Clique Is Hard on Average for Regular Resolution*

ALBERT ATSERIAS, Universitat Politècnica de Catalunya, Spain
ILARIO BONACINA, Universitat Politècnica de Catalunya, Spain
SUSANNA F. DE REZENDE, Institute of Mathematics of the Czech Academy of Sciences, Czech Republic
MASSIMO LAURIA, Sapienza - Università di Roma, Italy
JAKOB NORDSTRÖM, University of Copenhagen, Denmark and Lund University, Sweden
ALEXANDER RAZBOROV, University of Chicago, USA and Steklov Mathematical Institute, Russia

We prove that for $k \ll \sqrt[4]{n}$ regular resolution requires length $n^{\Omega(k)}$ to establish that an Erdős–Rényi graph with appropriately chosen edge density does not contain a k-clique. This lower bound is optimal up to the multiplicative constant in the exponent, and also implies unconditional $n^{\Omega(k)}$ lower bounds on running time for several state-of-the-art algorithms for finding maximum cliques in graphs.

CCS Concepts: • **Theory of computation → Proof complexity**; • **Mathematics of computing → Random graphs**.

Additional Key Words and Phrases: resolution, clique, average complexity

ACM Reference Format:
Albert Atserias, Ilario Bonacina, Susanna F. de Rezende, Massimo Lauria, Jakob Nordström, and Alexander Razborov. 2021. Clique Is Hard on Average for Regular Resolution. *J. ACM* 1, 1, Article 1 (January 2021), 25 pages. https://doi.org/10.1145/3449352

1 INTRODUCTION

Deciding whether a graph has a k-clique is one of the most basic computational problems on graphs, and has been extensively studied in computational complexity theory ever since it appeared in Karp's list of 21 NP-complete problems [26]. Not only is this problem widely believed to be intractable to solve exactly (unless P = NP), there does not even exist any polynomial-time algorithm for approximating the maximum size of a clique to within a factor $n^{1-\epsilon}$ for any constant $\epsilon > 0$, where n is the number of vertices in the graph [24, 62]. Furthermore, the problem appears to be hard not only in the worst case but also on average in the Erdős-Rényi random graph model—we know of no efficient algorithms for finding cliques of maximum size asymptotically almost surely on random graphs with appropriate edge densities [27, 48].

*A Preliminary version of this paper appeared in the proceedings of STOC'18 [2].

Authors' addresses: Albert Atserias, atserias@cs.upc.edu, Universitat Politècnica de Catalunya, Dept. Ciències de la Computació, Barcelona, Spain; Ilario Bonacina, bonacina@cs.upc.edu, Universitat Politècnica de Catalunya, Dept. Ciències de la Computació, Barcelona, Spain; Susanna F. de Rezende, rezende@math.cas.cz, Institute of Mathematics of the Czech Academy of Sciences, Department of Mathematical Logic and Theoretical Computer Science, Prague, Czech Republic; Massimo Lauria, massimo.lauria@uniroma1.it, Sapienza - Università di Roma, Department of Statistical Sciences, Rome, Italy; Jakob Nordström, jakobn@kth.se, University of Copenhagen, Department of Computer Science, Copenhagen, Denmark , Lund University, Department of Computer Science, Lund, Sweden; Alexander Razborov, razborov@math.uchicago.edu, University of Chicago, USA , Steklov Mathematical Institute, Moscow, Russia.

In terms of upper bounds, the k-clique problem can clearly be solved in time roughly n^k simply by checking if any of the $\binom{n}{k}$ many sets of vertices of size k forms a clique. This takes polynomial time if k is constant. This can be improved slightly to $O(n^{\omega k/3})$, where $\omega \leq 2.373$ is the matrix multiplication exponent, using algebraic techniques [39], although in practice such algebraic algorithms are outperformed by combinatorial ones [60].

The motivating problem behind this work is to determine the exact time complexity of the clique problem when k is given as a parameter. As noted above, all known algorithms require time $n^{\Omega(k)}$. It appears quite likely that some dependence on k is needed in the exponent, since otherwise we have the parameterized complexity collapse $\mathsf{FPT} = \mathsf{W}[1]$ [21]. Even more can be said if we are willing to believe the Exponential Time Hypothesis (ETH) [25]—then the exponent has to depend linearly on k [15], so that the trivial upper bound is essentially tight.

Obtaining such a lower bound unconditionally would, in particular, imply $\mathsf{P} \neq \mathsf{NP}$, and so currently seems completely out of reach. But is it possible to prove $n^{\Omega(k)}$ lower bounds in restricted but nontrivial models of computation? For circuit complexity, this challenge has been met for circuits that are of bounded depth [47] or are monotone [14, 49]. In this paper we focus on computational models that are powerful enough to capture several algorithms that are used in practice.

When analysing such algorithms, it is convenient to view the execution trace as a proof establishing the maximum clique size for the input graph. In particular, if this graph does not have a k-clique, then the trace provides an efficiently verifiable proof of the statement that the graph is k-clique-free. If one can establish a lower bound on the length of such proofs, then this implies a lower bound on the running time of the algorithm, and this lower bound holds even if the algorithm is a non-deterministic heuristic that somehow magically gets to make all the right choices. This brings us to the topic of *proof complexity* [16], which can be viewed as the study of upper and lower bounds in restricted nondeterministic computational models.

Using a standard reduction from k-clique to SAT, we can translate the problem of k-cliques in graphs to that of satisfiability of formulas in conjunctive normal form (CNF). If an algorithm for finding k-cliques is run on a graph G that is k-clique-free, then we can extract a proof of the unsatisfiability of the corresponding CNF formula—the k-clique formula on G—from the execution trace of the algorithm. Is it possible to show any non-trivial lower bound on the length of such proofs? Specifically, does the *resolution* proof system—the method of reasoning underlying state-of-the-art SAT solvers [3, 36, 38]—require length $n^{\Omega(k)}$, or at least $n^{\omega_k(1)}$ (i.e. the exponent as a function of k is not bounded by a constant), to prove the absence of k-cliques in a graph? This question was asked in, e.g., [9] and remains open.

The hardness of k-clique formulas for resolution is also a problem of intrinsic interest in proof complexity, since these formulas escape known methods of proving resolution lower bounds for a range of interesting values of k including $k = O(1)$. In particular, the interpolation technique [30, 45], the random restriction method [6], and the size-width lower bound [7] all seem to fail.

To make this more precise, we should mention that some previous works do use the size-width method, but only for very large k. It was shown in [5] that for $n^{5/6} \ll k \leq n/3$ resolution requires length $\exp\left(n^{\Omega(1)}\right)$ to certify that a dense enough Erdős-Rényi random graph is k-clique-free. The constant hidden in the $\Omega(1)$ increases with the density of the graph and, in particular, for very dense graphs and $k = n/3$ the length required is $2^{\Omega(n)}$. Also, for a specially tailored CNF encoding, where the ith member of the claimed k-clique is encoded in binary by $\log n$ variables, a lower bound of $n^{\Omega(k)}$ for $k \leq \log n$ can be extracted from a careful reading of [34]. However, in the more natural unary encodings, where indicator variables specify whether a vertex is in the clique, the size-width method cannot yield more than a $2^{\Omega(k^2/n)}$ lower bound since there are resolution proofs of width $O(k)$. This bound becomes trivial when $k \leq \sqrt{n}$.

In the restricted subsystem of *tree-like resolution*, optimal $n^{\Omega(k)}$ length lower bounds were established in [8] for k-clique formulas on complete $(k-1)$-partite as well as on average for Erdős-Rényi random graphs of appropriate edge density.

There is no hope to get hard instances for general resolution from complete $(k-1)$-partite graphs, however—in the same paper it was shown that all instances from the more general class of $(k-1)$-colourable graphs are easy for resolution. A closer study of these resolution proofs reveals that they are *regular*, meaning that if the proof is viewed as a directed acyclic graph (DAG), then no variable is eliminated more than once on any source-to-sink path.

More generally, regular resolution is an interesting and non-trivial model to analyse for the k-clique problem since it captures the reasoning used in many state-of-the-art algorithms used in practice (for a survey, see, e.g., [37, 44]). Nonetheless, it has remained consistent with state-of-the-art knowledge that for $k \leq n^{5/6}$ regular resolution might be able to certify k-clique-freeness in polynomial length independent of the value of k.

Our contributions. We prove optimal $n^{\Omega(k)}$ average-case lower bounds for regular resolution proofs of unsatisfiability for k-clique formulas on Erdős-Rényi random graphs.

Theorem 1.1 (Informal). *For any integer $k \ll \sqrt[4]{n}$, given an n-vertex graph G sampled at random from the Erdős-Rényi model with the appropriate edge density, regular resolution asymptotically almost surely requires length $n^{\Omega(k)}$ to certify that G does not contain a k-clique.*

At a high level, the proof is based on a bottleneck counting argument in the style of [23] with a slight twist that was introduced in [46]. In its classical form, such a proof takes four steps. First, one defines a distribution of random source-to-sink paths on the DAG representation of the proof. Second, a subset of the vertices of the DAG is identified—the set of *bottleneck nodes*—such that any random path must necessarily pass through at least one such node. Third, for any fixed bottleneck node, one shows that it is very unlikely that a random path passes through this particular node. Given this, a final union bound argument yields the conclusion that the DAG must have many bottleneck nodes, and so the resolution proof must be long.

The twist in our argument is that, instead of single bottleneck nodes, we need to define *bottleneck pairs* of nodes. We then argue that any random path passes through at least one such pair but that few random paths pass through any fixed pair; the latter part is based on Markov chain-type reasoning similar to [46, Theorems 3.2, 3.5]. Furthermore, it crucially relies on the graph satisfying a certain combinatorial property, which captures the idea that the common neighbourhood of a small set of vertices is well distributed across the graph. Identifying this combinatorial property is a key contribution of our work. In a separate argument (that, surprisingly, turned out to be much more elaborate than most arguments of this kind) we then establish that Erdős-Rényi random graphs of the appropriate edge density satisfy this property asymptotically almost surely. Combining these two facts yields our average-case lower bound.

The idea of counting bottlenecks of more than one node comes from [46] and was also used in [4].

Another contribution of this paper is a relatively simple observation that not only is regular resolution powerful enough to distinguish graphs that contain k-cliques from $(k-1)$-colourable graphs [8], but it can also distinguish them from graphs that have a homomorphism to any fixed graph H with no k-cliques.

Recent Developments. A preliminary version of this work appeared in the proceedings of STOC'18 [2]. The techniques used there to prove the $n^{\Omega(k)}$ average-case lower bound for regular resolution were recently extended by Pang [42] to work for a proof system between regular and general resolution. In the same paper, Pang also shows a $2^{\Omega(k^{(1-\epsilon)})}$ resolution lower bound for k-clique formulas on Erdős-Rényi random graphs, for $k = n^c$, $c < 1/3$ and $\epsilon > 0$.

Regarding the proof complexity of k-clique formulas for tree-like resolution, the lower bounds from [8] and [34] were simplified and unified in [33]. The resolution lower bound in [34] for k-clique formulas on Erdős-Rényi random graphs

under the binary encoding was recently extended to an $n^{\Omega(k)/d(s)}$ lower bound for $\mathrm{Res}(s)$, where $s = o((\log\log n)^{1/3})$ and $d(s)$ is a doubly exponential function [17].

Paper outline. The rest of this paper is organized as follows. Section 2 presents some preliminaries. We show that some nontrivial k-clique instances are easy for regular resolution in Section 3. Section 4 contains the formal statement of the lower bounds we prove for Erdős-Rényi random graphs. In Section 5 we define a combinatorial property of graphs and show that clique formulas on such graphs are hard for regular resolution, and the proof that Erdős-Rényi random graphs satisfy this property asymptotically almost surely is in Section 6. Section 7 explains why our results imply lower bounds on the running time of state-of-the-art algorithms for k-clique. We conclude in Section 8 with a discussion of open problems.

2 PRELIMINARIES

We write $G = (V, E)$ to denote a graph with vertices V and edges E, where G is always undirected, without loops and multiple edges. Given a vertex $v \in V$, we write $N(v)$ to denote the set of *neighbours of* v. For a set of vertices $R \subseteq V$ we write $\widehat{N}(R) = \bigcap_{v\in R} N(v)$ to denote the set of *common neighbours of* R. For two sets of vertices $R \subseteq V$ and $W \subseteq V$ we write $\widehat{N}_W(R) = \widehat{N}(R) \cap W$ to denote the set of *common neighbours of* R *inside* W. For a set $U \subseteq V$ we denote by $G[U]$ the subgraph of G induced by the set U. For $n \in \mathbb{N}^+$ we write $[n] = \{1, \ldots, n\}$. We say that $V_1 \dot\cup V_2 \dot\cup \cdots \dot\cup V_k = V$ is a *balanced* k-partition of V if for all $i, j \in [k]$ it holds that $|V_i| \leq |V_j| + 1$. All logarithms are natural (base e) if not specified otherwise.

