Cliques enumeration and tree-like resolution proofs

Massimo Lauria
massimo.lauria@uniroma1.it
Dipartimento di Scienze Statistiche - Sapienza Università di Roma, Italy

Abstract

We show the close connection between the enumeration of cliques in a k-clique free graph G and the length of tree-like resolution refutations for formula $\mathrm{Clique}(G,k)$, which claims that G has a k-clique. The length of any such tree-like refutation is within a "fixed parameter tractable" factor from the number of cliques in the graph. We then proceed to drastically simplify the proofs of the lower bounds for the length of tree-like resolution refutations of $\mathrm{Clique}(G,k)$ shown in [Beyersdorff et at. 2013, Lauria et al. 2017], which now reduce to a simple estimate of the number of cliques.

Keywords: Proof Complexity, Clique, Resolution, Decision Tree

1. Introduction

The k-clique problem asks whether a graph of n vertices has a set of k pairwise connected vertices, i.e., a *k-clique*. A exhaustive search can decide that in time roughly n^k. Being one of the standard NP-complete problems [26] an efficient solution seems very unlikely, even in approximation [24]. It is actually possible to slightly improve on the brute force approach [38]. Nevertheless all known algorithms run in time $n^{\Omega(k)}$. Is this the best we can hope for? If one believes the Exponential Time Hypothesis [25] we cannot even get to $n^{o(k)}$ [29]. Since the problem seems so difficult, we would like to prove its hardness without using unproved hypotheses. We cannot do that in general, but we can if we restrict the computational model. For example we know that circuits that solve k-clique must have size $n^{\Omega(k)}$ if they are restricted to be either of bounded depth [35] or monotone [36]. These results even apply to the relaxed requirement of just solving k-clique asymptotically almost surely in the Erdős-Renyi model, where the graph is random in the sense that edges are picked (independently) at random according to the appropriate density.

To determine which computational model to study, we first observe that an algorithm that search for k-cliques in a k-clique free graph G produces a refutation the propositional formula $\mathrm{Clique}(G,k)$, which falsely claims that G has a clique of size k. The trace of the algorithm itself is an efficiently verifiable proof that $\mathrm{Clique}(G,k)$ is unsatisfiable, a proof of length proportional to the length of the trace. The key observation here is that a simple algorithm produces a proof that can be written down in a simple language. If the language is simple enough we may manage to show strong lower

bounds on the length of such proof, and therefore to show lower bounds to the running time of the algorithm. We focus on how the proof is written down, hence we do not really care about how the algorithm finds it. A lot of those details can be abstracted away up to the point that we can see the proof as the result of a non-deterministic, i.e., "very lucky", proof search. Concretely, it means that a lower bound proved in this framework applies to a wide family of algorithms, rather than a specific one. This is one of the highlights of studying proofs.

The field of *proof complexity* [14, 6, 37] studies the length of proofs of propositional unsatisfiability. The languages in which such proofs are expressed are called *proof systems*. The stronger the proof system, more general is the class of algorithms that such system captures. Resolution is definitely the most famous one. It underlies state-of-the-art SAT algorithms [4, 30, 31] and it is the object of much study. The first proof length lower bound for resolution was proved in [23], followed by more general lower bound techniques [27, 8]. Memory is a precious resource in practical SAT solving, as much as time. In the same way proof length models running time, [19, 1] develop a notion of proof space that is supposed to model memory usage. Another important proof complexity parameter of resolution proofs is the "width", i.e., the size of each proof line. Estimating the required width of a resolution proof is a proxy to estimate the required length [8] or the required space [3, 21]. The correspondence between width and space is not very tight [32], which led to the study of resolution space using various pebbling games. Pebbling games model the memory allocation of a computation. While it is hard to know how much memory is needed to win these games in general [22, 13], it is possible to use and combine well understood cases of these games in order to prove resolution space lower bounds and resolution length vs resolution space trade-offs results [33].

