lines represent non-edges.

▶ **Theorem 17.** *Let $\mathcal{F}$ be any non-trivial chordal graph class. Unless the ETH is false, there does not exist an algorithm that solves $\mathcal{F}$-Contraction in time $n^{o(n)}$ where $n = |V(G)|$.*

For the proof of this theorem, the following well-known property of chordal graphs will come in handy. This property is a direct consequence of the alternative characterization of the class of chordal graphs as the class of graphs that admit clique-tree decompositions, see [10].

▶ **Proposition 18.** *Let G be a chordal graph, and let u and v be two non-adjacent vertices in G. Then, $G[N(u) \cap N(v)]$ is a clique.*

We are now ready to prove Theorem 17.

Proof of Theorem 17. Targeting a contradiction, suppose that there exists an algorithm, denoted by NonTrivChordAlg, that solves $\mathcal{F}$-Contraction in time $n^{o(n)}$ where n is the number of vertices in the input graph. We will show that this implies the existence of an algorithm, denoted by CliConAlg, that solves Structured Clique Contraction in time $n^{o(n)}$ where n is the number of vertices in the input graph, thereby contradicting Lemma 12 and hence completing the proof.

We define the execution of CliConAlg as follows. Given an instance (G, A, B, C, D, n) of Structured Clique Contraction, CliConAlg constructs an instance (H, n) of $\mathcal{F}$-Contraction as follows (see Fig. 7):

- Let $n = |A|$. Moreover, let K and K' be two cliques, each on $2n$ new vertices.
- $V(H) = V(G) \cup V(K) \cup V(K')$.
- $E(H) = E(G) \cup E(K) \cup E(K') \cup \{\{u, v\} : u \in V(G), v \in V(K) \cup V(K')\}$.

Then, CliConAlg calls NonTrivChordAlg with (H, n) as input, and returns the answer of this call.

First, note that by construction, $|V(H)| = 10n$. Thus, because NonTrivChordAlg runs in time $|V(H)|^{o(|V(H)|)} \leq n^{o(n)}$, it follows that CliConAlg runs in time $n^{o(n)}$.

For the correctness of the algorithm, first suppose that (G, A, B, C, D, n) is a Yes-instance of Structured Clique Contraction. This means that there exists a subset $F \subseteq E(G)$ of size at most n such that G/F is a clique. By the definition of H, we directly derive that H/F is a two-cliques graphs, and therefore it belongs to $\mathcal{F}$. Thus, (H, n) is a Yes-instance of $\mathcal{F}$-Contraction, which means that the call to NonTrivChordAlg with (H, n) as input returns Yes, and hence CliConAlg returns Yes.

Now, suppose that CliConAlg returns Yes, which means that the call to NonTrivChordAlg with (H, n) returns Yes. Thus, (H, n) is a Yes-instance of $\mathcal{F}$-Contraction, which means that there exists a subset $F \subseteq E(H)$ of size at most n such that $H/F \in \mathcal{F}$. In particular, H/F

is a chordal graph. Based on Proposition 18, we will first show that $H[A\cup B\cup C\cup D\cup X]/F$ is a clique, where $X = \{u \in V(K)\cup V(K') :$ there exists a vertex $v \in A\cup B\cup C\cup D$ such that u and v belong to the same connected component of $H[F]\}$.

Targeting a contradiction, suppose that $H[A \cup B \cup C \cup D \cup X]/F$ is not a clique, and therefore there exist two non-adjacent vertices u and v in this graph. By the definition of X, $H[A \cup B \cup C \cup D \cup X]/F$ is equal to the subgraph of H/F induced by the set of vertices derived from connected components that contain at least one vertex from $A \cup B \cup C \cup D$. In particular, u and v are also non-adjacent vertices in H/F. By Proposition 18, this implies that $(H/F)[N_{H/F}(u) \cap N_{H/F}(v)]$ is a clique. Let $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) be the set of connected components of $H[F]$ that contain at least one vertex from $V(K_1)$ (resp. $V(K_2)$). Because $|F| \leq n$ and $|V(K_1)| = |V(K_2)| = 2n$, there exists at least one component $C_1 \in \mathcal{C}_1$ (resp. $C_2 \in \mathcal{C}_2$) that does not contain any vertex from $A \cup B \cup C \cup D$. Let c_1 and c_2 be the vertices of H/F yielded by the replacement of C_1 and C_2, respectively. As all vertices in $V(K_1) \cup V(K_2)$ are adjacent to all vertices in $A \cup B \cup C \cup D$, we have that $c_1, c_2 \in N_{H/F}(u) \cap N_{H/F}(v)$. However, there do not exist a vertex in $V(K_1)$ and a vertex in $V(K_2)$ that are adjacent in H, and for every vertex in $V(K_1) \cup V(K_2)$, its neighborhood outside this set is contained in $A \cup B \cup C \cup D$. Thus, c_1 and c_2 must be non-adjacent in H/F. However, this is a contradiction to the argument that $(H/F)[N_{H/F}(u) \cap N_{H/F}(v)]$ is a clique. From this, we derive that $H[A \cup B \cup C \cup D \cup X]/F$ is indeed a clique.