Probability and Erdős-Rényi random graphs. We often denote random variables in boldface and write $\boldsymbol{X} \sim \mathscr{D}$ to denote that $\boldsymbol{X}$ is sampled from the distribution $\mathscr{D}$. A *p-biased coin*, or a *Bernoulli variable*, is the outcome of a coin flip that yields 1 with probability p and 0 with probability $1 - p$. We use the special case of Markov's inequality saying that if $\boldsymbol{X}$ is non-negative, then $\Pr[\boldsymbol{X} \geq 1] \leq \mathbb{E}[\boldsymbol{X}]$. We also need the following special case of the multiplicative Chernoff bound: if $\boldsymbol{X}$ is a binomial random variable (i.e., the sum of i.i.d. Bernoulli variables) with expectation $\mu = \mathbb{E}[\boldsymbol{X}]$, then $\Pr[\boldsymbol{X} \leq \mu/2] \leq \mathrm{e}^{-\mu/8}$.

We consider the Erdős-Rényi distribution $\mathscr{G}(n, p)$ of random graphs on a fixed set V of n vertices. A random graph sampled from $\mathscr{G}(n, p)$ is produced by placing each potential edge $\{u, v\}$ independently with probability p, $0 \leq p \leq 1$ (the edge probability p may be a function of n). A property of graphs is said to hold *asymptotically almost surely* on $\mathscr{G}(n, p(n))$ if it holds with probability that approaches 1 as n approaches infinity.

For a positive integer k, let $\boldsymbol{X}_k$ be the random variable that counts the number of k-cliques in a random graph from $\mathscr{G}(n, p)$. It follows from Markov's inequality that asymptotically almost surely there are no k-cliques in $\mathscr{G}(n, p)$ whenever p and k are such that $\mathbb{E}[\boldsymbol{X}_k] = p^{\binom{k}{2}}\binom{n}{k}$ approaches 0 as n approaches infinity. This is the case, for example, if $p = n^{-2\eta/(k-1)}$ for $k \geq 2$ and $\eta > 1$. Actually, the clique number, i.e. the size of the largest clique, $\omega(G)$ for a graph G sampled from $\mathscr{G}(n, p)$ is a well studied quantity and very strong concentrations bounds are known for it. For instance, one of the first concentration results is that $\omega(G) = (2 - o(1)) \log_{\frac{1}{p}}(n)$ with probability 1 as $n \to \infty$ (see for instance [10]).

CNF formulas and resolution. A *literal* over a Boolean variable x is either the variable x itself (a *positive literal*) or its negation $\neg x$ (a *negative literal*). A *clause* $C = \ell_1 \vee \cdots \vee \ell_w$ is a disjunction of literals; we say that the *width* of C is w. The empty clause will be denoted by $\bot$. A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We think of clauses as sets of literals and of CNF formulas as sets of clauses, so that order is irrelevant and there are no repetitions. For a formula F we denote by $\mathit{Vars}(F)$ the set of variables of F.

A *resolution derivation* from a CNF formula F is as an ordered sequence of clauses $\pi = (D_1, \ldots, D_L)$ such that for each $i \in [L]$ either D_i is a clause in F or there exist $j < i$ and $k < i$ such that D_i is derived from D_j and D_k by the *resolution rule*

$$\frac{B \vee x \qquad C \vee \neg x}{B \vee C}\,, \tag{2.1}$$

$D_i = B \vee C,\ D_j = B \vee x,\ D_k = C \vee \neg x$. We refer to $B \vee C$ as the *resolvent* of $B \vee x$ and $C \vee \neg x$ over x, and to x as the *resolved variable*. The *length* (or *size*) of a resolution derivation $\pi = (D_1, \ldots, D_L)$ is L and it is denoted by $|\pi|$. A *resolution refutation* of F, or *resolution proof* for (the unsatisfiability of) F, is a resolution derivation from F that ends in the empty clause $\perp$.

A resolution derivation $\pi = (D_1, \ldots, D_L)$ can also be viewed as a labelled DAG with the set of nodes $\{1, \ldots, L\}$ and edges (j, i), (k, i) for each application of the resolution rule deriving D_i from D_j and D_k. Each node i in this DAG is labelled by its associated clause D_i, and each non-source node is also labelled by the resolved variable in its associated derivation step in the refutation. A resolution refutation is called *regular* if along any source-to-sink path in its associated DAG every variable is resolved at most once.

For a partial assignment ρ we say that a clause C *restricted by* ρ, denoted $C\restriction_\rho$, is the trivial 1-clause if any of the literals in C is satisfied by ρ or otherwise is C with all falsified literals removed. We extend this definition to CNFs in the obvious way: $(C_1 \wedge \cdots \wedge C_m)\restriction_\rho = C_1\restriction_\rho \wedge \cdots \wedge C_m\restriction_\rho$. Applying a restriction preserves (regular) resolution derivations. To see this, observe that in every application of the resolution rule, the restricted consequence either becomes identically 1, or it is obtained, as before, by resolving the two restricted premises, or it is a weakening of one of them, but weakenings can be removed at no cost. Thus, we have:

Fact 2.1. *Let π be a (regular) resolution refutation of a CNF formula F. For any partial assignment ρ to the variables of F there is an efficiently constructible (regular) resolution refutation $\pi\restriction_\rho$ of the CNF formula $F\restriction_\rho$, so that the length of $\pi\restriction_\rho$ is at most the length of π.*

Branching programs. A branching program on variables $x_1, \ldots, x_n$ is a DAG that has one source node and where every non-sink node is labelled by one of the variables $x_1, \ldots, x_n$ and has exactly two outgoing edges labelled 0 and 1. The size of a branching program is the total number of nodes in the graph. In a *read-once branching program* it holds in addition that along every path every variable appears as a node label at most once.

For each node a in a branching program, let $X(a)$ denote the variable that labels a, and let a^0 and a^1 be the nodes that are reached from a through the edges labelled 0 and 1, respectively. A truth-value assignment $\sigma : \{x_1, \ldots, x_n\} \to \{0, 1\}$ determines a path in a branching program in the following way. The path starts at the source node. At an internal node a, the path is extended along the edge labelled $\sigma(X(a))$ so that the next node in the path is $a^{\sigma(X(a))}$. The path ends when it reaches a sink. We write $\mathrm{path}(\sigma)$ for the path determined by σ. When extending the path from a node a to the node $a^{\sigma(X(a))}$, we say that the *answer to the query* $X(a)$ at a is $\sigma(X(a))$ and that the path *sets* the variable $X(a)$ to the value $\sigma(X(a))$. For a node a of $\mathrm{path}(\sigma)$, let $\beta(\sigma, a)$ be the restriction of σ to the variables that are queried in $\mathrm{path}(\sigma)$ in the segment of the path that goes from the source to a. For each node a of the branching program, let $\beta(a)$ be the maximal partial assignment that is contained in every $\beta(\sigma, a)$ for all σ such that $\mathrm{path}(\sigma)$ passes through a. Equivalently, this is the set of all those assignments $x_i \mapsto \gamma$ for which the query x_i is made, and answered by γ, along every consistent path from the source to a. If the program is read-once, the consistency condition becomes redundant.

The *falsified clause search problem* for an unsatisfiable CNF formula F is the task of finding a clause $C \in F$ that is falsified by a given truth value assignment σ. A branching program P on the variables $\mathit{Vars}(F)$ *solves* the falsified clause search problem for F if each sink is labelled by a clause of F such that for every assignment σ, the clause that labels the

sink reached by path(σ) is falsified by σ. The minimal size of any regular resolution refutation of an unsatisfiable CNF formula F is exactly the same as the minimal size of any read-once branching program solving the falsified clause search problem for F. This can be seen by taking the refutation DAG and reversing the edges to get a branching program or vice versa. For a formal proof see, e.g., [31, Theorem 4.3].

The k-clique formula. In order to analyse the complexity of resolution proofs that establish that a given graph does not contain a k-clique we must formulate the problem as a propositional formula in conjunctive normal form (CNF). We consider two distinct encodings for the clique problem originally defined in [5].

The first propositional encoding we present, $\mathit{Clique}(G,k)$, is based on mapping of vertices to clique members. This formula is defined over variables $x_{v,i}$ ($v \in V, i \in [k]$) and consists of the following set of clauses:

$$\neg x_{u,i} \vee \neg x_{v,j} \qquad i,j \in [k], i \neq j, u,v \in V, \{u,v\} \notin E \ , \tag{2.2a}$$

$$\bigvee_{v \in V} x_{v,i} \qquad i \in [k] \ , \tag{2.2b}$$

$$\neg x_{u,i} \vee \neg x_{v,i} \qquad i \in [k], u,v \in V, u \neq v \ , \tag{2.2c}$$

We refer to (2.2a) as *edge axioms*, (2.2b) as *clique axioms* and (2.2c) as *functionality axioms*. Note that $\mathit{Clique}(G,k)$ is satisfiable if and only if G contains a k-clique, and that this is true even if clauses (2.2c) are omitted—we write $\mathit{Clique}^*(G,k)$ to denote this formula with only clauses (2.2a) and (2.2b).

The second version of clique formulas that we consider is the block encoding $\mathit{Clique}_{\text{block}}(G,k)$. This formula differs from the previous ones in that it requires a k-clique that has a certain "block-respecting" structure. Let $V_1 \dot\cup V_2 \dot\cup \cdots \dot\cup V_k = V$ be a *balanced* k-partition of V, that is a partition of V into k disjoint sets each of them of size at most one integer away from $\frac{|V|}{k}$. The formula $\mathit{Clique}_{\text{block}}(G,k)$, defined over variables x_v, encodes the fact that the graph contains a *transversal k-clique*, that is, a k-clique in which each clique member belongs to a different block. Formally, for any positive k and any graph G, the formula $\mathit{Clique}_{\text{block}}(G,k)$ consists of the following set of clauses:

$$\neg x_u \vee \neg x_v \qquad u,v \in V, u \neq v, \{u,v\} \notin E \ , \tag{2.3a}$$

$$\bigvee_{v \in V_i} x_v \qquad i \in [k] \ , \tag{2.3b}$$

$$\neg x_u \vee \neg x_v \qquad i \in [k], u,v \in V_i, u \neq v \ . \tag{2.3c}$$

We refer to (2.3a) as *edge axioms*, (2.3b) as *clique axioms*, and (2.3c) as *functionality axioms*.

Note that a graph can contain a k-clique but contain no transversal k-clique for a given partition. Intuitively it is clear that proving that a graph does not contain a transversal k-clique should be easier than proving it does not contain any k-clique, since any proof of the latter fact must in particular establish the former. We make this intuition formal below.

Lemma 2.2 ([5]). *For any graph G and any $k \in \mathbb{N}^+$, the size of a minimum regular resolution refutation of $\mathit{Clique}(G,k)$ is bounded from below by the size of a minimum regular resolution refutation of $\mathit{Clique}_{\text{block}}(G,k)$.*

This lemma was proven in [5] for tree-like and for general resolution via a restriction argument, and it is straightforward to see that the same proof holds for regular resolution as well.

3 GRAPHS THAT ARE EASY FOR REGULAR RESOLUTION

Before proving our main $n^{\Omega(k)}$ lower bound, in this section we exhibit classes of graphs whose clique formulas have regular resolution refutations of fixed-parameter tractable length, i.e., length $f(k) \cdot n^{O(1)}$ for some function f. This illustrates the strength of regular resolution for the k-clique problem. We note that the upper bounds claimed in this section hold not only for $\mathit{Clique}(G, k)$ but even for the subformula $\mathit{Clique}^*(G, k)$ that omits the functionality axioms (2.2c).

The first example is the class of $(k-1)$-colourable graphs. Such graphs are hard for tree-like resolution [8], and the known algorithms that distinguish them from graphs that contain k-cliques are highly non-trivial [28, 35]. The second example is the class of graphs that have a homomorphism into a fixed k-clique free graph.

Recall that a homomorphism from a graph $G = (V, E)$ into a graph $G' = (V', E')$ is a mapping $h : V \to V'$ that maps edges $\{u, v\} \in E$ into edges $\{h(u), h(v)\} \in E'$. A graph is $(k-1)$-colourable if and only if it has a homomorphism into the $(k-1)$-clique, which is of course k-clique free. Therefore our second example is a generalization of the first one (but the function $f(k)$ becomes larger).

Both upper bounds follows from a generic procedure, based on Algorithm 1, that builds read-once branching programs for the falsified clause search problem for $\mathit{Clique}^*(G, k)$.

Given a k-clique free graph G define

$$I(G) = \left\{ G\left[\widehat{N}(R)\right] \;:\; R \text{ is a clique in } G \right\} \ . \tag{3.1}$$

Proposition 3.1. *There is an efficiently constructible read-once branching program for the falsified clause search problem on formula $\mathit{Clique}^*(G, k)$ of size at most $|I(G)| \cdot k^2 \cdot |V(G)|^2$.*

Proof. We build the branching program recursively, following the strategy laid out by Algorithm 1. For the base case $k = 1$, G must be the graph with no vertices. The branching program is a single sink node that outputs the clique axiom of index 1, i.e., the empty clause.