We now go back to the problem of determining the resolution proof complexity of $\mathrm{Clique}(G, k)$. For $k \approx n$, super polynomial lower bounds have been proved [5] using the size-width relation from [8], which is by now a standard technique in proof complexity. For $k \ll n$ the problem was raised in [11] and it is still open. Neither the size-width relation nor interpolation [27], which is the other main tool of resolution proof complexity, give anything useful for $k \ll n$. This problem seems difficult, and new techniques may be necessary to solve it.

In this paper we focus on the *tree-like resolution*, which is a restricted form of resolution that captures DPLL style algorithms [16, 15] and decision trees, namely algorithms that explore the space of solutions by guessing and backtracking. Tree-like resolution is weaker than general resolution [12], but we understand better its proof length and space [9, 20]. There are three types of graphs for which the clique formula requires tree-like resolution refutations of length $n^{\Omega(k)}$. In [10] the lower bound is proved for the complete $(k-1)$-partite graph, as well as for Erdős-Renyi random graph with appropriate edge density. Tree-like resolution needs refutations of length $n^{\Omega(\log(n))}$ for Ramseys graph as well, i.e., graphs of n vertices that have neither a $2\log(n)$-clique nor a $2\log(n)$ independent set, when $k = 2\log(n)$ [28]. This is the state of the art so far. We know that the complete $(k-1)$-partite graph is easy for stronger subsystem of resolution [10], while the Erdős-Renyi and the Ramsey graphs may as be hard for general resolution too, as far as we know. Here we do not improve on the state of the art, but we drastically simplifies the three lower bounds mentioned above, by showing a connection

between the length of tree-like resolution refutations of $\mathrm{Clique}(G, k)$ and the number of cliques in G.

The paper is organized as follows. In Section 2 we give necessary definitions and notations for the objects of this paper. In Section 3 we show the close correspondence between the number of cliques in a graph and the size of tree-like resolution refutations for the clique formulas on that graph. In Section 4 we show classes of k-clique free graphs with many cliques, hence they need large refutations for the corresponding clique formulas.

2. Preliminaries

In this paper we consider simple undirected loop-less graphs $G = (V, E)$. We write $\Gamma(v)$ to denote the set of vertices in V that are neighbors of a vertex $v \in V$. For an arbitrary set of vertices $U \subseteq V$ we denote as $\Gamma(U)$ the set of vertices $\bigcap_{u \in U} \Gamma(u)$, i.e., the common neighbors of U in G. A clique $K \subseteq V$ of G is a set of vertices so that there is an edge between any two of them. We denote as $\mathcal{C}(G)$ the set of cliques of G. We denote as $[m]$ the set of integers from 1 to m.

A CNF formula over a set of variables is a conjunction of distinct clauses, each of them being the disjunction of distinct literals. A literal is either an occurrence of a variable or its negation. D is a subclause of a clause C when D is a disjunction of literals contained in C. We indicate that D is a subclause of C with notation $C \subseteq D$.

The *k-clique formula* $\mathrm{Clique}(G, k)$ is a CNF formula over variables $x_{i,v}$ for every $v \in V$ and $i \in [k]$, where the boolean variable $x_{i,v}$ indicates whether the i-th vertex of the clique is v. The formula is the conjunction of clauses

$$\bigvee_{v \in V} x_{i,v} \qquad \forall i \in [k] \,, \tag{1a}$$

$$\neg x_{i,u} \vee \neg x_{j,v} \qquad \forall i, j \in [k], i \neq j, \forall u, v \in V, \{u, v\} \notin E \,, \tag{1b}$$

$$\neg x_{i,u} \vee \neg x_{i,v} \qquad \forall i \in [k], \forall u, v \in V, u \neq v \,. \tag{1c}$$

Clauses (1a) are called clique axioms, clauses (1b) are called edge axioms, and clauses (1c) are called functionality axioms. Clearly $\mathrm{Clique}(G, k)$ is satisfiable if and only if G contains a clique of k vertices, and this holds even without the functionality axioms.

