Parameterized Complexity of DPLL Search Procedures[⋆]

Olaf Beyersdorff[1], Nicola Galesi[2⋆⋆], and Massimo Lauria[2]

[1] Institut für Theoretische Informatik, Leibniz Universität Hannover, Germany
[2] Dipartimento di Informatica, Sapienza Università di Roma, Italy

Abstract. We study the performance of DPLL algorithms on parameterized problems. In particular, we investigate how difficult it is to decide whether small solutions exist for satisfiability and combinatorial problems. For this purpose we develop a Prover-Delayer game which models the running time of DPLL procedures and we establish an information-theoretic method to obtain lower bounds to the running time of parameterized DPLL procedures. We illustrate this technique by showing lower bounds to the parameterized pigeonhole principle and to the ordering principle. As our main application we study the DPLL procedure for the problem of deciding whether a graph has a small clique. We show that proving the absence of a k-clique requires $n^{\Omega(k)}$ steps for a non-trivial distribution of graphs close to the critical threshold. For the restricted case of tree-like Parameterized Resolution, this result answers a question asked in [11] of understanding the Resolution complexity of this family of formulas.

1 Introduction

Resolution was introduced by Blake [12] and since the work of Robinson [25] and Davis, Putnam, Logemann, and Loveland [19, 20] has been highly employed in proof search and automated theorem proving. In the last years, the study of Resolution has gained great significance in at least two important fields of computer science. (1) *Proof complexity*, where Resolution is one of the most intensively investigated proof systems [1, 6, 8, 13, 16, 22, 30]. The study of lower bounds for proof lengths in this system has opened the way to lower bounds in much stronger proof systems [7, 28]. (2) *Algorithms for the satisfiability problem* of CNF formulas, where the DPLL algorithm [4, 19] is the core of the most important and modern algorithms employed for the satisfiability problem [4, 5].

Parameterized Resolution was recently introduced by Dantchev, Martin, and Szeider [18] in the context of *parameterized proof complexity*, an extension of the

[⋆] Part of this work was done while the first author was visiting Sapienza University of Rome. This Research is part of the project "Limits of Theorem Proving" supported by grant N. 20517 by the John Templeton Foundation.

[⋆⋆] Partly supported by Sapienza Research Project: Complessità e Rappresentabilità Compatta di Strutture Discrete.

proof complexity approach of Cook and Reckhow [17] to parameterized complexity. Analogously to the case of Fixed Parameter Tractable (FPT) algorithms for optimization problems, the study of Parameterized Resolution provides new approaches and insights to proof search and to proof complexity. Loosely speaking, to refute a parameterized contradiction (F, k) in Parameterized Resolution we have built-in access to new *axioms*, which encode some property on assignments. In the most common case the new axioms are the clauses forbidding assignments of hamming weight greater than k. We underline that only those axioms appearing in the proof account for the proof length. Hence Parameterized DPLL refutations can be viewed as traces of executions of a (standard) DPLL algorithm in which some branches are cut because they falsify one of the new axioms.

In spite of its recent introduction, research in this direction is already active. Gao [21] analyzes the effect of the standard DPLL algorithm on the problem of weighted satisfiability for random d-CNFs. Beyersdorff et al. [11], using an idea also developed in [15], proved that there are FPT efficient Parameterized Resolution proofs for *all* bounded-width unsatisfiable CNF formulae. The discovery of new implications for SAT-solving algorithms in Parameterized Resolution appears to be a promising research field at a very early stage of investigation.

As our first contribution, we look inside the structure of Parameterized DPLL giving a new information-theoretical characterization of proofs in terms of a two-player game, the *Asymmetric Prover-Delayer (APD) game*. The APD-game was also used in [10] to prove simplified optimal lower bounds for the pigeonhole principle (PHP) in tree-like (classical) Resolution. Compared to [10] we present here a completely different analysis of APD-games based on an information-theoretical argument which is new and interesting by itself.

Parameterized Resolution is also a refutational proof system for (parameterized) contradictions. Hence proving proof length lower bounds for (parameterized) contradictions is important in order to understand the strength of such a proof system. Dantchev et al. [18] proved significant lower bounds for Parameterized DPLL proofs of PHP and of the ordering principle (OP). Moreover, recently the work [11] extended the PHP lower bounds to the case of parameterized dag-like bounded-depth Frege.[3]

As our second contribution we provide a unified approach to reach significative lower bounds in Parameterized DPLL using the APD-game. As a simple application of our characterization, we obtain the optimal lower bounds given in [18] for PHP and OP.