For $k > 1$, fix $n = |V(G)|$ and an ordering $v_1, \ldots, v_n$ of the vertices in $V(G)$. We first build a decision tree T by querying the variables $x_{v_1,k}, x_{v_2,k}, \ldots$ in order, until we get an answer 1, or until all variables with second index k have been queried. If $x_{v_j,k} = 0$ for all $j \in [n]$ then the kth clique axiom (2.2b) is falsified by the assignment (see line 9). Otherwise, let v be the first vertex in the order where $x_{v,k} = 1$. The decision tree now queries $x_{w,i}$ for all $w \in V(G) \setminus N(v)$ and all $i < k$ to check whether an edge axiom involving v is falsified (lines 4–5). If any of these variables is set to 1 the branching stops and the leaf node is labelled with the corresponding edge axiom $\neg x_{v,k} \vee \neg x_{w,i}$.

The decision tree T built so far has at most kn^2 nodes, and we can identify n "open" leaf nodes $a_{v_1}, a_{v_2}, \ldots, a_{v_n}$, where a_{v_i} is the leaf node reached by the path that sets $x_{v_i,k} = 1$ and that does not yet determine the answer to the search problem. Let us focus on a specific node a_v for some $v \in V(G)$. The partial assignment $\mathrm{path}(a_v)$ sets v to be the kth member of the clique and every vertex in $V(G) \setminus N(v)$ to not be in the clique. Let G_v be the subgraph induced on G by $N(v)$, let S_v be the set of variables $x_{w,i}$ for $w \in N(v)$ and $i < k$, and let ρ_v be the partial assignment setting $x_{w,i} = 0$ for $w \in V(G) \setminus N(v)$ and $i < k$. Clearly $\rho_v \subseteq \mathrm{path}(a_v)$.

By the inductive hypothesis there exists a branching program B_v that solves the search problem on $\mathit{Clique}^*(G_v, k-1)$ querying only variables in S_v. This corresponds to the recursive call for the subgraph G_v and $k-1$ (lines 6–8). If we attach each B_v to a_v we get a complete branching program for $\mathit{Clique}^*(G, k)$. This is read-once because B_v only queries variables in S_v and these variables are not in $\mathrm{path}(a_v)$.

To prove that the composed program is correct we consider an assignment σ to the variables in S_v and show that the clause output by B_v on σ is also a valid output for the search problem on $\mathit{Clique}^*(G, k)$, i.e., it is falsified by the assignment

Algorithm 1 Read-once branching program for the falsified clause search problem on $\mathit{Clique}^*(G,k)$.

Input : $k \in \mathbb{N}^+$, a k-clique free graph G, an assignment $\alpha : \{x_{v,i} \text{ for } v \in V(G), i \in [k]\} \to \{0,1\}$
Output : A clause of $\mathit{Clique}^*(G,k)$ falsified by α

```
Search(G, k, α): begin
    for v ∈ V(G) do
        if α(x_{v,k}) = 1 then
            for w ∈ V(G) \ N(v) and i < k do
                if α(x_{w,i}) = 1 then return edge axiom ¬x_{v,k} ∨ ¬x_{w,i} (2.2a)
            G' ← G[N(v)]
            α' ← α restricted to variables x_{w,j} for w ∈ V(G') and 1 ≤ j ≤ k − 1
            return Search(G', k − 1, α')
    return the kth clique axiom (2.2b)
```

$\mathrm{path}(a_v) \cup \sigma$. Actually we show the stronger claim that it is falsified by $\rho_v \cup \sigma$, which is a subset of $\mathrm{path}(a_v) \cup \sigma$. To this end, note that if the output of B_v on σ is an edge axiom of $\mathit{Clique}^*(G_v, k-1)$, this must be some $\neg x_{u,i} \vee \neg x_{w,j}$ for $i, j < k$, which is also an edge axiom of $\mathit{Clique}^*(G,k)$ and is falsified by σ. Now if the output of B_v on σ is the ith clique axiom of $\mathit{Clique}^*(G_v, k-1)$, then σ falsifies $\bigvee_{w \in N(v)} x_{v,i}$, and therefore $\rho_v \cup \sigma$ falsifies the ith clique axiom of $\mathit{Clique}^*(G,k)$.

The construction so far is correct but produces a very large branching program (in particular it has tree-like structure on top). In order to create a smaller branching program, we observe that if $u, v \in V(G)$ are such that $N(u) = N(w)$ then $G_u = G_w$, $B_u = B_w$ and $\rho_u = \rho_w$. This observation allows us to merge together all nodes a_v that have the same value of $N(v)$ into a single node, and to identify all the corresponding copies of the same branching program B_v. Now let us focus on some node a^* obtained by this merge process, and pick arbitrarily some a_v that was merged into it (the specific choice is irrelevant). By construction ρ_v is consistent with all paths reaching a^*, but we can claim further: ρ_v is consistent with all paths *passing through* a^* because B_v only queries variables in S_v, which is disjoint from the domain of ρ_v. Because of this last fact all paths that pass through node a^* and reach an output node b^* in the attached copy of B_v must contain the partial assignment $\rho_v \cup \sigma$, where σ is the common partial assignment consistent with all paths from the root of B_v to b^*. If b^* outputs an edge axiom, this is already falsified by σ because of the correctness of B_v. If b^* outputs the ith clique axiom, the correctness of B_v guarantees that σ falsifies the ith axiom for G_v, and therefore $\rho_v \cup \sigma$ falsifies the ith clique axiom of G. Hence the new branching program is correct.

This merge process leads to having only one subprogram for each distinct induced subgraph at each level of the recursion. In order to bound the size of this program, we decompose it into k levels. The source is at level zero and corresponds to the graph G. At level i there are nodes corresponding to all subgraphs induced by the common neighbourhood of cliques of size i. Each node in the ith level connects to the nodes of the $(i+1)$th level by a branching program of size at most kn^2. Notice that an induced subgraph in $I(G)$ cannot occur twice in the same layers, so the total size of the final branching program is at most $|I(G)| \cdot k^2 n^2$ nodes. □

We now proceed to prove the upper bounds mentioned previously. A graph G that has a homomorphism into a small k-clique free graph H may still have a large set $I(G)$, making Proposition 3.1 inefficient. The first key observation is that if G has a homomorphism into a graph H then it is a subgraph of a blown up version of H, namely, of a graph obtained by transforming each vertex of H into a "cloud" of vertices where a cloud does not contain any edge, two clouds corresponding to two adjacent vertices in H have all possible edges between them, and two clouds corresponding to

two non-adjacent vertices in H have no edges between them. A second crucial point is that if G' is a blown up version of H then it turns out that $|I(G')| = |I(H)|$, making Proposition 3.1 effective for G'. The upper bound then follows from observing that the task of proving that G is k-clique free should not be harder than the same task for a supergraph of G. Indeed Fact 3.2 formalises this intuition. It is interesting to observe that the constructions in Proposition 3.1 and in Fact 3.2 are efficient. The non-constructive part is guessing the homomorphism to H.

Fact 3.2. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs with no k-clique such that $V \subseteq V'$ and $E \subseteq E' \cap \binom{V}{2}$. If Clique$^*(G', k)$ has a (regular) refutation of length L, then Clique$^*(G, k)$ has a (regular) refutation of length at most L.*

Proof. Consider the partial assignment ρ that sets $x_{v,i} = 0$ for every $v \notin V$ and $i \in [k]$. The restricted formula *Clique*$^*(G', k)\restriction_\rho$ is isomorphic to *Clique*$^*(\widetilde{G}, k)$, where $V(\widetilde{G}) = V$ and $E(\widetilde{G}) = E' \cap \binom{V}{2}$, and thus, by Fact 2.1, has a (regular) refutation π of length at most L. Removing edges from a graph only introduces additional edge axioms (2.2a) in the corresponding formula, therefore *Clique*$^*(\widetilde{G}, k) \subseteq$ *Clique*$^*(G, k)$ and π is a valid refutation of *Clique*$^*(G, k)$ as well. □

It was shown in [8] that the k-clique formula of a complete $(k-1)$-partite graph on n vertices has a regular resolution refutation of length $2^k n^{O(1)}$, although the regularity is not stressed in that paper. Since it is instructive to see how this refutation is constructed in this framework, we give a self-contained proof.

Proposition 3.3 ([8, Proposition 5.3]). *If G is a $(k-1)$-colourable graph on n vertices, then Clique$^*(G, k)$ has a regular resolution refutation of length at most $2^k k^2 n^2$.*

Proof. Let $V = V(G)$ and let $V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_{(k-1)}$ be a partition of V into colour classes. Define the graph $G' = (V, E')$ where the edge set E' has an edge between any pair of vertices belonging to two different colour classes. Clearly G is a subgraph of G'. Observe that any clique R in G' has at most one vertex in each colour class, and that the common neighbours of R are all the vertices in the colour classes not touched by R.

Therefore, there is a one-to-one correspondence between the members of $I(G')$ and the subsets of $[k-1]$. By Proposition 3.1 there is a read-once branching program for the falsified clause search problem on formula *Clique*$^*(G', k)$ of size at most $2^k k^2 n^2$. This read-once branching program corresponds to a regular resolution refutation of *Clique*$^*(G', k)$ of the same size. By Fact 3.2 there must be a regular resolution refutation of size at most $2^k k^2 n^2$ for *Clique*$^*(G, k)$ as well. □

Next we generalize Proposition 3.3 to graphs G that have a homomorphism to a k-clique free graph H.

Proposition 3.4. *If G is a graph on n vertices that has a homomorphism into a k-clique free graph H on m vertices, then Clique$^*(G, k)$ has a regular resolution refutation of length at most $m^k k^2 n^2$.*

Proof. Fix a homomorphism $h : V(G) \to V(H)$ and an ordering $u_1, \ldots, u_m$ of the vertices of H. Let $V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_m$ be the partition of $V(G)$ such that V_i is the set of vertices of G mapped to u_i by h. We define the graph $G' = (V, E')$ where

$$E' = \bigcup_{\{u_i, u_j\} \in E(H)} V_i \times V_j \ , \tag{3.2}$$

that is, G' is a blown up version of H that contains G as a subgraph. To prove our result we note that, by Proposition 3.1, there is a read-once branching program for the falsified clause search problem on *Clique*$^*(G', k)$—and hence also a regular resolution refutations of the same formula—of size at most $|I(G')| \cdot k^2 n^2$. This implies that, by Fact 3.2, there is a regular resolution refutation of *Clique*$^*(G, k)$ of at most the same size.

To conclude the proof it remains only to show that $|I(G')| \leq m^k$. By construction, h maps injectively a clique $R \subseteq V(G')$ into a clique $R_H \subseteq V(H)$ of the same size. Moreover, note that if $U = \widehat{N}(R_H)$, then $\widehat{N}(R) = \cup_{u_i \in U} V_i$. Therefore, for any clique $R' \subseteq V(G')$ that is mapped by h to R_H it holds that $\widehat{N}(R) = \widehat{N}(R')$, i.e., $\widehat{N}(R')$ is completely characterized by the clique in H it is mapped to. Thus $I(G)$ has at most one element for each clique in H and we have that $|I(G')| = |I(H)|$. Finally, note that $|I(H)| \leq m^k$ since, being k-clique free, H cannot have more than $\sum_{i=0}^{k-1} m^i \leq m^k$ cliques. □

4 RANDOM GRAPHS ARE HARD FOR REGULAR RESOLUTION

The main result of this paper is an average case lower bound of $n^{\Omega(k)}$ for regular resolution for the k-clique problem. As we saw in Section 2, the k-clique problem can be encoded in different ways and depending on the preferred formula the range of k for which we can obtain a lower bound differs. In this section we present a summary of our results for the different encodings.

Theorem 4.1. *For any real constant $\epsilon > 0$, any sufficiently large integer n, any positive integer $k \leq n^{1/4-\epsilon}$, and any real $\xi > 1$, if $\boldsymbol{G} \sim \mathscr{G}(n, n^{-2\xi/(k-1)})$ is an Erdős-Rényi random graph, then, with probability at least $1 - \exp(-\sqrt{n})$, any regular resolution refutation of $\mathit{Clique}_{\text{block}}(\boldsymbol{G}, k)$ has length at least $n^{\Omega(k/\xi^2)}$.*

The parameter ξ determines the density of the graph: the larger ξ the sparser the graph and the problem of determining whether $\boldsymbol{G}$ contains a k-clique becomes easier. For constant ξ, the edge probability implies the graph $\boldsymbol{G}$ has clique number concentrated around k/ξ and the theorem yields a $n^{\Omega(k)}$ lower bound which is tight up to the multiplicative constant in the exponent. The lower bound decreases smoothly with the edge density and is non-trivial for $\xi = o(\sqrt{k})$.