Tree-like resolution and Decision trees. The *resolution rule* is an inference rule that allows to logically derive a clause from two clauses, called premises, as follows

$$\frac{A \vee x \qquad B \vee \neg x}{A \vee B} \,. \tag{2}$$

To apply rule (2) the two premises must contain, respectively, the positive and negated literal of some variable x, and thus we say that we *resolve* the two clauses over variables x. The derived clause is their *resolvant*.

A *tree-like resolution proof* of a clause C from some CNF formula F is a rooted binary tree, directed from the leaves to the root, where each node in the tree is labeled by a clause over the variables of F. The clauses labeling the nodes in the proof must have the following properties:

- no clause contains both a literal and its negation;
- the clause labeling an internal node is the resolvant of the clauses labeling its two immediate predecessors;
- the clause at the root is a subclause of C;
- any clause labeling a leaf is a subclause of some clause of F.

A tree-like resolution *refutation* is a proof of the empty clauses. The *length* of a tree-like resolution proof is the number of its leaves. The *depth* of a tree-like resolution proof is the maximum length of a root-to-leaf path in it.[1]

A *decision tree* that computes a function over some variables $Y = \{y_1, \ldots, y_n\}$ with values in some set $\mathcal{R}$ is a rooted binary tree, directed from the root to the leaves. Every internal node is labeled by a *query*, i.e., some variable in Y. The two directed edges going out from an internal node are labeled by 0 and by 1 and correspond to the two possible answers to the query. Leaves are labeled by values in $\mathcal{R}$. Given an assignment $\rho : \{y_i\}_{i=1}^n \to \{0,1\}$, we define $\mathrm{path}(\rho)$ as follow: start at the root and whenever at an internal node ν labeled by a variable y_i, add to the path the outgoing edge (ν, ν') labeled by $\rho(y_i)$, and move to ν'. The process eventually ends up at a leaf, where it stops. The value of the function for the assignment ρ is the label of the leaf reached by $\mathrm{path}(\rho)$.

It is also useful to associate some partial assignment for the variables $y_1, \ldots, y_n$ to any directed path in the decision tree. A directed edge corresponds to a query y_i and an answer b, hence we associate it to the partial assignment $\{y_i \to b\}$. The partial assignment associated to a path is the union of the assignments associated to the edges in it. Since no variable occurs twice in any path, the assignment is well defined. For each node ν of the tree we denote as ρ_ν the partial assignment corresponding to the path from the root to ν.

In this paper we are interested in decision trees that solves the *search problem* for an unsatisfiable CNF formula. The search problem, given a formula F over variables Y and a total assignment ρ on them, asks to output a clause of F falsified by ρ. A decision tree that solves the search problem is a decision tree over the variables of F, where each leaf ℓ is labeled by a clause of F falsified by the partial assignment ρ_ℓ.

There is well know correspondence between tree-like resolution refutations and decision trees for the search problem. In particular for every decision tree T for the search problem over F there is a tree-like refutation of F with the same tree structure as T, but with the edges reversed from the leaves to the root [7]. The vice versa holds as well.

3. Cliques and tree-like proofs

We prove that the number of cliques in a k-clique free graph G is a lower bound to the length of tree-like refutations of $\mathrm{Clique}(G, k)$.

[1] This definition of tree-like resolution differs a bit from others in literature. Our definition does not need a weakening rule and it is still complete for clauses with no opposite literals.

Lemma 1. *Let G be a k-clique free graph. The length of any tree-like resolution refutation of* $\mathrm{Clique}(G,k)$ *is at least* $|\mathcal{C}(G)|$.

Proof. We fix a decision tree T for the search problem on $\mathrm{Clique}(G,k)$. Our goal is to show that T has at least $|\mathcal{C}(G)|$ leaves, and we do that by defining an injective mapping from the set of cliques of G to the set of leafs of T. For any fixed clique $K \in \mathcal{C}(G)$ we define the image of this mapping as follows: we walk through the decision tree from the root to some leaf ℓ_K and we define this to be the image of K. Such root-to-leaf path is completely specified by the answers to the queries of the decision tree, which are determined by K. The walk starts at the root of T. When the walk is at some node ν with query $x_{i,v}$ the answer is decided according to these rules:

- answer 0 if $v \notin K$; otherwise
- answer 0 if ρ_ν contains some $\{x_{j,w} \to 1\}$ where either $w = v$ or $j = i$; otherwise
- answer 1.