It is a natural question what happens when we equip a proof system with a more efficient way of encoding the exclusion of assignments with hamming weight $\geq k$, than just adding all possible clauses with $k + 1$ negated variables. Dantchev et al. [18] proved that this is a significant point. They presented a different and more efficient encoding, and showed that under this encoding *PHP* admits efficient FPT Parameterized Resolution proofs.

[3] The APD-game appeared also in the technical report [9], together with a lower bound for dag-like Parameterized Resolution, but all results in [9] are subsumed and improved by [11] and the present paper.

In the previous work [11] we investigated this question further and noticed that for propositional encodings of prominent combinatorial problems like k-independent set or k-clique, the separation between the two encodings vanishes. Hence we proposed (see Question 5 in [11]) to study the performance of Parameterized Resolution on CNF encodings of such combinatorial problems and in particular to prove lower bounds. This will capture the real proof-theoretic strength of Parameterized Resolution, since it is independent of the encodings. The k-clique principle (see also [3, 11] for similar principles) simply says that a given graph contains a clique of size k. When applied on a graph not containing a k-clique it is a contradiction. On canonical graphs not containing a k-clique (as the $(k-1)$-partite complete graph) the k-clique principle admits efficient refutations in Parameterized Resolution.

As a third contribution, we prove significant lower bounds for the k-clique principle in the case of Parameterized DPLL. Our k-clique formula is based on random graphs distributed according to a simple variation of the Erdős-Rényi model $G(n, p)$. It is well known [23, Chapter 3] that when G is drawn according to $G(n, p)$ and $p \ll n^{-\frac{2}{k-1}}$, with high probability G has no k-clique.

The paper is organized as follows. Section 2 contains all preliminary notions and definitions concerning fixed-parameter tractability, parameterized proof systems, and Parameterized Resolution. In Section 3 we define our asymmetric Prover-Delayer game and establish its precise relation to the proof size in tree-like Parameterized Resolution. In Section 4, as an example of the application of the APD-game, we give a simplified lower bound for the pigeonhole principle in tree-like Parameterized Resolution. In Section 5 we introduce the formula $\mathit{Clique}(G, k)$ which is satisfiable if and only if there is a k-clique in the graph G and we show that on a certain distribution of random graphs the following holds with high probability: G has no k-clique and the size of the shortest refutation of $\mathit{Clique}(G, k)$ is $n^{\Omega(k)}$. From an algorithmic perspective, this result can be formulated as: Any algorithm for k-clique which cleverly selects a vertex and branches in whether it is in the clique or not, deletes all its non-neighbors and stops branching when there are no vertices left must use at least n^{ck} steps a constant $c > 0$ for most random graphs with a certain edge probability.

2 Preliminaries

Parameterized complexity is a branch of complexity theory where problems are analyzed in a finer way than in the classical approach: we say that a problem is *fixed-parameter tractable* (FPT) with parameter k if it can be solved in time $f(k)n^{O(1)}$ for some computable function f of arbitrary growth. In this setting classically intractable problems may have efficient solutions, assuming the parameter is small, even if the total size of the input is large. Parameterized complexity also has a completeness theory: many parameterized problems that appear not to be fixed-parameter tractable have been classified as being complete under fpt-reductions for complexity classes in the so-called weft hierarchy $\mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \mathsf{W}[3] \subseteq \ldots$.

Consider the problem Weighted CNF Sat of finding a satisfying assignment of Hamming weight at most k for a formula in conjunctive normal form. Many combinatorial problems can be naturally encoded in Weighted CNF Sat: finding a vertex cover of size at most k; finding a clique of size k; or finding a dominating set of size at most k. In the theory of parameterized complexity, the hardness of the Weighted CNF Sat problem is reflected by the fact that it is $\mathsf{W}[2]$-complete (see [11, 18]).

Dantchev, Martin, and Szeider [18] initiated the study of *parameterized proof complexity*. After considering the notions of propositional *parameterized tautologies* and *fpt-bounded* proof systems, they laid the foundations for the study of complexity of proofs in a parameterized setting. The problem Weighted CNF Sat leads to parameterized contradictions:

Definition 1 (Dantchev et al. [18]). *A* parameterized contradiction *is a pair* (F, k) *consisting of a propositional formula* F *and* $k \in \mathbb{N}$ *such that* F *has no satisfying assignment of weight* $\leq k$.