A problem which is closely related to the problem we consider is that of distinguishing a random graph sampled from $\mathscr{G}(n, p)$ from a random graph from the same distribution with a planted k-clique. The most studied setting is when $p = 1/2$. In this scenario the problem can be solved in polynomial time with high probability for $k \approx \sqrt{n}$ [1, 32]. It is still an open problem whether there exists a polynomial time algorithm solving this problem for $\log n \ll k \ll \sqrt{n}$. For $\boldsymbol{G} \sim \mathscr{G}(n, 1/2)$, setting $\xi = \frac{k-1}{2\log_2(n)}$, Theorem 4.1 implies that to refute $\mathit{Clique}_{\text{block}}(\boldsymbol{G}, k)$ asymptotically almost surely regular resolution requires $n^{\Omega(\log^2(n)/k)}$ size; which is $n^{\Omega(\log n)}$ size for $k = O(\log n)$ and super-polynomial size for $k = o(\log^2 n)$. We note that, in the case $k = O(\log n)$, the lower bound is tight. This follows from Proposition 3.1 since asymptotically almost surely there are at most $n^{O(\log n)}$ different cliques in $\boldsymbol{G} \sim \mathscr{G}(n, 1/2)$ (because asymptotically almost surely the largest clique has size at most $2\log n$) and, therefore, the set $I(\boldsymbol{G})$ in Proposition 3.1 has size at most $n^{O(\log n)}$.

An interesting question is whether Theorem 4.1 holds for larger values of k. We show that for the formula $\mathit{Clique}(G, k)$ (recall that by Lemma 2.2 this encoding is easier for the purpose of lower bounds) we can prove the lower bound for $k \leq n^{1/2-\epsilon}$ as long as the edge density of the graph is close to the threshold for containing a k-clique.

Theorem 4.2. *For any real constant $\epsilon > 0$, any sufficiently large integer n, any positive integer k, and any real $\xi > 1$ such that $k\sqrt{\xi} \leq n^{1/2-\epsilon}$, if $\boldsymbol{G} \sim \mathscr{G}(n, n^{-2\xi/(k-1)})$ is an Erdős-Rényi random graph, then, with probability at least $1 - \exp(-\sqrt{n})$, any regular resolution refutation of $\mathit{Clique}(\boldsymbol{G}, k)$ has length at least $n^{\Omega(k/\xi^2)}$.*

In this paper we prove Theorem 4.1 and we refer to the conference version of this paper [2] for the proof of Theorem 4.2. We note, however, that both proofs are very similar and having seen one it is an easy exercise to obtain the other. The proof of Theorem 4.1 is deferred to Section 6 and is based on a general lower bound technique we develop in Section 5.

5 CLIQUE-DENSENESS IMPLIES HARDNESS FOR REGULAR RESOLUTION

In this section we define a combinatorial property of graphs, which we call *clique-denseness*, and prove that if a k-clique-free graph G is clique-dense with the appropriate parameters, then this implies a lower bound $n^{\Omega(k)}$ on the length of any regular resolution refutation of the k-clique formula on G.

In order to argue that regular resolution has a hard time certifying the k-clique-freeness of a graph G, one property that seems useful to have is that for every small enough clique in the graph there are many ways of extending it to a larger clique. In other words, if $R \subseteq V$ forms a clique and R is small, we would like the common neighbourhood $\widehat{N}_V(R)$ to be large. This motivates the following definitions.

Definition 5.1 (Neighbour-dense set). Given $G = (V, E)$ and $q, r \in \mathbb{R}^+$, a set $W \subseteq V$ is *q-neighbour-dense for $R \subseteq V$* if $\left|\widehat{N}_W(R)\right| \geq q$. We say that W is *(r, q)-neighbour-dense* if it is q-neighbour-dense for every $R \subseteq V$ of size $|R| \leq r$.

If W is an (r, q)-neighbour-dense set, then we know that any clique of size r can be extended to a clique of size $r + 1$ in at least q different ways by adding some vertex of W. Note, however, that the definition of (r, q)-neighbour-dense is more general than this since R is not required to be a clique.

Next we define a more robust notion of neighbour-denseness. For some settings of r and q of interest to us it is too much to hope for a set W that is q-neighbour-dense for every $R \subseteq V$ of size at most r. In this case we would still like to be able to find a "mostly neighbour-dense" set W in the sense that we can "localize" bad (i.e., those for which W fails to be q-neighbour-dense) sets $R \subseteq V$ of size $|R| \leq r$.

Definition 5.2 (Mostly neighbour-dense set). Given $G = (V, E)$ and $r', r, q, s \in \mathbb{R}^+$ with $r' \geq r$, a set $W \subseteq V$ is *(r', r, q, s)-mostly neighbour-dense* if there exists a set $S \subseteq V$ of size $|S| \leq s$ such that for every $R \subseteq V$ with $|R| \leq r'$ for which W is not q-neighbour-dense, it holds that $|R \cap S| \geq r$.

In what follows, it might be helpful for the reader to think of r' and r as linear in k, and q and s as polynomial in n, where we also have $s \ll q$.

Now we are ready to define a property of graphs that makes it hard for regular resolution to certify that graphs with this property are indeed k-clique-free.

Definition 5.3 (Clique-dense graph). Given $k \in \mathbb{N}^+$ and $t, s, \varepsilon \in \mathbb{R}^+$, $1 \leq t \leq k$ we say that a graph $G = (V, E)$ with a k-partition $V_1 \cup \cdots \cup V_k = V$ is *(k, t, s, ε)-clique-dense* if there exist $r, q \in \mathbb{R}^+$, $r \geq 4k/t^2$, such that

(1) V_i is (tr, tq)-neighbour-dense for all $i \in [k]$, and
(2) every (r, q)-neighbour-dense set $W \subseteq V$ is (tr, r, q', s)-mostly neighbour-dense for $q' = \varepsilon r s^{1+\varepsilon} \log s$.

Remark 1 (The complete $(k-1)$-partite graph is not clique-dense). Since the property of clique-denseness in Definition 5.3 is a sufficient condition for the lower bound, it is worth to pause and observe that this property does not hold for examples such as $(k-1)$-colourable graphs, which have non-trivially short proofs.

Indeed, consider the $(k-1)$-colourable graph $G = (V, E)$ with balanced colour classes and maximum edge set. Namely, $V = \bigcup_c U_c$ for $c \in [k-1]$ and $|U_c| = n/(k-1)$, and the edges of G are all pairs $\{u, v\}$ for $u \in U_c$ and $v \in U_{c'}$ with $c \neq c'$. The graph G satisfies property (1) of clique-denseness for any k-partition of V that splits each colour class roughly equally among parts, but fails to satisfy property (2) in a rather extreme way. To see why, fix any integer $r < k - 1$ and let W be the union of $r + 1$ arbitrarily chosen colour classes. The set W is (r, q)-neighbour-dense for any q up to $n/(k-1)$, because the common neighbourhood of any r vertices in V must contain one of the colour classes $U_c \subseteq W$.

Can W be (tr, r, q', s)-mostly neighbour-dense for some choice of parameters? First note that $tr \geq r + 1$ (since $r \leq k$ implies $t \geq 2$) and that $\widehat{N}_W(R) = \emptyset$ for any set R of size $r + 1$ that has one vertex from each colour class in W. So in order for W to be (tr, r, q', s)-mostly neighbour-dense there should be a set S of size $s \ll q' \leq n/(k-1)$ that has a large intersection with any such R. This, however, is not possible since S cannot completely cover any of the colour classes in W (because $s \ll n/(k-1)$) and thus, for any choice of S, there are sets R completely disjoint from S for which $\widehat{N}_W(R) = \emptyset$.

Theorem 5.4. *Given $k \in \mathbb{N}^+$ and $t, s, \varepsilon \in \mathbb{R}^+$ if the graph $G = (V, E)$ with balanced k-partition $V_1 \cup \cdots \cup V_k = V$ is (k, t, s, ε)-clique-dense, then every regular resolution refutation of the CNF formula $Clique_{\text{block}}(G, k)$ has length at least $\Omega\big(s^{\varepsilon k/t^2}\big)$.*

The value of q' in Definition 5.3 can be tailored in order to prove Theorem 4.1 for slightly larger values of k. For example, setting $q' = 3\varepsilon s^{1+\varepsilon} \log s$ and making the necessary modifications in the proof would yield Theorem 4.1 for $k \ll n^{1/3}$ but for a smaller range of edge densities. A similar adjustment was done in the conference version of this paper [2] to obtain Theorem 4.2 for $k \ll n^{1/2}$.

We will spend the rest of this section establishing Theorem 5.4. Fix $r, q \in \mathbb{R}^+$ witnessing that G is (k, t, s, ε)-clique-dense as per Definition 5.3. We first note that we can assume that $tr \leq k$ since otherwise, by property 1 of Definition 5.3, G contains a block-respecting k-clique and the theorem follows immediately.

By the discussion in Section 2 it is sufficient to consider read-once branching programs, since they are equivalent to regular resolution refutations, and so in what follows this is the language in which we will phrase our lower bound. Thus, for the rest of this section let P be an arbitrary, fixed read-once branching program that solves the falsified clause search problem for $Clique_{\text{block}}(G, k)$. We will use the convention of referring to "vertices" of the graph G and "nodes" of the branching program P to distinguish between the two. We sometime abuse notation and say that a vertex $v \in V$ is set to 0 or to 1 when we mean that the corresponding variable x_v is set to 0 or to 1.

Recall that for a node a of P, $\beta(a)$ denotes the maximal partial assignment that is contained in every $\beta(\sigma, a)$ for all σ such that $\text{path}(\sigma)$ passes through a, where $\beta(\sigma, a)$ is the restriction of σ to the variables that are queried in $\text{path}(\sigma)$ in the segment of the path that goes from the source to a. For any partial assignment β we write β^1 to denote the partial assignment that contains exactly the variables that are set to 1 in β. Clearly, if β falsifies an edge axiom or a functionality axiom, then so does β^1. Furthermore, for any $\gamma \supseteq \beta$, if β falsifies an axiom so does γ. We will use this monotonicity property of partial assignments throughout the proof.

For each node a of P and each index $i \in [k]$ we define two sets of vertices

$$V_i^0(a) = \{u \in V_i \mid \beta(a) \text{ sets } x_u \text{ to } 0\} \tag{5.1a}$$

$$V_i^1(a) = \{u \in V_i \mid \beta(a) \text{ sets } x_u \text{ to } 1\} \tag{5.1b}$$

of G. Observe that for $\beta = \beta(a)$ the set of vertices referenced by variables in β^1 is $\bigcup_i V_i^1(a)$.

Intuitively, one can think of $V_i^0(a)$ and $V_i^1(a)$ as the only sets of vertices in V_i assigned 0 and 1, respectively, that are "remembered" at the node a (in the language of resolution, they correspond to negative and positive occurrences of variables in the clause D_a associated with the node a). Other assignments to vertices in V_i encountered along some path to a have been "forgotten" and may not be queried any more on any path starting at a. Formally, we say that a vertex v is *forgotten at a* if there is a path from the source of P to a passing through a node b where v is queried, but v is not in $V_i^0(a)$ nor in $V_i^1(a)$. Furthermore, we say index *i is forgotten at a* if some vertex $v \in V_i$ is forgotten at a. Of utter importance is the fact that these notions are persistent: if a variable or an index is forgotten at a node a, then it will also be the case for any node reachable from a by a path. We say that a path in P *ends in the ith clique axiom* if the clause that labels its last

node is the clique axiom (2.3b) of $Clique_{\text{block}}(G, k)$ with index i. The above observation implies that the index i cannot be forgotten at any node along such a path.

We establish our lower bound via a bottleneck counting argument for paths in P. To this end, let us define a distribution $\mathscr{D}$ over paths in P by the following random process. The path starts at the source and ends whenever it reaches a sink of P. At an internal node a with successor nodes a^0 and a^1, reached by edges labelled 0 and 1 respectively, the process proceeds as follows.

(1) If $X(a) = x_u$ for $u \in V_i$ and i is forgotten at a then the path proceeds via the edge labelled 0 to a^0.
(2) If $X(a) = x_u$ and $\beta(a) \cup \{x_u = 1\}$ falsifies an edge axiom (2.3a) or a functionality axiom (2.3c), then the path proceeds to a^0.
(3) Otherwise, an independent $s^{-(1+\varepsilon)}$-biased coin is tossed with outcome $\gamma \in \{0, 1\}$ and the random path proceeds to a^γ.

We say that in cases 1 and 2 the answer to the query $X(a)$ is *forced*. Note that any path α in the support of $\mathscr{D}$ must end in a clique axiom since α does not falsify any edge or functionality axiom by item 2 in the construction. Moreover, a property that will be absolutely crucial is that only answers 0 can be forced—answers 1 are always the result of a coin flip.

Claim 5.5. *Every path in the support of $\mathscr{D}$ sets at most k variables to* 1.