There is always a well defined answer for any query at any internal node of T, therefore the walk reaches one of the leaf nodes, which is set to be ℓ_K.

To see that the mapping is injective, we observe that the assignment ρ_{ℓ_K}, i.e., the assignment on the path from the root to ℓ_K, univocally identifies the clique K. For sake of notation we denote ρ_{ℓ_K} as ρ'. The assignment ρ' falsifies the clause of $\mathrm{Clique}(G,k)$ that labels ℓ_K, and since the answers to the queries do not violate neither edge axioms nor functional axioms, this clause must be a clique axiom. For some index i the assignment ρ' sets $\{x_{i,v} \to 0\}$ for all vertices v, and in particular for all vertices in K. Hence, the rules of the walk ensure that for each vertex w in K there is some index $j_w \neq i$ such that $\{x_{j_w,w} \to 1\} \in \rho'$, because otherwise the query $x_{i,w}$ would have been answered by 1 instead of 0. On the other hand for $w \notin K$ and $j \in [k]$ variable $x_{j,w}$ is either unassigned or 0. Concluding, the set of vertices w for which ρ_{ℓ_K} contains $\{x_{j,w} \to 1\}$ for some j is exactly the clique K itself. □

It would be nice to prove a converse result, namely to show it is possible to find a tree like refutation of length at most $\mathrm{poly}(|V(G)|) \cdot |\mathcal{C}(G)|$. This is not possible: consider a graph with n vertices which is made by the union of an isolated vertex and a complete graph of $n-1$ vertices. The $\mathrm{Clique}(G,n)$ formula for this graph is essentially a pigeonhole principle formula and therefore is hard for tree-like resolution and even for general resolution as well [23]. There are two possible solutions to this issue.

- The approach of [5] is to add *monotonicity axioms* to the formula to enforce that the k indexes point to vertices in an increasing fashion. Vertices of the graph are enumerated as $v_1, v_2, \ldots v_n$, and monotonicity axioms are

$$\neg x_{j_1,v_{i_1}} \vee \neg x_{j_2,v_{i_2}} \qquad \forall\, 1 \leq i_2 < i_1 \leq n, \quad \forall\, 1 \leq j_1 < j_2 \leq k\,. \tag{3}$$

- To consider values of $k \ll |V(G)|$ and threat clique as a parameterized problem [17].

In the latter case there is a tree-like resolution refutation of $\mathrm{Clique}(G,k)$ of length at most $f(k)\cdot n\cdot|\mathcal{C}(G)|$ for any k-clique free graph of n vertices.

Theorem 2. *Let G be a k-clique free graph over n vertices. There is a tree-like refutation of $\mathrm{Clique}(G,k)$ of length $f(k)\cdot n\cdot|\mathcal{C}(G)|$, for some function f.*

Proof. We argue the existence of such refutation by building the corresponding decision tree. Consider the order $v_1, v_2, \ldots, v_n$ of the vertices of G and fix $\mu = \emptyset$. We use μ to keep track of the partial mapping between $[k]$ and the vertices of the clique identified by the answers to the queries. The tree queries the variables x_{1,v_j} for j from 1 to n in order, until one query is answered 1. If none is, then the clique axiom for $i = 1$ is falsified. Otherwise we add $\{1 \to v_{j_1}\}$ to μ, where $x_{1,v_{j_1}}$ is the query that got 1 as answer. The tree repeats this process for $i > 1$ up to k, querying the variables x_{i,v_j} for j from 1 to n in order, until one of these variables in answered 1. If none is, then the i-th clique axiom is falsified. If instead some $x_{i,v_{j_i}}$ is answered 1 then either $v_{j_i} \notin \Gamma(\mathrm{rng}(\mu))$ and then we are at a leaf because an edge axiom has been falsified, or we add $\{i \to v_{j_i}\}$ to μ and continue with index $i+1$. At every node in the tree there is a corresponding value of μ which identifies a clique of G. In particular we can associate a specific value of μ for every leaf in the tree.