Now, notice that (H, A, B, C, D, N, n) where $N = V(K_1)\cup V(K_2)$ is an instance of Noisy Structured Clique Contraction. Furthermore, since $|F| \leq n$ and we have already shown that $H[A\cup B\cup C\cup D\cup X]/F$ is a clique, we have that F is a solution to this instance. Therefore, by Lemma 11, F is a matching of size n in H such that each edge in F has one endpoint in A and the other in B. In particular, $F \subseteq E(G)$ and hence $X = \emptyset$. Because $G = H[A \cup B \cup C \cup D]$, we thus derive that G/F is a clique. Thus, we conclude that (G, A, B, C, D, n) is a Yes-instance of Structured Clique Contraction. This completes the proof of the reverse direction. ◀

Now, we give definitions for several classes of graphs for which lower bounds will follow from Theorem 18. First, a graph is an *interval graph* if there exists a set of intervals on the real line such that the vertices of the graph are in bijection with these intervals, and there exists edge between two vertices if and only if their intervals intersect. A graph is a *proper interval graph* if, in the former definition, we also add the constraint that all intervals must have the same length. A graph is a *threshold graph* if it can be constructed from a one-vertex graph by repeated applications of the following two operations: addition of a single isolated vertex to the graph; addition of a single vertex that is connected to all other vertices. A graph is *trivially perfect* if in each of its induced subgraphs, the maximum size of an independent set equals the number of maximal cliques.

It is well-known that every graph that is a (proper) interval graph, or a threshold graph, or a trivially perfect graph, is also a chordal graph (see [10]). Moreover, it is immediate to verify that the two-cliques class is a subclass of the classes of (proper) interval graphs, threshold graphs and trivially perfect graphs. Thus, these classes are non-trivial chordal graphs classes, and therefore Theorem 17 directly implies lower bounds for them:

▶ **Corollary 19.** *Unless the ETH is false, none of the following problems admits an algorithm that solves it in time $n^{o(n)}$ where $n = |V(G)|$: Chordal Contraction, Interval Contraction, Proper Interval Contraction, Threshold Contraction and Trivially Perfect Contraction.*

Other Graph Classes. In Section 4, we proved a lower bound for a class of graphs that is not non-trivial chordal, namely, the class of cliques. In the full version of this paper, we show that our approach can yield lower bounds also for other classes of graphs that are not non-trivially chordal, including the classes of Split Graphs, Complete Split Graphs and Perfect Graphs.

References

1. Akanksha Agrawal, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Split contraction: The untold story. In *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
2. Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.
3. Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion–exclusion. *SIAM J. Computing*, 39(2):546–563, 2009.
4. B. Bollobás, P. A. Catlin, and P. Erdős. Hadwiger's conjecture is true for almost every graph. *European J. Combin.*, 1(3):195–199, 1980. doi:10.1016/S0195-6698(80)80001-1.
5. Jianer Chen, Benny Chor, Michael R. Fellows, Xiuzhen Huang, David Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.
6. Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socala. Tight lower bounds on graph embedding problems. *J. ACM*, 64(3):18:1–18:22, 2017. doi:10.1145/3051094.
7. Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer-Verlag, New York, 1999.
8. Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010. An EATCS Series: Texts in Theoretical Computer Science.
9. Fedor V. Fomin, Daniel Lokshtanov, Ivan Mihajlin, Saket Saurabh, and Meirav Zehavi. Computation of hadwiger number and related contraction problems: Tight lower bounds. *CoRR*, abs/2004.11621, 2020. arXiv:2004.11621.
10. Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2004.
11. Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 479–488, 2011.
12. Eugene L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5(3):66–67, 1976.
13. Andrzej Lingas and Martin Wahlen. An exact algorithm for subgraph homeomorphism. *J. Discrete Algorithms*, 7(4):464–468, 2009. doi:10.1016/j.jda.2008.10.003.
14. Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combinatorial Theory Ser. B*, 63(1):65–110, 1995.
15. Patrick Traxler. The time complexity of constraint satisfaction. In *Parameterized and Exact Computation*, pages 190–201. Springer, 2008.