The notions of a parameterized proof system and of fpt-bounded proof systems were also developed in [18]:

Definition 2 (Dantchev et al. [18]). *A* parameterized proof system *for a parameterized language* $L \subseteq \Sigma^* \times \mathbb{N}$ *is a function* $P : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ *such that* $rng(P) = L$ *and* $P(x, k)$ *can be computed in time* $O(f(k)|x|^{O(1)})$ *with some computable function* f. *The system* P *is* fpt-bounded *if there exist computable functions* s *and* t *such that every* $(x, k) \in L$ *has a* P*-proof* (y, k') *with* $|y| \leq s(k)|x|^{O(1)}$ *and* $k' \leq t(k)$.

The main motivation behind the work of [18] was that of generalizing the classical approach of Cook and Reckhow [17] to the parameterized case and working towards a separation of complexity classes as FPT and $\mathsf{W}[2]$ by techniques developed in proof complexity.

2.1 Parameterized Resolution and Parameterized DPLL

A *literal* is a positive or negated propositional variable and a *clause* is a set of literals. The *width* of a clause is the number of its literals. A clause is interpreted as the disjunction of its literals and a set of clauses as the conjunction of the clauses. Hence clause sets correspond to formulas in CNF. The *Resolution system* is a refutation system for the set of all unsatisfiable CNF. Resolution gets its name from its only rule, the *Resolution rule* $\frac{\{x\} \cup C \quad \{\neg x\} \cup D}{C \cup D}$ for clauses C, D and a variable x. The aim in Resolution is to demonstrate unsatisfiability of a clause set by deriving the empty clause. If in a derivation every derived clause is used at most once as a prerequisite of the Resolution rule, then the derivation is called *tree-like*, otherwise it is *dag-like*. The *size* of a Resolution proof is the number of its clauses where multiple instances of the same clause are counted separately.

For the remaining part of this paper we will concentrate on *Parameterized Resolution* as introduced by Dantchev, Martin, and Szeider [18]. Parameterized

Resolution is a refutation system for the set of parameterized contradictions (cf. Definition 1). Given a set of clauses F in variables $x_1, \dots, x_n$, a *Parameterized Resolution refutation* of (F, k) is a Resolution refutation of the set of clauses $F \cup \{\neg x_{i_1} \vee \dots \vee \neg x_{i_{k+1}} \mid 1 \leq i_1 < \dots < i_{k+1} \leq n\}$. Thus, in Parameterized Resolution we have built-in access to all parameterized clauses of the form $\neg x_{i_1} \vee \dots \vee \neg x_{i_{k+1}}$. All these clauses are available in the system, but when measuring the size of a refutation we only count those which occur in the refutation.

If refutations are tree-like we speak of *tree-like Parameterized Resolution*. Running parameterized DPLL procedures on parameterized contradictions produces tree-like Parameterized Resolution refutations, thus tree-like Resolution proof lengths are connected with the running time of DPLL procedures. Exactly as in usual tree-like Resolution, a tree-like Parameterized refutation of (F, k) can equivalently be described as a *boolean decision tree* where inner nodes are labeled with variables from F and leaves are labeled with clauses from F or parameterized clauses $\neg x_{i_1} \vee \dots \vee \neg x_{i_{k+1}}$.

3 Asymmetric Prover-Delayer Games for DPLL

The original Prover-Delayer game for tree-like Resolution has been developed by Pudlák and Impagliazzo [24], and arises from the well-known fact that a tree-like Resolution refutation for a CNF F can be viewed as a decision tree which solves the search problem of finding a clause of F falsified by a given assignment. In the game, Prover queries a variable and Delayer either gives it a value or leaves the decision to Prover and receives *one* point. The number of Delayer's points at the end of the game bounds from below the height of the proof tree. Our new game, in contrast, assigns points to the Delayer asymmetrically ($\log c_0$ and $\log c_1$) according to two functions c_0 and c_1 (s.t. $c_0^{-1} + c_1^{-1} = 1$) which depend on the principle, the variable queried, and the current partial assignment. In fact, the original Prover-Delayer game of [24] is the case where $c_0 = c_1 = 2$.