To bind the size of the tree observe that if the last mapping added to μ is $\{i \to v_{j_i}\}$, then the tree queries all variables x_{i+1,v_j}. The branch reaches a leaf as soon as the answers are all 0 or when one answer is 1 and violates an edge axiom. In all other cases μ gets extended. Hence there are at most $n+1$ leaves for each value of μ. How many values of μ occur in the branching? Recall that μ is a mapping from some indices in $[k]$ to a clique in G, and there at at most $|\mathcal{C}(G)|$ cliques in G. This observation gives the upper bound. □

4. Classes of graph with many cliques

Lemma 1 allows much easier proofs of the lower bounds for tree-like resolution shown in [10, 28]. The first example is the complete $(k-1)$-partite graph. For the sake of simplicity we assume n to be a multiple of $k-1$. The $(k-1)$-partite graph over n vertices is made by $(k-1)$ blocks of $\frac{n}{k-1}$ vertices each. The graph has no edges within a block and an edge between any two vertices in different blocks. The graph has obviously $n^{\Omega(k)}$ cliques and next theorem follows immediately from Lemma 1.

Theorem 3 ([10])**.** *Let G be the complete $(k-1)$-partite over n vertices. Any tree-like resolution refutation for $\mathrm{Clique}(G,k)$ has length $n^{\Omega(k)}$.*

While we know that $(k-1)$-partite graphs are easy for resolution, there are graphs for which the current state of the art is an $n^{\Omega(k)}$ lower bound for tree-like resolution even if it is likely that the bound holds in general resolution too. The most interesting example is the random graph $\mathcal{G}(n,p)$ where $p = n^{-(1+\epsilon)\frac{2}{k-1}}$ for any $\epsilon > 0$. For constant k the graph $G \sim \mathcal{G}(n,p)$ has no k-clique with high probability by Markov inequality, because the expected number of them is

$$\binom{n}{k}\left(n^{-(1+\epsilon)\frac{2}{k-1}}\right)^{\binom{k}{2}} < n^k n^{-(1+\epsilon)k} = \frac{1}{n^{\epsilon k}}\ . \tag{4}$$

Still, any tree-like resolution refutation of $\mathrm{Clique}(G,k)$ has length $n^{\Omega(k)}$ with high probability.

Theorem 4 ([10]). *For any constant $\epsilon > 0$, let $G \sim \mathcal{G}(n,p)$ where $p = n^{-(1+\epsilon)\frac{2}{k-1}}$. Graph G has no k-clique with high probability, and still any tree-like resolution refutation of $\mathrm{Clique}(G,k)$ has length $n^{\Omega(k)}$.*

For $p = 1/2$ the right value of k for which the problem is interesting is $2\log(n)+2$. For $G \sim \mathcal{G}(n,1/2)$ the expected number of k-clique where $k = 2\log(n)+2$ is at most

$$\binom{n}{k}\left(\frac{1}{2}\right)^{\binom{k}{2}} < \frac{1}{n} \tag{5}$$

therefore by Markov inequality with probability $1-o(1)$ graph G has no k-clique.

Theorem 5 ([10]). *Let $G \sim \mathcal{G}(n,1/2)$. With probability $1-o(1)$ graph G has no $(2\log(n)+2)$-clique, and still any tree-like resolution refutation of $\mathrm{Clique}(G, 2\log(n)+2)$ has length $n^{\Omega(\log(n))}$.*

Both Theorems 4 and 5 can be easily proved using our Lemma 1. To lower bound the number of cliques we use the following extension lemma. Various versions of this extension lemma exist in literature, e.g., in [10, 28]. We formulate one that applies to our two settings of parameters.

Lemma 6. *Let $G \sim \mathcal{G}(n,p)$ with $p \leq \frac{1}{2}$. With high probability $1-o(1)$ over the sampling of G, G has $n^{\Omega(-\frac{\log n}{\log p})}$ distinct cliques.*

Proof. We want to find many different cliques in G.