Proof. Let α be a path in the support of $\mathscr{D}$. We argue that for each $i \in [k]$ at most one vertex $u \in V_i$ is such that the variable x_u is set to 1 on α. Let a and b be two nodes that appear in this order in α. If for some $i \in [k]$, and for some $u, v \in V_i$, x_u is set to 1 by α at node a and x_v is queried at b, then $v \neq u$ by regularity and, by definition of $\mathscr{D}$, the answer to query x_v will be forced to 0, either to avoid violating a functionality or an edge axiom, or because i is forgotten at b. □

Let us call a pair (a, b) of nodes of P *useful* if there exists an index i such that $V_i^1(b) = \emptyset$, i is not forgotten at b, and the set $V_i^0(b) \setminus V_i^0(a)$ is (r, q)-neighbour-dense. In particular, if a appears before b in some path, then $V_i^1(a) = \emptyset$ and $V_i^0(a) \subseteq V_i^0(b)$. For each useful pair (a, b), let $i(a, b)$ be an arbitrary but fixed index witnessing that (a, b) is useful. A path is said to *usefully* traverse a useful pair (a, b) if it goes through a and b in that order and sets at most $\lceil k/t \rceil$ variables to 1 between a and b (with a included and b excluded).

As already mentioned, the proof of Theorem 5.4 is based on a bottleneck counting argument in the spirit of [23], with the twist that we consider pairs of bottleneck nodes. To establish the theorem we make use of the following two lemmas which will be proven subsequently.

Lemma 5.6. *Every path in the support of $\mathscr{D}$ usefully traverses a useful pair.*

Lemma 5.7. *For every useful pair (a, b), the probability that a random $\boldsymbol{\alpha}$ chosen from $\mathscr{D}$ usefully traverses (a, b) is at most $2s^{-\varepsilon r/2}$.*

Combining the above lemmas, it is immediate to prove Theorem 5.4. By Lemma 5.6 the probability that a random path $\boldsymbol{\alpha}$ sampled from $\mathscr{D}$ usefully traverses some useful pair is 1. By Lemma 5.7, for any fixed useful pair (a, b), the probability that a random $\boldsymbol{\alpha}$ usefully traverses (a, b) is at most $2s^{-\varepsilon r/2}$. By a standard union bound argument, it follows that the number of useful pairs is at least $\frac{1}{2}s^{\varepsilon r/2}$, so the number of nodes in P cannot be smaller than $\Omega\left(s^{\varepsilon r/4}\right) \geq \Omega\left(s^{\varepsilon k/t^2}\right)$ (recall that $r \geq 4k/t^2$ according to Definition 5.3).

To conclude the proof it remains only to establish Lemmas 5.6 and 5.7.

Proof of Lemma 5.6. Consider any path in the support of $\mathscr{D}$. As we already remarked, this path ends in the i^*th clique axiom for some $i^* \in [k]$ which in particular implies that $V^1_{i^*}(b) = \emptyset$ and that i^* is not forgotten at any b along this path. By Claim 5.5, the path sets at most k variables to 1 and hence we can split it into t pieces by nodes $a_0, a_1, \ldots, a_t$ (a_0 is the source, a_t the sink) so that between a_j and a_{j+1} at most $\lceil k/t \rceil$ variables are set to 1. It remains to prove that for at least one $j \in [t]$ the set

$$W_j = V^0_{i^*}(a_j) \setminus V^0_{i^*}(a_{j-1}) \tag{5.2}$$

is (r, q)-neighbour-dense. Note that this will prove Lemma 5.6 since by construction (a_{j-1}, a_j) is then a pair that is usefully traversed by the path.

Towards contradiction, assume instead that no W_j is (r, q)-neighbour-dense, i.e., that for all $j \in [t]$ there exists a set of vertices $R_j \subseteq V$ with $|R_j| \leq r$ such that $\left|\widehat{N}_{W_j}(R_j)\right| \leq q$. Let $R = \bigcup_{j\in[t]} R_j$. Since the path ends in the i^*th clique axiom we have $V^0_{i^*}(a_t) = V_{i^*}$. It follows that the sets $W_1, \ldots, W_t$ in (5.2) form a partition of V_{i^*}, and therefore

$$\left|\widehat{N}_{V_{i^*}}(R)\right| = \sum_{j\in[t]} \left|\widehat{N}_{W_j}(R)\right| \leq \sum_{j\in[t]} \left|\widehat{N}_{W_j}(R_j)\right| \leq tq \ . \tag{5.3}$$

Since $|R| \leq \sum_{j\in[t]} |R_j| \leq tr$ this contradicts the assumption that V_{i^*} is (tr, tq)-neighbour-dense. Lemma 5.6 follows. □

Proof of Lemma 5.7. Fix a useful pair (a, b). Let E denote the event that a random path sampled from $\mathscr{D}$ usefully traverses (a, b). Let $i^* = i(a, b)$, $V^1(a) = \bigcup_{j\in[k]} V^1_j(a)$, and $W = V^0_{i^*}(b) \setminus V^0_{i^*}(a)$. Notice that W is guaranteed to be (r, q)-neighbour-dense by our definition of $i(a, b)$. Since G is (k, t, s, ε)-clique-dense by assumption, this implies that W is (tr, r, q', s)-mostly neighbour-dense, and we let S be the set that witnesses this as per Definition 5.2. We bound the probability of the event E by a case analysis based on the size of the set $V^1(a)$. We remark that all probabilities in the calculations that follow are over the choice of $\boldsymbol{\alpha} \sim \mathscr{D}$.

Case 1 ($|V^1(a)| > r/2$): In this case, we simply prove that already the probability of reaching a is small. By definition of $V^1(a)$, we have that $|\beta^1(a)| = |V^1(a)|$. Recall that every answer 1 is necessarily the result of a $s^{-(1+\varepsilon)}$-biased coin flip, and that all these decisions are irreversible. That is, if a path ever decides to set a variable in $V^1(a)$ to 0, then its case is lost and it is guaranteed to miss a. Thus we can upper bound the probability of the event E by the probability that a random $\boldsymbol{\alpha}$ passes through a, and, in particular, by the probability of setting all variables in $\beta^1(a)$ to 1 as follows:

$$\Pr[\mathsf{E}] \leq \Pr[\boldsymbol{\alpha} \text{ passes through } a] \tag{5.4}$$

$$\leq \left(s^{-(1+\varepsilon)}\right)^{|\beta^1(a)|} \tag{5.5}$$

$$\leq s^{-\varepsilon|V^1(a)|} \tag{5.6}$$

$$\leq 2s^{-\varepsilon r/2} \ . \tag{5.7}$$

Case 2 ($|V^1(a)| \leq r/2$): For every path α, let $R(\alpha)$ denote the set of vertices u set to 1 by the path α at some node between a and b (with a included and b excluded); note that $R(\alpha) = \emptyset$ if α does not go through a and b, and that $|R(\alpha)| \leq \lceil k/t \rceil$ for all paths α that satisfy the event E. For the sets

$$\mathcal{R}_0 = \{R : |R| \leq \lceil k/t \rceil \text{ and } \left|\widehat{N}_W(R \cup V^1(a))\right| < q'\} \tag{5.8a}$$

$$\mathcal{R}_1 = \{R : |R| \leq \lceil k/t \rceil \text{ and } \left|\widehat{N}_W(R \cup V^1(a))\right| \geq q'\} \tag{5.8b}$$

we have that

$$\Pr[\mathsf{E}] = \Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_0] + \Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_1] \ . \tag{5.9}$$

The first term in (5.9) is bounded from above by the probability of $R(\boldsymbol{\alpha}) \in \mathcal{R}_0$. Note that $|R| \leq \lceil k/t \rceil \leq 2k/t \leq tr/2$ (since $r \geq 4k/t^2$) for $R \in \mathcal{R}_0$. Hence we have $|R \cup V^1(a)| \leq tr/2 + r/2 \leq tr$ and therefore $|(R \cup V^1(a)) \cap S| \geq r$ by the choice of S. Thus, the probability of $R(\boldsymbol{\alpha}) \in \mathcal{R}_0$ is bounded by the probability that $|R(\boldsymbol{\alpha}) \cap S| \geq r/2$ since $|V^1(a)| \leq r/2$. But since S is small, we can now apply the union bound and conclude that

$$\Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_0] \leq \Pr[R(\boldsymbol{\alpha}) \in \mathcal{R}_0] \tag{5.10}$$

$$\leq \Pr[|R(\boldsymbol{\alpha}) \cap S| \geq r/2] \tag{5.11}$$

$$\leq \binom{|S|}{r/2} (s^{-(1+\varepsilon)})^{r/2} \tag{5.12}$$

$$\leq |S|^{r/2} s^{-(1+\varepsilon)r/2} \tag{5.13}$$

$$\leq s^{-\varepsilon r/2} \ , \tag{5.14}$$

where for (5.12) we used the same "irreversibility" argument as in Case 1.

We now bound the second term in (5.9). First note that, by definition of W, if α is a path that passes through a and b in this order, then all $u \in W$ must be set to 0 in α at some node between a and b. For each path in the support of $\mathscr{D}$ that passes through a and b, some of the vertices in W will be set to zero as a result of a coin flip and others will be forced choices.

Fix a path α contributing to the second term in (5.9). We claim that along this path all the $\geq q'$ variables in $\widehat{N}_W(R(\alpha) \cup V^1(a))$ are set to 0 as a result of a coin flip. Indeed, since $V^1_{i^*}(b) = \emptyset$ and i^* is not forgotten at b, by the monotonicity property the same holds for every node along α before b. This implies that the answer to a query of the form x_u ($u \in W$) made along α cannot be forced by neither item 1 (forgetfulness) in the definition of $\mathscr{D}$ nor by a functionality axiom. Moreover, since $V^1(c) \subseteq R(\alpha) \cup V^1(a)$ for any node c on the path α between a and b, it holds that all variables x_u with $u \in \widehat{N}_W(R(\alpha) \cup V^1(a))$ can not be forced to 0 by an edge axiom either.

The analysis of the second term in (5.9) is completed by the same type of argument as in Case 1, where we again use the fact that, due to the read-once property of the branching program, the decisions that the random path makes are irreversible:

$$\Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_1] \leq \Pr[\boldsymbol{\alpha} \text{ flips } \geq q' \text{ coins and gets 0-answers}] \tag{5.15}$$

$$\leq (1 - s^{-(1+\varepsilon)})^{q'} \tag{5.16}$$

$$\leq s^{-\varepsilon r/2} \ . \tag{5.17}$$

Adding (5.14) and (5.17) we obtain the lemma. □

6 RANDOM GRAPHS ARE ALMOST SURELY CLIQUE-DENSE

In this section we show that asymptotically almost surely an Erdős-Rényi random graph $\boldsymbol{G} \sim \mathscr{G}(n,p)$ is (k,t,s,ε)-clique-dense for the right choice of parameters.

Theorem 6.1. *For any real constant $\varepsilon \in (0, 1/4)$, any sufficiently large integer n, any positive integer $k \leq n^{1/4-\varepsilon}$, and any real $\xi > 1$, if $\boldsymbol{G} \sim \mathscr{G}(n, n^{-2\xi/(k-1)})$ is an Erdős-Rényi random graph then with probability at least $1 - \exp(-\sqrt{n})$ it holds that $\boldsymbol{G}$ is (k,t,s,ε)-clique-dense with $t = 32\xi/\varepsilon$ and $s = \sqrt{n}$.*

As a corollary of Theorem 5.4 and Theorem 6.1 we obtain Theorem 4.1, the main result of this paper.

Proof of Theorem 4.1. Clearly $t \geq 1$ as required by Definition 5.3. We can also assume w.l.o.g. that $t \leq k$ since otherwise $k/\xi^2 \leq 32/(\xi\epsilon) \leq O(1)$ and the bound becomes trivial. By plugging in the parameters given by Theorem 6.1 to Theorem 5.4 we immediately get that any regular refutation π of $\mathit{Clique}_{\mathrm{block}}(G, k)$ has length

$$|\pi| \geq \Omega\big(s^{\varepsilon k/t^2}\big) \geq n^{\Omega(k/\xi^2)} \ , \tag{6.1}$$

as stated. □

We will spend the rest of this section proving Theorem 6.1.

Let $\delta = 2\xi/(k-1)$. We show that, with probability at least $1 - \mathrm{e}^{-\sqrt{n}}$, the random graph $\boldsymbol{G}$ is (k, t, s, ε)-clique-dense for parameters as in the statement of the theorem, $r = 4k/t^2$ and $q = \frac{n^{1-t\delta r}}{4kt}$.

Recall that $q' = \varepsilon r s^{1+\varepsilon} \log s$. Let us argue that the parameters we use satisfy constraints

$$t\delta r \leq \frac{\varepsilon}{2} \ , \tag{6.2}$$

$$\log k + tr \log n \leq \frac{n^{1-t\delta r}}{32k} \cdot \frac{2 \log n}{n^{1/2}} \ , \tag{6.3}$$

$$\frac{q n^{-t\delta r} s}{16tr} \geq \frac{n^{1+\varepsilon}}{256} \ , \tag{6.4}$$

$$q' \leq \frac{q n^{-t\delta r}}{4} \cdot \frac{\log n}{n^{\varepsilon/2}} \ , \tag{6.5}$$

$$tr \leq \frac{q}{2} \ , \tag{6.6}$$

which will be used further on in the proof.