Loosely speaking, we interpret the inverse of the score functions as a way to define a distribution on the choices made by the DPLL algorithm. Under this view the Delayer's score at each step is just the entropy of the bit encoding the corresponding choice. Since root-to-leaf paths are in bijection with leaves, this process induces a distribution on the leaves. Hence the entropy collected on the path is the entropy of the corresponding leaf choice. In this interpretation, the asymmetric Prover-Delayer game becomes a challenge between a Prover, who wants to end the game giving up little entropy, and Delayer, who wants to get a lot of it. This means that the average score of the Delayer is a measure (actually a lower bound) of the number of leaves. In our setup the DPLL algorithm decides the Prover queries, and the score function defines the distribution on paths. The Delayer role corresponds to a conditioning on such a distribution.

Let (F, k) be a parameterized contradiction where F is a set of clauses in n variables $x_1, \dots, x_n$. We define a Prover-Delayer game: Prover and Delayer build a (partial) assignment to $x_1, \dots, x_n$. The game is over as soon as the partial assignment falsifies either a clause from F or a parameterized clause

$\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$ where $1 \leq i_1 < \cdots < i_{k+1} \leq n$. The game proceeds in rounds. In each round, Prover suggests a variable x_i, and Delayer either chooses a value 0 or 1 for x_i or leaves the choice to the Prover. In this last case, if the Prover sets the value, then the Delayer gets some points. The number of points Delayer earns depends on the variable x_i, the assignment α constructed so far in the game, and two functions $c_0(x_i, \alpha)$ and $c_1(x_i, \alpha)$. More precisely, the number of points that Delayer will get is

$$\begin{array}{cl} 0 & \text{if Delayer chooses the value,} \\ \log c_0(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 0, and} \\ \log c_1(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 1.} \end{array}$$

Moreover, the functions $c_0(x, \alpha)$ and $c_1(x, \alpha)$ are non negative and are chosen in such a way that for each variable x and assignment α

$$\frac{1}{c_0(x, \alpha)} + \frac{1}{c_1(x, \alpha)} = 1 \tag{1}$$

holds. We remark that (1) is not strictly necessary for all α and x, but it must hold at least for those assignments α and choices x of the Delayer that can actually occur in any game with the Delayer strategy. We call this game the (c_0, c_1)-game on (F, k). The connection of this game to size of proofs in tree-like Parameterized Resolution is given by the next theorem:

Theorem 3. *Let (F, k) be a parameterized contradiction and let c_0 and c_1 be two functions satisfying (1) for all partial assignments α to the variables of F. If (F, k) has a tree-like Parameterized Resolution refutation of size at most S, then for each (c_0, c_1)-game played on (F, k) there is a Prover strategy (possibly dependent on the Delayer) that gives the Delayer at most $\log S$ points.*

Proof. Let (F, k) be a parameterized contradiction using variables $x_1, \ldots, x_n$. Choose any tree-like Parameterized Resolution refutation of (F, k) of size S and interpret it as a boolean decision tree T for F. The decision tree T completely specifies the query strategy for Prover: at the first step he will query the variable labeling the root of T. Whatever decision is made regarding the value of the queried variable, Prover moves to the root of the corresponding subtree and queries the variable which labels it. This process induces a root-to-leaf walk on T, and such walks are in bijection with the set of leafs.

To completely specify Prover's strategy we need to explain how Prover chooses the value of the queried variable in case Delayer asks him to. A game position is completely described by the partial assignment α computed so far, and by the variable $x \notin dom(\alpha)$ queried at that moment. If the Prover is asked to answer the query for x, the answer will be: $\begin{cases} 0 & \text{with probability } \frac{1}{c_0(x,\alpha)} \\ 1 & \text{with probability } \frac{1}{c_1(x,\alpha)} \end{cases}$. Thus we are dealing with a randomized Prover strategy. In a game played between our randomized Prover and a specific Delayer D, we denote by $p_{D,\ell}$ the probability of such a game to end at a leaf ℓ. We call π_D this distribution on the leaves. To prove the theorem the following observation is crucial:

If the game ends at leaf ℓ, then Delayer D scores exactly $\log \frac{1}{p_{D,\ell}}$ *points.*

Before proving this claim, we show that it implies the theorem. The expected score of a Delayer D is

$$H(\pi_D) = \sum_{\ell} p_{D,\ell} \log \frac{1}{p_{D,\ell}}$$

which is the information-theoretic entropy of π_D. Since the support of π_D has size at most S, we obtain $H(\pi_D) \leq \log S$, because the entropy is maximized by the uniform distribution. By fixing the random choices of the Prover, we can force Delayer D to score at most $\log S$ points.