First we show that with high probability an arbitrary sets of $\Omega\left(-\frac{\log n}{\log p}\right)$ vertices has a polynomial number of common neighbors. We fix a "canonical" clique size $k' = -\frac{\log n}{2\log p}$, which is at most $\frac{\log n}{2}$, and we consider $|\Gamma(R)|$ for an arbitrary set $R \subseteq V(G)$ with $|R| < k'$. For a fixed set of vertices R of size less than k' we consider the random variables X_v for $v \in V(G)\setminus R$, where X_v is 1 if $v \in \Gamma(R)$ and 0 otherwise. Randomness of X_v is with respect of the sampling of G. These are $n-k' = n(1-o(1))$ independent Bernoulli variables with expectation $p^{k'} = n^{-1/2}$. Therefore the expectation of $|\Gamma(R)|$ is $\sqrt{n}\,(1-o(1))$ The probability that $|\Gamma(R)| < \sqrt{n}/2$ is, by Chernoff Bound [18, Theorem 1.1], at most

$$\exp\left(\frac{-(1-o(1))\sqrt{n}}{8}\right). \tag{6}$$

Since there are at most $n^{O(\log n)}$ possible sets R of size less than k', by union bound we get that all such R have, simultaneously, at least $\sqrt{n}/2$ common neighbors with probability $1-o(1)$ with respect to the sampling of G.

Now we can easily show that there must be many "ordered" cliques: pick a sequence of vertices $v_1, v_2, v_3, \ldots$ so that each $v_i \in \Gamma(\{v_1,\ldots,v_{i-1}\})$, until the sequence cannot be extended anymore. By the choice of each v_i in the sequence there are at least $\sqrt{n}/2$ vertices, as long as $i \leq k'$, therefore we produce $n^{\Omega(k')}$ distinct ordered cliques.

Different sequences could potentially correspond to the same clique, therefore we must factor out the order of the sequence. With probability $1-o(1)$ with respect to the

sampling of G the latter has no clique larger than $4k'$, and therefore no clique corresponds to more than $4k'!$ sequences. There must at least $n^{\Omega(k')}$ distinct cliques in the end.

To bound the size of the maximum clique is the graph we use the standard first moment method. The expected number of cliques of size $4k'$ in G is

$$\binom{n}{4k'} p^{\binom{4k'}{2}} \approx \frac{1}{n^{\Omega(k)}} \tag{7}$$

hence by Markov inequality the probability that there is one is $o(1)$. □

Previous lemma and Lemma 1 immediately imply Theorems 4 and 5, by setting $p = n^{-(1+\epsilon)\frac{2}{k-1}}$ and $p = \frac{1}{2}$ respectively.

A generalization of the Theorem 5, discussed in [28], is the problem of witnessing that a graph is c-Ramsey, i.e., that it has neither a $c\log(n)$-clique nor $c\log(n)$-independent set. The key result there it is hard to witness even the absence of a $c\log(n)$-clique from any c-Ramsey graph.

Theorem 7 ([28])**.** *Let G be a c-Ramsey graphs on n vertices. Any tree-like resolution refutation of* $\mathrm{Clique}(G, c\log(n))$ *has length* $n^{\Omega(\log(n))}$.

This theorem is a generalization of Theorem 5 because a random graph has neither an independent set nor a clique of size $2\log(n)+2$, with high probability. Therefore it is c-Ramsey for any $c > 2$. To prove the theorem we will need a corollary of the following lemma about the edge density of Ramsey graphs. For any two sets of non empty disjoint sets of vertices A and B in a graph G, the *edge density* $d(A,B)$ is the ratio between the number of edges with one end in A and the other in B, and $|A|\cdot|B|$. It turns out that in most of a c-Ramsey graph the edge density must be balanced.