As a first step note that

$$t\delta r = \frac{8\xi k}{t(k-1)} \leq \frac{\varepsilon}{2} \ , \tag{6.7}$$

and hence (6.2) holds. Equation (6.3) follows from the chain of inequalities

$$\log k + tr \log n \leq 2tr \log n = \frac{8k \log n}{t} \leq \frac{k \log n}{16} \leq \frac{n^{1/2-2\varepsilon} \log n}{16k} \leq \frac{n^{1-t\delta r}}{32k} \cdot \frac{2 \log n}{n^{1/2}} \ . \tag{6.8}$$

To obtain (6.4) observe that

$$\frac{q n^{-t\delta r} s}{16tr} = \frac{n^{1-2t\delta r+1/2}}{256k^2} \geq \frac{n^{1-2t\delta r+2\varepsilon}}{256} \geq \frac{n^{1+\varepsilon}}{256} \ . \tag{6.9}$$

To see that (6.5) holds, note that

$$q' = \frac{2\varepsilon k n^{(1+\varepsilon)/2} \log n}{t^2} \leq \frac{k^2 n^{(1+\varepsilon)/2} \log n}{16kt} \leq \frac{n^{1-3\varepsilon/2} \log n}{16kt} \leq \frac{q n^{-t\delta r}}{4} \cdot \frac{\log n}{n^{\varepsilon/2}} \ . \tag{6.10}$$

Finally, for (6.6), we just observe that

$$tr = \frac{4k}{t} \leq \frac{k^3}{8k^2} \leq \frac{n^{1-t\delta r}}{8kt} = \frac{q}{2} \ , \tag{6.11}$$

using the fact that $k \geq t$ and $k^3 \leq n^{1-t\delta r}$.

We must now prove that asymptotically almost surely $\boldsymbol{G}$ is (k, t, s, ε)-clique-dense for the chosen parameters. All probabilities in this section are over the choice of $\boldsymbol{G}$, and all previously introduced concepts like $\widehat{N}_W(R)$, neighbour-denseness etc. should be understood with respect to $\boldsymbol{G}$ as well (so that they are actually random variables and events in this sample space). Let $V = V(\boldsymbol{G})$ and $V_1 \cup \cdots \cup V_k = V$ be a balanced k-partition of V.

The fact that asymptotically almost surely V_i is (tr, tq)-neighbour-dense for all $i \in [k]$ is quite immediate. First, for any $i \in [k]$ and any $R \subseteq V$ with $|R| \leq tr$,

$$\mathbb{E}\left[\left|\widehat{N}_{V_i}(R)\right|\right] = |V_i \setminus R| n^{-\delta|R|} \geq \left(\frac{n}{k} - tr\right) n^{-\delta tr} \geq \left(\frac{n}{k} - \frac{q}{2}\right) n^{-\delta tr} \geq \frac{n^{1-\delta tr}}{2k} \ , \tag{6.12}$$

where the second-to-last inequality follows from (6.6) and the last inequality from the trivial fact that $q \leq \frac{n}{k}$. Hence, we can bound the probability that there exists an $i \in [k]$ such that V_i is not (tr, tq)-neighbour-dense by

$$\Pr\left[\exists i \in [k]\ \exists R \subseteq V,\ |R| = \lfloor tr \rfloor \wedge \left|\widehat{N}_{V_i}(R)\right| \leq tq\right] \leq k\binom{n}{tr} \max_{i,R} \Pr\left[\left|\widehat{N}_{V_i}(R)\right| \leq tq\right] \tag{6.13}$$

$$\leq k n^{tr} \max_{i,R} \Pr\left[\left|\widehat{N}_{V_i}(R)\right| \leq \frac{n^{1-t\delta r}}{4k}\right] \tag{6.14}$$

$$\leq k n^{tr} \exp\left(-\frac{n^{1-t\delta r}}{16k}\right) \tag{6.15}$$

$$\leq \exp\left(-\frac{n^{1-t\delta r}}{32k} \cdot \left(2 - 2\frac{\log n}{n^{1/2}}\right)\right) \tag{6.16}$$

$$\leq \mathrm{e}^{-\sqrt{n}} \ . \tag{6.17}$$

We note that (6.13) is a union bound, (6.14) follows from the definition of q, (6.15) is the multiplicative form of Chernoff bound (note that the events $v \in \widehat{N}_{V_i}(R) (v \in V \setminus R)$ are mutually independent), (6.16) follows from (6.3), and (6.17) holds for large enough n by (6.2) and the fact that $\varepsilon < 1/4$ and $k < n^{1/4}$.

All that is left to prove is that asymptotically almost surely G satisfies property 2 in Definition 5.3, that is that every (r, q)-neighbour-dense set $W \subseteq V$ is (tr, r, q', s)-mostly neighbour-dense. For shortness let P be the event that G satisfies this property. We wish to show that $\Pr[\neg \mathrm{P}] \leq \mathrm{e}^{-\Omega(n)}$, and it turns out that due to our choice of parameters we can afford to use the crude union bound over all 2^n choices of W.

To be more specific, let $Q(W)$ denote the event that W is (r, q)-neighbour-dense. Given an (r, q)-neighbour-dense set $W \subseteq V$ we will define a set S_W which will be a "candidate witness" of the fact that W is (tr, r, q', s)-mostly neighbour-dense. First observe that, since W is (r, q)-neighbour-dense and $q' \leq q$ by (6.5), any set $R \subseteq V$ with $|R| \leq tr$ and $\left|\widehat{N}_W(R)\right| \leq q'$ must be such that $|R| > r$. We will use a sequence of such sets R and construct S_W in a greedy fashion. To this end, the following definition will be useful. A tuple of sets $(R_1, \ldots, R_m)$ is said to be *r-disjoint* if $\left|R_i \cap \left(\bigcup_{j<i} R_j\right)\right| \leq r$ for every $i \in [m]$.

Fix an arbitrary ordering of the subsets of V. Define $\vec{R}_W = (R_1, \ldots, R_m)$ to be a maximally long tuple such that, for every $i = 1, \ldots, m$, the set R_i is the first in the ordering such that $|R_i| \leq tr$, $\left|\widehat{N}_W(R_i)\right| \leq q'$ and $\left|R_i \cap \left(\bigcup_{j<i} R_j\right)\right| \leq r$. Note that $\vec{R}_W$ is r-disjoint. Now let $S_W = \bigcup_{i \leq m} R_i$.

Observe that, by maximality of $\vec{R}_W$, any set $R \subseteq V$ with $|R| \leq tr$ and $\left|\widehat{N}_W(R)\right| \leq q'$ must be such that $|R \cap S| > r$. This implies that if $|S_W| \leq s$ then S_W witnesses the fact that W is (tr, r, q', s)-mostly neighbour-dense. Therefore we have that

$$\Pr[\neg \mathrm{P}] \leq \Pr[\exists W \subseteq V,\ Q(W) \wedge |S_W| > s] \ . \tag{6.18}$$

Moreover, let $\mathcal{W}$ be the collection of all pairs $(W, \vec{R})$ such that $W \subseteq V$, $\vec{R} = (R_1, \ldots, R_\ell)$ for $\ell = \lceil s/tr \rceil$, $R_j \subseteq V$ and $0 < |R_j| \leq tr$ for each $j \in [\ell]$, and $\vec{R}$ is r-disjoint. Notice that if there exists an (r, q)-neighbour-dense W such that $\vec{R}_W = (R_1, \ldots, R_m)$ and $|S_W| > s$, then $m \geq \ell$ and $(W, (R_1, \ldots, R_\ell)) \in \mathcal{W}$. Furthermore, by definition of $\vec{R}_W$, for

every $j \in [\ell]$ it holds that $\left|\widehat{N}_W(R_j)\right| \leq q'$. Hence we can conclude that

$$\Pr[\neg \mathsf{P}] \leq \Pr\left[\exists (W, \vec{R}) \in \mathcal{W},\ Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq q'\right] \tag{6.19}$$

$$\leq 2^n n^{tr\ell} \max_{(W,\vec{R})\in\mathcal{W}} \Pr\left[Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq q'\right] \tag{6.20}$$

$$\leq 2^n n^s \max_{(W,\vec{R})\in\mathcal{W}} \Pr\left[Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq \frac{q}{4} n^{-t\delta r}\right] \ , \tag{6.21}$$

where (6.21) follows for n large enough from the bound in (6.5).

Now fix $(W, \vec{R}) \in \mathcal{W}$ and let R_j^d (resp. R_j^c) be the subset of R_j disjoint from (resp. contained in) $\bigcup_{j'<j} R_{j'}$. Since $|R_j^c| \leq r$ by definition, it holds that if W is (r, q)-neighbour-dense then $\left|\widehat{N}_W(R_j^c)\right| > q$. Let $\mathsf{F}(j)$ be the event that $\left|\widehat{N}_W(R_j^c)\right| > q$ and $\left|\widehat{N}_W(R_j)\right| \leq \frac{q}{4} n^{-t\delta r}$. Note that $\Pr\left[Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq \frac{q}{4} n^{-t\delta r}\right]$ is at most $\Pr\left[\forall j \in [\ell],\ \mathsf{F}(j)\right]$. Let $\mathsf{F}'(j)$ be the event that $\mathsf{F}(j')$ holds for all $j' \in [j-1]$. We have that

$$\Pr\left[\forall j \in [\ell],\ \mathsf{F}(j)\right] = \prod_{j\in[\ell]} \Pr\left[\mathsf{F}(j) \mid \mathsf{F}'(j)\right] \ . \tag{6.22}$$

We can consider the factors of the previous product separately and bound each one by

$$\Pr\left[\mathsf{F}(j) \mid \mathsf{F}'(j)\right]$$

$$\leq \sum_{\substack{U\subseteq W\\ |U|\geq q}} \Pr\left[\left|\widehat{N}_U(R_j^d)\right| \leq \frac{q}{4} n^{-t\delta r} \ \middle|\ \widehat{N}_W(R_j^c) = U \wedge \mathsf{F}'(j)\right] \cdot \Pr\left[\widehat{N}_W(R_j^c) = U \ \middle|\ \mathsf{F}'(j)\right] \tag{6.23}$$

$$\leq \sum_{\substack{U\subseteq W\\ |U|\geq q}} \Pr\left[\left|\widehat{N}_U(R_j^d)\right| \leq \frac{q}{4} n^{-t\delta r}\right] \cdot \Pr\left[\widehat{N}_W(R_j^c) = U \ \middle|\ \mathsf{F}'(j)\right] \tag{6.24}$$

$$\leq \sum_{\substack{U\subseteq W\\ |U|\geq q}} \exp\left(-\frac{qn^{-t\delta r}}{16}\right) \cdot \Pr\left[\widehat{N}_W(R_j^c) = U \ \middle|\ \mathsf{F}'(j)\right] \tag{6.25}$$

$$= \exp\left(-\frac{qn^{-t\delta r}}{16}\right) \cdot \sum_{\substack{U\subseteq W\\ |U|\geq q}} \Pr\left[\widehat{N}_W(R_j^c) = U \ \middle|\ \mathsf{F}'(j)\right] \tag{6.26}$$

$$\leq \exp\left(-\frac{qn^{-t\delta r}}{16}\right) \ . \tag{6.27}$$

Equation (6.24) follows from the independence of any two events that involve disjoint sets of potential edges and (6.25) follows from the multiplicative Chernoff bound and the fact that

$$\mathbb{E}\left[\left|\widehat{N}_U(R_j^d)\right|\right] = |U \setminus R_j^d| n^{-\delta|R_j^d|} \geq (|U| - tr) n^{-\delta tr} \geq \frac{q}{2} n^{-\delta tr} \ . \tag{6.28}$$

So, putting everything together, we have that

$$\Pr[\neg \mathsf{P}] \leq 2^n n^s \exp\left(-\frac{qn^{-t\delta r}\ell}{16}\right) \leq \mathrm{e}^{(\log 2)n + \sqrt{n}\log n - (n^{1+\varepsilon})/256} \leq \mathrm{e}^{-\Omega(n)} \ , \tag{6.29}$$

where the last inequality holds for n large enough, and the second to last inequality follows immediately from the bound in (6.4). This concludes the proof of Theorem 6.1.

7 STATE-OF-THE-ART ALGORITHMS FOR CLIQUE

In this section we describe state-of-the-art algorithms for maximum clique and explain how regular resolution proofs bound from below the running time of these algorithms.

At the heart of most (if not all) of the state-of-the-art algorithms for maximum clique is a backtracking search, which in its simplest form examines all maximal cliques by enlarging a set of vertices that form a clique and backtracking when it certifies that the current set forms a maximal clique. A classical example of such a backtracking search is the Bron–Kerbosch [11] algorithm which enumerates all maximal cliques in a graph. This algorithm can be adapted to find a maximum clique as done in [13] improving the running time considerably by using a branch and bound strategy. At some point in the search tree it becomes clear that the current search-branch will not lead to a clique larger than the largest one found so far—in such cases the algorithm cuts off the search and backtracks immediately.