To prove the claim consider a leaf ℓ and the unique path that reaches it. W. l. o. g. we assume that this path corresponds to the ordered sequence of assignments $x_1 = \epsilon_1, \dots, x_m = \epsilon_m$. The probability of reaching the leaf is

$$p_{D,\ell} = p_1 p_2 \cdots p_m$$

where p_i is the probability of setting $x_i = \epsilon_i$ conditioned on the previous choices. If Prover chooses the value of the variable x_i, the score Delayer D gets at step i is

$$\log c_{\epsilon_i}(x_i, \{x_1 = \epsilon_1, x_2 = \epsilon_2, \dots, x_{i-1} = \epsilon_{i-1}\})$$

which is exactly $\log \frac{1}{p_i}$. If Delayer makes the choice at step i, then $p_i = 1$ and the score is 0, which is also $\log \frac{1}{p_i}$. Thus the score of the game play is

$$\sum_{i=1}^{m} \log \frac{1}{p_i} = \log \frac{1}{\prod_{i=1}^{m} p_i} = \log \frac{1}{p_{D,\ell}} \ ,$$

and this concludes the proof of the claim and the theorem. □

4 An Application of the Lower Bound Method

We will illustrate the use of asymmetric Prover-Delayer games with an application on the *pigeonhole principle* PHP_n^{n+1} which uses variables $x_{i,j}$ with $i \in [n+1]$ and $j \in [n]$, indicating that pigeon i goes into hole j. PHP_n^{n+1} consists of the clauses $\bigvee_{j \in [n]} x_{i,j}$ for all pigeons $i \in [n+1]$ and $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ for all choices of distinct pigeons $i_1, i_2 \in [n+1]$ and holes $j \in [n]$. We prove that PHP_n^{n+1} is hard for tree-like Parameterized Resolution.

Theorem 4. *Any tree-like Parameterized Resolution refutation of* (PHP_n^{n+1}, k) *has size* $n^{\Omega(k)}$.

Proof. Let α be a partial assignment to the variables $\{x_{i,j} \mid i \in [n+1], j \in [n]\}$. Let $z_i(\alpha) = |\{j \in [n] \mid \alpha(x_{i,j}) = 0\}|$, i. e., $z_i(\alpha)$ is the number of holes already excluded by α for pigeon i. We define

$$c_0(x_{i,j}, \alpha) = \frac{n - z_i(\alpha)}{n - z_i(\alpha) - 1} \quad \text{and} \quad c_1(x_{i,j}, \alpha) = n - z_i(\alpha)$$

which clearly satisfies (1). We now describe Delayer's strategy in a (c_0, c_1)-game played on (PHP_n^{n+1}, k). If Prover asks for a value of $x_{i,j}$, then Delayer decides as follows:

$$\begin{array}{ll}
\text{set } \alpha(x_{i,j}) = 0 & \text{if there exists } i' \in [n+1] \setminus \{i\} \text{ such that } \alpha(x_{i',j}) = 1 \text{ or} \\
 & \text{if there exists } j' \in [n] \setminus \{j\} \text{ such that } \alpha(x_{i,j'}) = 1 \\
\text{set } \alpha(x_{i,j}) = 1 & \text{if there is no } j' \in [n] \text{ with } \alpha(x_{i,j'}) = 1 \text{ and } z_i(\alpha) \geq n-k \\
\text{let Prover decide} & \text{otherwise.}
\end{array}$$

Intuitively, Delayer leaves the choice to Prover as long as pigeon i does not already sit in a hole, but there are at least k holes free for pigeon i, and there is no other pigeon sitting already in hole j. If Delayer uses this strategy, then clauses from PHP_n^{n+1} will not be violated in the game, i. e., a contradiction will always be reached on some parameterized clause. To verify this claim, let α be a partial assignment constructed during the game with $w(\alpha) \leq k$ (we denote the the weight of α by $w(\alpha)$). Then, for every pigeon which has not been assigned to a hole yet, there are at least k holes where it could go (and of these only $w(\alpha)$ holes are already occupied by other pigeons). Thus α can be extended to a one-one mapping of exactly k pigeons to holes.