Lemma 8 ([34])**.** *There exist constants $\beta > 0$, $\delta > 0$ such that if G is a c-Ramsey graph, then there is a set $S \subseteq V(G)$ with $|S| \geq n^{\frac{3}{4}}$ such that, for all $A, B \subseteq S$, if $|A|, |B| \geq |S|^{1-\beta}$ then $\delta \leq d(A,B) \leq 1-\delta$.*

Corollary 9. *There exist constants $\beta > 0$, $\delta' > 0$ such that if G is a c-Ramsey graph, then there is a set $S \subseteq V(G)$ with $|S| \geq n^{\frac{3}{4}}$ such that, for all $A \subseteq S$ of size at least $2|S|^{1-\beta}$, at least a quarter of vertices $v \in A$ have $|\Gamma(v) \cap A| \geq \delta'|A|$.*

Proof. Fix S, β and δ as in Lemma 8. Consider a set $A \subseteq S$ of size $2|S|^{1-\beta}$, and split arbitrarily A in half as two disjoint parts of the same size $A_0 \dot\cup A_1$. By Lemma 8 $d(A_0, A_1) \geq \delta$ therefore by Markov inequality at least half of the vertices in A_0, i.e., a quarter of A, must have $\delta|A_1|$ neighbors in A_1, and therefore we can fix $\delta' = \delta/2$. □

Proof of Theorem 7. By the Lemma 1 we just need to show that there are $n^{\Omega(\log(n))}$ cliques in a c-Ramsey graph. We pick a sequence of vertices $v_1, v_2, v_3, \ldots$ so that each $v_i \in \Gamma(\{v_1, \ldots, v_{i-1}\})$ to identify a clique. We will show that there are at least $n^{\Theta(1)}$ choices for each new vertex v_i, given $\{v_1, \ldots, v_{i-1}\}$, and that the sequence can be extended for at least $\Omega(\log(n))$ steps.

Using Corollary 9 we can pick an initial set of vertices V_1 of size $n^{3/4}$, and constants $\beta > 0$ and $\delta' > 0$ as stated in the corollary itself. We fix $t = \lfloor \frac{\log(n^{3\beta/4})}{\log 1/\delta'} \rfloor$, so that $n^{3/4} \cdot \delta'^t \geq n^{3(1-\beta)/4}$. Observe that $t = \Omega(\log n)$.

For $1 \leq i \leq t$ we pick some $v_i \in V_i$ with $|\Gamma(v_i) \cap V_i| \geq \delta'|V_i|$, and we fix V_{i+1} to be $\Gamma(v_i) \cap V_i$. This is possible because we start with $|V_1| = n^{3/4}$ and at each step $i \leq t$ we can apply Corollary 9 to ensure that $|V_{i+1}| \geq n^{3/4} \cdot (\delta')^{i+1}$, given that $|V_i| \geq n^{3/4} \cdot (\delta')^i \geq n^{3(1-\beta)/4}$.

The choice of each v_i is made among $|V_i|/4 = n^{\Theta(1)}$ vertices therefore we have $n^{\Omega(t)} = n^{\Omega(\log n)}$ ordered cliques of length $c\log(n)$. At most $c\log(n)!$ ordered cliques correspond to a clique in the graph, hence the number of cliques in G is $n^{\Omega(\log(n))}$. An application of Lemma 1 concludes the proof. □

Conclusion

We have discussed the connection between DPLL algorithms for k-clique and the number of cliques in a graph. In particular we have shown that the number of cliques in a k-clique free graph G is a lower bound for the running time of any DPLL algorithm for the k-clique problem on G, or equivalently for the length of any tree-like resolution refutation of the $\mathrm{Clique}(G, k)$ formula. Furthermore we have shown that this is also an upper bound in the parameterized complexity framework. Using these results we reprove all known lower bounds for tree-like resolution refutation of $\mathrm{Clique}(G, k)$ formula, just by estimating the number of cliques.

Of course the major open problem left open in this work, and first mentioned in [11], is to extend some of the lower bounds to general resolution. It turns out that resolution can refute k-clique easily on $(k-1)$-colorable graphs, namely in length $2^k\mathrm{poly}(n)$, therefore the correspondence between proof length and number of cliques does not hold there [10]. In a recent (unpublished) paper some of the results here have been generalized to regular resolution, which is a subsystem of resolution vastly more powerful than tree-like resolution [2].