The most successful algorithms in practice are search trees with clever branch and bound strategies. In this section we will discuss the algorithm by Östergård [41] using Russian doll search and a collection of algorithms that use colour-based branch and bound strategies [22, 29, 50–54, 56–59, 61].

Östergård's algorithm. Östergård's algorithm [41] is a branch and bound algorithm that uses Russian doll search as a pruning strategy: it considers smaller subinstances recursively and solves them in ascending order using previous solutions as upper bounds. This algorithm, which is the main component of the Cliquer software, is often used in practice and has been available online since 2003 [40]. Cliquer is also the software of choice to compute maximum cliques in the open source mathematical software SageMath [55].

The `Cliquer`(G) algorithm described in Figure 2 is essentially the same as Algorithm 2 in [41]. The algorithm first permutes the vertices of G according to some criteria. Let $v_1, \ldots, v_n$ be the enumeration of $V(G)$ induced by said permutation, and $V_i = \{v_i, \ldots, v_n\}$ for $i \in [n]$. In practice this permutation has a large impact on the running time of the algorithm, but for our analysis the knowledge of the specific order is irrelevant.

In the main loop (lines 5–8) subgraphs of G are considered and at each iteration the size of a maximum clique containing only vertices of V_i is stored in $bounds[i]$. The algorithm keeps the best solution (largest clique) found so far in the global variable *incumbent* which is initially empty. The array *bounds* and the flag *found* are global variables. The current growing clique is stored in *solution* and passed as an argument of the subroutine `expand` together with the current subgraph $H \subseteq G$ being considered.

The main subroutine `expand` recursively goes through all vertices of H from smallest to largest index. First note that if the size of the current growing clique plus $|H|$ is not larger than the current maximum clique (line 13) then this branch can be cut. Moreover, if v_i is the smallest-index vertex in H then $V(H) \subseteq V_i$ and $bounds[i]$ is an upper bound on the size of a maximum clique in H. This implies that this branch can be cut if the size of the current growing clique plus $bounds[i]$ is not larger than the current maximum clique (line 15). If it is larger, the algorithm branches on the vertex v_i.

First v_i is taken to be part of the solution: it is added to (a copy of) the current growing solution, (a copy of) the graph is updated to contain only neighbours of v_i and a recursive call is made (lines 16–18). If the recursive call finds a clique larger than the current largest clique, it sets the flag *found* to true. This allows the algorithm can return to the main routine (line 8) since a maximum clique containing only vertices of V_i can be at most one unit larger than a maximum clique containing only vertices of V_{i+1}. If no larger clique was found, the algorithm then proceeds to the opposite branch choice, that is, taking vertex v_i to not be in the solution (line 20) and considering the next vertex in the ordering. If $V(H)$ is empty and a larger clique has been found, the best solution so far is updated and the flag *found* is set to true (lines 22–23).

Algorithm 2 `Cliquer`(G) algorithm

```
1  Cliquer(G):
2  begin
3      G ← permute(G)
4      incumbent ← ∅
5      for i = n down to 1 do
6          found ← false
7          expand(G[V_i ∩ N(v_i)], {v_i})
8          bounds[i] ← |incumbent|
9      return incumbent
10 expand(H, solution):
11 begin
12     while V(H) ≠ ∅ do
13         if |solution| + |V(H)| ≤ |incumbent| then return
14         i ← min{j | v_j ∈ V(H)}
15         if |solution| + bounds[i] ≤ |incumbent| then return
16         solution′ ← solution ∪ {v_i}
17         V′ ← V(H) ∩ N(v_i)
18         expand(H[V′], solution′)
19         if found = true then return
20         H ← H \ {v_i}
21     if |solution′| > |incumbent| then
22         incumbent ← solution′
23         found ← true
24     return
```

We now argue that the running time of the `Cliquer`(G) algorithm is bounded from below by the size of a regular resolution refutation of $Clique_{\text{block}}(G,k)$ up to a constant factor. First note that a straightforward modification of the `Cliquer`(G) algorithm gives an algorithm that determines whether G contains a block-respecting k-clique.

Given a graph G that does not contain a block-respecting k-clique, the last call of the subroutine `expand` in the main loop (lines 5–8, when i is set to 1) can be represented by an ordered decision tree with labelled leafs. A decision tree is said to be ordered if there exists a linear ordering of the variables such that if x is queried before y then $x \prec y$. In our setting, the order is determined by the permutation of the vertices, and without loss of generality we assume $v_i \prec v_j$ if $i < j$. For each leaf, if R is the set of vertices identified as clique members by the branch leading to this leaf, then the leaf is labelled either by a pair (u,v) such that $u,v \in R$ and there is no edge between u and v or by an index $\ell \in [k]$ such that all vertices in the ℓth block are outside the clique, or by a vertex v_i such that $i = \min\{j \mid v_j \in N(R)\}$ and the largest clique containing only vertices of V_i has size at most $k - |R| - 1$. For each vertex v_i that labels some leaf, we construct the decision tree corresponding to the ith call of the subroutine `expand`.

In order to weave these decision trees into a read-once branching program, at each leaf labelled v_i we query all non yet queried vertices v_j such that $j < i$ and v_j is in the same block as v_i. Let B_i denote the set of vertices. Observe that taking any vertex in B_i to be in the clique yields an immediate contradiction since $B_i \cap N(R) = \emptyset$ by definition of i. Moreover note that the branch leading to the leaf where all of B_i is taken to be outside the clique does not contain any query to

Algorithm 3 `MaxClique`(G) algorithm

```
1  MaxClique(G):
2  begin
3      global incumbent ← ∅
4      expand(G, ∅)
5      return incumbent
6  expand(H, solution):
7  begin
8      (order, bounds) ← colourOrder(H)
9      while V(H) ≠ ∅ do
10         i ← |V(H)|
11         if |solution| + bounds[i] ≤ |incumbent| then return
12         v ← order[i]
13         solution′ ← solution ∪ {v}
14         V′ ← V(H) ∩ N(v)
15         expand(H[V′], solution′)
16         H ← H \ {v}
17     if |solution′| > |incumbent| then incumbent ← solution′
18     return
```

vertices in V_i. We can therefore identify this leaf with the root of the decision tree corresponding to v_i and still maintain regularity. After repeating this procedure at every leaf labelled by some vertex, only leafs labelled by indices $\ell \in [k]$ and by pairs (u, v) remain, which have a direct correspondence to falsified clauses of $\mathit{Clique}_{\text{block}}(G, k)$. Therefore, the directed graph obtained by this process corresponds to a read-once branching program that solves the falsified clause search problem on $\mathit{Clique}_{\text{block}}(G, k)$ and the bound on the running time follows immediately.

Colour-based branch and bound algorithms. We consider a class of algorithms which are arguably the most successful in practice. An extended survey together with a computational analysis of algorithms published until 2012 can be found in [44] and an overview of algorithms reported since then in [37]. These algorithms are branch and bound algorithms that use colouring as a bounding—and often also as a branching—strategy. The basic idea is that if a graph can be coloured with ℓ colours then it does not contain a clique larger than ℓ.

The `MaxClique`(G) algorithm described in Figure 3, a generalized version of Algorithm 2.1 in [37], is a basic maximum clique algorithm which uses a colour-based branch and bound strategy. The algorithm keeps the best solution (largest clique) found so far in the global variable *incumbent* which is initially empty. The current clique is stored in *solution* and passed as an argument of the subroutine `expand` together with the current subgraph $H \subseteq G$ being considered. The subroutine `colourOrder`(H) (line 8) returns an ordering of the vertices in H, say $v_1, v_2, \ldots, v_n$, and for every $i \in [n]$ an upper bound on the number of colours needed to colour the graph induced by vertices v_1 to v_i.

The vertices are then considered in reverse order. If the vertex v is being considered and the size of the current growing clique plus the (upper bound on the) number of colours needed to colour the remaining graph is not larger than the current maximum clique (line 11) then this branch can be cut. If it is larger, the algorithm branches on the vertex v. First v is taken to be part of the solution: v is added to (a copy of) the current growing solution, (a copy of) the graph is updated to contain only neighbours of v and a recursive call is made (lines 13–15). If the recursive call finds a clique larger than the current

largest clique, the best solution so far is updated (line 17). The algorithm proceeds to the opposite branch choice, that is, considering vertex v not in the solution (line 16). Returning to the loop the algorithm continues to consider the next vertex in the ordering.

It was reported in [12] that it is possible to capture the algorithms for solving the maximum clique problem in [13, 22, 29, 56–58] in a same framework. The general algorithm they present is an iterative version of the $\texttt{MaxClique}(G)$ algorithm. We observe that $\texttt{MaxClique}(G)$ captures also the more recent algorithms in [50–54, 59]. The differences in these algorithms reside in the colouring procedure and in how the graph operations are implemented (see [37, 44] for details). For our purpose, that is, in order to show that the running time of these algorithms can be bounded from below by the length of the shortest regular resolution refutation of the k-clique formula, we assume that the colouring algorithm and the graph operations take constant time and prove the lower bound for this general framework. Moreover, we can assume that optimal colouring bounds and optimal ordering of vertices are given.

We now argue that the running time of the $\texttt{MaxClique}(G)$ algorithm is bounded from below by the size of a regular resolution refutation of $\mathit{Clique}_{\text{block}}(G, k)$ up to a multiplicative factor of $2^k n^{\mathrm{O}(1)}$. We first note that a straightforward modification of the $\texttt{MaxClique}(G)$ algorithm gives an algorithm, which we refer to as $\texttt{Clique}(G, k)$, that determines whether G contains a k-clique. Given a graph G that does not contain a k-clique, an execution of $\texttt{Clique}(G, k)$ can be represented by a search tree with leafs labelled by a subgraph $H \subseteq G$ of potential clique-members and a number q such that the branch leading to this leaf has identified $k - q$ clique members, has not queried any vertex of H, and H is $(q-1)$-colourable. Note that a read-once branching program can simulate this search tree and, by Proposition 3.3 and the equivalence between read-once branching programs and regular resolution, at each leaf establish that H does not contain a q-clique in size at most $2^q \cdot q^2 \cdot |V(H)|^2$. The bound on the running time follows directly.

Observe that establishing that H does not contain a q-clique is done in a read-once fashion by querying only vertices of H. Since the vertices of H were not queried earlier on this branch, the whole branching program is read-once.

8 CONCLUDING REMARKS

In this paper we prove optimal average-case lower bounds for regular resolution proofs certifying k-clique-freeness of Erdős-Rényi graphs not containing k-cliques. These lower bounds are also strong enough to apply for several state-of-the-art clique algorithms used in practice.

The most immediate and compelling question arising from this work is whether the lower bounds for regular resolution can be strengthened to hold also for general resolution. A closer study of our proof reveals that there are several steps that rely on regularity. However, there is no connection per se between regular resolution and the abstract combinatorial property of graphs that we show to be sufficient to imply regular resolution lower bounds. Thus, it is tempting to speculate that this property, or perhaps some modification of it, might be sufficient to obtain lower bounds also for general resolution. If so, a natural next step would be to try to extend the lower bound further to the polynomial calculus proof system capturing Gröbner basis calculations. It is worth mentioning that proving a general resolution lower bound of $n^{\Omega(k)}$ for the k-clique formula would have interesting consequences in parameterized proof complexity [20].

Another intriguing question is whether the lower bounds we obtain asymptotically almost surely for random graphs can also be shown to hold deterministically under the weaker assumption that the graph has certain pseudorandom properties. Specifically, is it possible to get an $n^{\Omega(\log n)}$ length lower bound for the class of Ramsey graphs? A graph on n vertices is called *Ramsey* if it has no set of $\lceil 2 \log_2 n \rceil$ vertices forming a clique or an independent set. It is known that for sufficiently large n a random graph sampled from $\mathscr{G}(n, 1/2)$ is Ramsey with high probability. Is it true that for a Ramsey graph G on n vertices the formula $\mathit{Clique}(G, \lceil 2 \log_2 n \rceil)$ requires (regular) resolution refutations of length $n^{\Omega(\log n)}$? The main

difficulty towards adapting our argument to this setting is that Ramsey graphs are, in some sense, less well structured than random graphs. For example, a random graph plus a constant number of isolated vertices is, with high probability, still a Ramsey graph, but it no longer satisfies the first property of clique-denseness (Definition 5.3). This particular problem can be circumvented using a result from [43, Theorem 1]—as was done in [34] to obtain a lower bound for tree-like resolution—but proving that a Ramsey graph satisfies the second property of clique-denseness, or some suitable version of it, seems significantly more challenging.

ACKNOWLEDGMENTS

This work has been a long journey, and different subsets of the authors want to acknowledge fruitful and enlightening discussions with different subsets of Christoph Berkholz, Olaf Beyersdorff, Nicola Galesi, Ciaran McCreesh, Toni Pitassi, Pavel Pudlák, Ben Rossman, Navid Talebanfard, and Neil Thapen. A special thanks to Shuo Pang for having pointed out an inaccuracy in the probabilistic argument in Section 6 and having suggested a fix.