Therefore, at the end of the game exactly $k+1$ variables have been set to 1. Let us denote by p the number of variables set to 1 by Prover and let d be the number of 1's assigned by Delayer. As argued before $p + d = k + 1$. Let us check how many points Delayer earns in this game. If Delayer assigns 1 to a variable $x_{i,j}$, then pigeon i was not assigned to a hole yet and, moreover, there must be $n - k$ holes which are already excluded for pigeon i by α, i. e., for some $J \subseteq [n]$ with $|J| = n - k$ we have $\alpha(x_{i,j'}) = 0$ for all $j' \in J$. Most of these 0's have been assigned by Prover, as Delayer has only assigned a 0 to $x_{i,j'}$ when some other pigeon was already sitting in hole j', and there can be at most k such holes. Thus, before Delayer sets $\alpha(x_{i,j}) = 1$, she has already earned points for at least $n - 2k$ variables $x_{i,j'}$, $j' \in J$, yielding at least

$$\sum_{z=0}^{n-2k-1} \log\frac{n-z}{n-z-1} = \log\prod_{z=0}^{n-2k-1}\frac{n-z}{n-z-1} = \log\frac{n}{2k} = \log n - \log 2k$$

points for the Delayer. Note that because Delayer never allows a pigeon to go into more than one hole, she will earn at least the number of points calculated above for *each* of the d variables which she sets to 1.

If, conversely, Prover sets variable $x_{i,j}$ to 1, then Delayer gets $\log(n - z_i(\alpha))$ points for this, but she also received points for most of the $z_i(\alpha)$ variables set to 0 before that. Thus, in this case Delayer earns on pigeon i at least

$$\log\left(n - z_i(\alpha)\right) + \sum_{z=0}^{z_i(\alpha)-k-1} \log\frac{n-z}{n-z-1} =$$

$$\log\left(n - z_i(\alpha)\right) + \log\frac{n}{n - z_i(\alpha) + k} = \log n - \log\frac{n - z_i(\alpha) + k}{n - z_i(\alpha)} \geq \log n - \log k$$

points. In total, Delayer gets at least

$$d(\log n - \log 2k) + p(\log n - \log k) \geq k(\log n - \log 2k)$$

points in the game. By Theorem 3, we obtain $(\frac{n}{2k})^k$ as a lower bound to the size of each tree-like Parameterized Resolution refutation of (PHP_n^{n+1}, k). □

As a second example we discuss the DPLL performance on the parameterized *ordering principle* OP, also called *least element principle*. The principle claims that any finite partially ordered set has a minimal element. There is a direct propositional translation of OP to a family OP_n of unsatisfiable CNFs. Each CNF OP_n expresses that there exists a partially ordered set of size n such that any element has a predecessor. The ordering principle has the following clauses:

$$\begin{array}{lll}
\neg x_{i,j} \vee \neg x_{j,i} & \text{for every } i,j & \text{(Antisymmetry)} \\
\neg x_{i,j} \vee \neg x_{j,k} \vee \; x_{i,k} & \text{for every } i,j,k & \text{(Transitivity)} \\
\displaystyle\bigvee_{j \in [n] \setminus \{i\}} x_{j,i} & \text{for every } i & \text{(Predecessor)}
\end{array}$$

With respect to parameterization the ordering principles are interesting. Both OP and the *linear ordering principle* (LOP), which additionally assumes the order to be total, do not admit short tree-like Resolution refutations [14] and have general Resolution refutations of polynomial size [29]. In the parameterized setting things are different: LOP has short tree-like refutations (see [11]) while OP does not and provides a separation between tree-like and dag-like Parameterized Resolution. The following theorem has been first proved in [18]. Their proof is based on a model-theoretic criterion, while ours is based on the Prover-Delayer game. The proof will appear in the full version of this paper (see also [9]).

Theorem 5. *Any tree-like Parameterized Resolution refutation of* (OP_n, k) *has size* $n^{\Omega(k)}$.

5 DPLL and the Decision Tree Complexity of *k*-Clique

Instead of adding parameterized clauses of the form $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$, there are also more succinct ways to enforce only satisfying assignments of weight $\leq k$. One such method was considered in [18] where for a formula F in n variables $x_1, \ldots, x_n$ and a parameter k, a new formula $M = M(F,k)$ is computed such that $F \wedge M$ is satisfiable if and only if F has a satisfying assignment of weight at most k. The formula M uses new variables $s_{i,j}$, where $i \in [k]$ and $j \in [n]$, and consists of the clauses

$$\neg x_j \vee \bigvee_{i=1}^{k} s_{i,j} \quad \text{and} \quad \neg s_{i,j} \vee \; x_j \qquad \text{for } i \in [k] \text{ and } j \in [n] \tag{2}$$

$$\neg s_{i,j} \vee \; \neg s_{i,j'} \qquad \text{for } i \in [k] \text{ and } j \neq j' \in [n] \tag{3}$$

$$\neg s_{i,j} \vee \; \neg s_{i',j} \qquad \text{for } i \neq i' \in [k] \text{ and } j \in [n]. \tag{4}$$

The clauses (2) express the fact that an index i is associated to a variable x_j if and only if such variable is set to true. The fact that the association is an injective function is expressed by the clauses (3) and (4).