During part of this work the author was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ERC-2014-CoG 648276 AUTAR). This work was partially done during the Special Semester on Computational and Proof Complexity organized at the Chebyshev Laboratory of St.Petersburg, in the Spring of 2016. Special thanks go to Nicola Galesi and Olaf Beyersdorff for the many conversations and joint works about this problem.

References

[1] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002.

[2] Albert Atserias, Ilario Bonacina, Susanna F. de Rezende, Massimo Lauria, and Alexander A. Razborov. Clique is hard on average for regular resolution. Submitted, 2017.

[3] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008.

[4] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[5] Paul Beame, Russell Impagliazzo, and Ashish Sabharwal. The resolution complexity of independent sets and vertex covers in random graphs. *Comput. Complex.*, 16(3):245–297, 2007.

[6] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present, and future. In *Current Trends in Theoretical Computer Science*, pages 42–70. World Scientific Publishing, 2001.

[7] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.

[8] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001.

[9] O. Beyersdorff, N. Galesi, and M. Lauria. A characterization of tree-like resolution size. *Information Processing Letters*, 113(18):666–671, 2013.

[10] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. Parameterized complexity of dpll search procedures. *ACM Transactions on Computational Logic (TOCL)*, 14(3):20, 2013.

[11] Olaf Beyersdorff, Nicola Galesi, Massimo Lauria, and Alexander A. Razborov. Parameterized bounded-depth frege is not optimal. *ACM Trans. Comput. Theory*, 4(3):7:1–7:16, September 2012.

[12] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001.

[13] Siu Man Chan, Massimo Lauria, Jakob Nordström, and Marc Vinyals. Hardness of approximation in pspace and separation results for pebble games. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 466–485, Washington, DC, USA, 2015. IEEE Computer Society.

[14] Stephen A. Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, March 1979.

[15] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5:394–397, July 1962.

[16] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7:201–215, July 1960.

[17] R. Downey and M. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag, 2013.

[18] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[19] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001. Preliminary versions of these results appeared in *STACS '99* and *CSL '99*.

[20] Juan Luis Esteban and Jacobo Torán. A combinatorial characterization of treelike resolution space. *Information Processing Letters*, 87(6):295–300, 2003.

[21] Yuval Filmus, Massimo Lauria, Mladen Miksa, Jakob Nordström, and Marc Vinyals. From small space to small width in resolution. In *31st International Symposium on Theoretical Aspects of Computer Science, STACS*, pages 300–311, 2014.

[22] John R Gilbert, Thomas Lengauer, and Robert Endre Tarjan. The pebbling problem is complete in polynomial space. *SIAM Journal on Computing*, 9(3):513–524, 1980.

[23] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

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

[25] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[26] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer, 1972.

[27] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.

[28] Massimo Lauria, Pavel Pudlák, Vojtěch Rödl, and Neil Thapen. The complexity of proving that a graph is ramsey. *Combinatorica*, 37(2):253–268, Apr 2017.

[29] Daniel Lokshtanov, Dániel Marx, Saket Saurabh, et al. Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS*, 3(105), 2013.

[30] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

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

[32] Jakob Nordström. Narrow proofs may be spacious: separating space and width in resolution. In *STOC*, pages 507–516, 2006.

[33] Jakob Nordström. Pebble games, proof complexity and time-space trade-offs. *Logical Methods in Computer Science*, 9:15:1–15:63, September 2013.

[34] H.J. Prömel and V. Rödl. Non-ramsey graphs are c log n-universal. *Journal of Combinatorial Theory, Series A*, 88(2):379–384, 1999.

[35] Benjamin Rossman. On the constant-depth complexity of k-clique. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 721–730. ACM, 2008.

[36] Benjamin Rossman. The monotone complexity of k-clique on random graphs. *SIAM Journal on Computing*, 43(1):256–279, 2014. Preliminary version in *FOCS '10*.

[37] Nathan Segerlind. The complexity of propositional proofs. *Bulletin of symbolic Logic*, 13(4):482–537, 2007.

[38] V. Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 455–464. ACM, 2009.