The first, second, and fourth authors were supported by the European Research Council under the European Union's Horizon 2020 Research and Innovation Programme / ERC grant agreement no. 648276 AUTAR.

The third and fifth authors were supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611 as well as by Swedish Research Council grants 621-2012-5645 and 2016-00782, and the second author did part of this work while at KTH Royal Institute of Technology supported by the same grants. The last author was supported by the Russian Foundation for Basic Research.

REFERENCES

[1] Noga Alon, Michael Krivelevich, and Benny Sudakov. 1998. Finding a large hidden clique in a random graph. *Random Structures and Algorithms* 13, 3-4 (1998), 457–466.

[2] Albert Atserias, Ilario Bonacina, Susanna F. de Rezende, Massimo Lauria, Jakob Nordström, and Alexander Razborov. 2018. Clique Is Hard on Average for Regular Resolution. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18)*. 866–877.

[3] Roberto J. Bayardo Jr. and Robert Schrag. 1997. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*. 203–208.

[4] Paul Beame, Chris Beck, and Russell Impagliazzo. 2016. Time-Space Tradeoffs in Resolution: Superpolynomial Lower Bounds for Superlinear Space. *SIAM J. Comput.* 45, 4 (Aug. 2016), 1612–1645. Preliminary version in *STOC '12*.

[5] Paul Beame, Russell Impagliazzo, and Ashish Sabharwal. 2007. The Resolution Complexity of Independent Sets and Vertex Covers in Random Graphs. *Computational Complexity* 16, 3 (Oct. 2007), 245–297. Preliminary version in *CCC '01*.

[6] Paul Beame and Toniann Pitassi. 1996. Simplified and Improved Resolution Lower Bounds. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS '96)*. 274–282.

[7] Eli Ben-Sasson and Avi Wigderson. 2001. Short Proofs are Narrow—Resolution Made Simple. *J. ACM* 48, 2 (March 2001), 149–169. Preliminary version in *STOC '99*.

[8] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. 2013. Parameterized Complexity of DPLL Search Procedures. *ACM Transactions on Computational Logic* 14, 3 (Aug. 2013), 20:1–20:21. Preliminary version in *SAT '11*.

[9] Olaf Beyersdorff, Nicola Galesi, Massimo Lauria, and Alexander A. Razborov. 2012. Parameterized Bounded-Depth Frege Is not Optimal. *ACM Transactions on Computation Theory* 4, 3 (Sept. 2012), 7:1–7:16. Preliminary version in *ICALP '11*.

[10] Béla Bollobás and Paul Erdős. 1976. Cliques in random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society* 80, 3 (Nov. 1976), 419–427.

[11] Coen Bron and Joep Kerbosch. 1973. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* 16, 9 (Sept. 1973), 575–577.

[12] Renato Carmo and Alexandre Züge. 2012. Branch and bound algorithms for the maximum clique problem under a unified framework. *Journal of the Brazilian Computer Society* 18, 2 (June 2012), 137–151.

[13] Randy Carraghan and Panos M. Pardalos. 1990. An exact algorithm for the maximum clique problem. *Operations Research Letters* 9, 6 (Nov. 1990), 375–382.

[14] Bruno Pasqualotto Cavalar, Mrinal Kumar, and Benjamin Rossman. 2020. Monotone Circuit Lower Bounds from Robust Sunflowers. In *Proceedings of the 14th Latin American Symposium on Theoretical Informatics (LATIN '20) (Lecture Notes in Computer Science, Vol. 12118)*. 311–322.

[15] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. 2004. Linear FPT Reductions and Computational Lower Bounds. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC '04)*. 212–221.

[16] Stephen A. Cook and Robert Reckhow. 1979. The Relative Efficiency of Propositional Proof Systems. *Journal of Symbolic Logic* 44, 1 (March 1979), 36–50. Preliminary version in *STOC '74*.

[17] Stefan Dantchev, Nicola Galesi, Abdul Ghani, and Barnaby Martin. 2020. *Proof complexity and the binary encoding of combinatorial principles*. Technical Report 2008.02138. arXiv.org. Preliminary versions of these results appeared in [18] and [19].

[18] Stefan Dantchev, Nicola Galesi, and Barnaby Martin. 2019. Resolution and the Binary Encoding of Combinatorial Principles. In *Proceedings of the 34th Computational Complexity Conference (CCC '19) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 137)*. 6:1–6:25.

[19] Stefan Dantchev, Abdul Ghani, and Barnaby Martin. 2020. Sherali-Adams and the Binary Encoding of Combinatorial Principles. In *Proceedings of the 14th Latin American Symposium on Theoretical Informatics (LATIN '20) (Lecture Notes in Computer Science, Vol. 12118)*. 336–347.

[20] Stefan Dantchev, Barnaby Martin, and Stefan Szeider. 2011. Parameterized Proof Complexity. *Computational Complexity* 20, 1 (March 2011), 51–85. Preliminary version in *FOCS '07*.

[21] Rodney Downey and Michael R. Fellows. 1995. Fixed-Parameter Tractability and Completeness II: Completeness for W[1]. *Theoretical Computer Science A* 141, 1–2 (April 1995), 109–131.

[22] Torsten Fahle. 2002. Simple and Fast: Improving a Branch-And-Bound Algorithm for Maximum Clique. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA '02) (Lecture Notes in Computer Science, Vol. 2461)*. 485–498.

[23] Armin Haken. 1985. The Intractability of Resolution. *Theoretical Computer Science* 39, 2-3 (Aug. 1985), 297–308.

[24] Johan Håstad. 1999. Clique is Hard to Approximate within $n^{1-\epsilon}$. *Acta Mathematica* 182 (1999), 105–142. Preliminary version in *FOCS '96*.

[25] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k-SAT. *J. Comput. System Sci.* 62, 2 (March 2001), 367–375. Preliminary version in *CCC '99*.

[26] Richard M. Karp. 1972. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*. Springer, 85–103.

[27] Richard M. Karp. 1976. The probabilistic analysis of some combinatorial search algorithms. In *Algorithms and Complexity: New Directions and Recent Results*. Academic Press, New York, 1–19.

[28] Donald E. Knuth. 1994. The sandwich theorem. *The Electronic Journal of Combinatorics* 1, A1 (1994), 1–48.

[29] Janez Konc and Dušanka Janežič. 2007. An improved branch and bound algorithm for the maximum clique problem. *MATCH Communications in Mathematical and Computer Chemistry* 58 (2007), 569–590.

[30] Jan Krajíček. 1997. Interpolation Theorems, Lower Bounds for Proof Systems, and Independence Results for Bounded Arithmetic. *Journal of Symbolic Logic* 62, 2 (June 1997), 457–486.

[31] Jan Krajíček. 1995. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge University Press, New York.

[32] Luděk Kučera. 1995. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics* 57, 2-3 (Feb. 1995), 193–212.

[33] Massimo Lauria. 2018. Cliques enumeration and tree-like resolution proofs. *Inform. Process. Lett.* 135 (July 2018), 62–67.

[34] Massimo Lauria, Pavel Pudlák, Vojtěch Rödl, and Neil Thapen. 2017. The Complexity of Proving That a Graph is Ramsey. *Combinatorica* 37, 2 (April 2017), 253–268. Preliminary version in *ICALP '13*.

[35] László Lovász. 1979. On the Shannon capacity of a graph. *IEEE Transactions on Information theory* 25, 1 (Jan. 1979), 1–7.

[36] João P. Marques-Silva and Karem A. Sakallah. 1999. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Comput.* 48, 5 (May 1999), 506–521. Preliminary version in *ICCAD '96*.

[37] Ciaran McCreesh. 2017. *Solving Hard Subgraph Problems in Parallel*. Ph.D. Dissertation. University of Glasgow.

[38] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. 2001. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*. 530–535.

[39] Jaroslav Nešetřil and Svatopluk Poljak. 1985. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae* 026, 2 (1985), 415–419.

[40] Sampo Niskanen and Patric R. J. Östergård. 2003. *Cliquer User's Guide, Version 1.0*. Technical Report T48. Communications Laboratory, Helsinki University of Technology, Espoo, Finland.

[41] Patric R. J. Östergård. 2002. A Fast Algorithm for the Maximum Clique Problem. *Discrete Applied Mathematics* 120, 1–3 (Aug. 2002), 197–207.

[42] Shuo Pang. 2019. *Large Clique is Hard on Average for Resolution*. Technical Report TR19-068. Electronic Colloquium on Computational Complexity (ECCC). To appear in Proceedings of the CSR '21.

[43] Hans Jürgen Prömel and Vojtěch Rödl. 1999. Non-Ramsey Graphs Are $c \log n$-Universal. *Journal of Combinatorial Theory, Series A* 88, 2 (Nov. 1999), 379–384.

[44] Patrick Prosser. 2012. Exact Algorithms for Maximum Clique: A Computational Study. *Algorithms* 5, 4 (Nov. 2012), 545–587.

[45] Pavel Pudlák. 1997. Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations. *Journal of Symbolic Logic* 62, 3 (Sept. 1997), 981–998.

[46] Alexander Razborov, Avi Wigderson, and Andrew Yao. 2002. Read-Once Branching Programs, Rectangular Proofs of the Pigeonhole Principle and the Transversal Calculus. *Combinatorica* 22, 4 (Oct. 2002), 555–574. Preliminary version in *STOC '97*.

[47] Benjamin Rossman. 2008. On the Constant-Depth Complexity of k-Clique. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*. 721–730.

[48] Benjamin Rossman. 2010. *Average-Case Complexity of Detecting Cliques*. Ph.D. Dissertation. Masschussets Institute of Technology.

[49] Benjamin Rossman. 2014. The Monotone Complexity of k-Clique on Random Graphs. *SIAM J. Comput.* 43, 1 (Jan. 2014), 256–279. Preliminary version in *FOCS '10*.

[50] Pablo San Segundo, Alvaro Lopez, and Mikhail Batsyn. 2014. Initial Sorting of Vertices in the Maximum Clique Problem Reviewed. In *Proceedings of the 8th International Conference on Learning and Intelligent Optimization (LION '14), Selected Revised Papers (Lecture Notes in Computer Science, Vol. 8426)*. Springer, 111–120.

[51] Pablo San Segundo, Alvaro Lopez, Mikhail Batsyn, Alexey Nikolaev, and Panos M. Pardalos. 2016. Improved Initial Vertex Ordering for Exact Maximum Clique Search. *Applied Intelligence* 45, 3 (Oct. 2016), 868–880.

[52] Pablo San Segundo, Fernando Matia, Diego Rodríguez-Losada, and Miguel Hernando. 2013. An improved bit parallel exact maximum clique algorithm. *Optimization Letters* 7, 3 (March 2013), 467–479.

[53] Pablo San Segundo, Diego Rodríguez-Losada, and Agustín Jiménez. 2011. An Exact Bit-parallel Algorithm for the Maximum Clique Problem. *Computers and Operations Research* 38, 2 (Feb. 2011), 571–581.

[54] Pablo San Segundo and Cristóbal Tapia. 2010. A new implicit branching strategy for exact maximum clique. In *Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI '10)*, Vol. 1. 352–357.

[55] W. A. Stein et al. 2017. *Sage Mathematics Software (Version 8.1)*. The Sage Development Team.

[56] Etsuji Tomita and Toshikatsu Kameda. 2007. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *Journal of Global Optimization* 37, 1 (Jan. 2007), 95–111.

[57] Etsuji Tomita and Tomokazu Seki. 2003. An Efficient Branch-and-bound Algorithm for Finding a Maximum Clique. In *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science (DMTCS '03)*, Vol. 3. 278–289.

[58] Etsuji Tomita, Yoichi Sutani, Takanori Higashi, Shinya Takahashi, and Mitsuo Wakatsuki. 2010. A simple and faster branch-and-bound algorithm for finding a maximum clique. In *Proceedings of the 4th International Workshop on Algorithms and Computation (WALCOM '10) (Lecture Notes in Computer Science, Vol. 5942)*. Springer, 191–203.

[59] Etsuji Tomita, Kohei Yoshida, Takuro Hatta, Atsuki Nagao, Hiro Ito, and Mitsuo Wakatsuki. 2016. A Much Faster Branch-and-Bound Algorithm for Finding a Maximum Clique. In *Proceedings of the 10th International Workshop on Frontiers in Algorithmics (FAW '16) (Lecture Notes in Computer Science, Vol. 9711)*. Springer, 215–226.

[60] Virginia Vassilevska. 2009. Efficient Algorithms for Clique Problems. *Inform. Process. Lett.* 109, 4 (Jan. 2009), 254–257.

[61] David R. Wood. 1997. An Algorithm for Finding a Maximum Clique in a Graph. *Operations Research Letters* 21, 5 (Jan. 1997), 211–217.

[62] David Zuckerman. 2007. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory of Computing* 3, 6 (Aug. 2007), 103–128. Preliminary version in *STOC '06*.