In [11] we argue that the clique formulas are "invariant" with respect to this transformation, thus its classical proof complexity is equivalent to its parameterized proof complexity (in both the formulation with explicit parameterized axioms and the succinct encoding). Therefore in [11] we posed the question of determining the complexity of the clique formulas in Resolution. Theorem 7 below provides an answer to this question for the tree-like case.

Our study focuses on the average-case complexity of proving the absence of a k-clique in random graphs distributed according to a variation of the Erdős-Rényi model $G(n, p)$. It is known that k-cliques appear at the threshold probability $p^* = n^{-\frac{2}{k-1}}$. If $p < p^*$, then with high probability there is no k-clique; while for $p > p^*$ with high probability there are many. For $p = p^*$ there is a k-clique with constant probability.

The complexity of k-clique has been already studied in restricted computational models by Rossman [26, 27]. He shows that in these models any circuit which succeeds with good probability on graph distributions close to the critical threshold requires size $\Omega(n^{\frac{k}{4}})$, and even matching upper bounds exist in these models [2, 27]. Since we want to study negative instances of the clique problem, we focus on probability distributions with $p < p^*$. To ease the proof presentation we will prove a lower bound for slightly sparser distributions. We now give the CNF formulation of a statement claiming that a k-clique exists in a graph.

Definition 6. *Given a graph $G = (V, E)$ and a parameter k, $Clique(G, k)$ is a formula in conjunctive normal form containing the following clauses*

$$\bigvee_{v \in V} x_{i,v} \qquad \text{for every } i \in [k] \tag{5}$$

$$\neg x_{i,u} \vee \neg x_{j,v} \qquad \text{for every } i, j \in [k],\ i \neq j \text{ and every } \{u, v\} \notin E \tag{6}$$

$$\neg x_{i,u} \vee \neg x_{i,v} \qquad \text{for every } u \neq v \in V. \tag{7}$$

Clearly, the formula $Clique(G, k)$ is satisfiable if and only if the graph G has a clique of size k.

We now describe a family of hard graph instances for k-clique: such graphs have a simplified structure to make the proof more understandable. We also restrict the formula, which makes it easier. This only strengthens eventual lower bounds. We consider a random graph G on kn vertices. The set of vertices V is divided into k blocks of n vertices each, named $V_1, V_2, \ldots, V_k$. Edges may be present only between vertices of different blocks. The random variable in the graph is the set of edges. For any constant ϵ and any pair of vertices (u, v) with $u \in V_i$, $v \in V_{i'}$ and $i < i'$, the edge $\{u, v\}$ is present with probability

$$p = n^{-(1+\epsilon)\frac{2}{k-1}}.$$

We call this distribution of graphs $\mathcal{G}_\epsilon$. Notice that all graphs in $\mathcal{G}_\epsilon$ are properly colorable with k colors. Later we will focus on a specific range for ϵ.

In a k-colorable graph, each clique contains at most one vertex per color class. Because of this observation we can simplify the k-clique formula in the following way, which we call $h(G)$

$$\bigvee_{v \in V_i} x_v \qquad \text{for every } i \in [k] \tag{8}$$

$$\neg x_u \vee \neg x_v \qquad \text{for every } \{u, v\} \notin E(G). \tag{9}$$

We omit the parameter k in the notation of h. This is only a notational detail and we want to keep notation as simple as possible. We now see that a lower bound to the size of a (tree-like) Resolution refutation of $h(G)$ transfers to the same lower bound for $\mathit{Clique}(G, k)$.

Fact 1 *Let G be a k-colorable graph. Then each (tree-like) Resolution refutation of $\mathit{Clique}(G, k)$ can be transformed into a (tree-like) Resolution refutation of $h(G)$ of the same size (with the partition in $h(G)$ induced by the coloring).*

A comment regarding the encoding is required. In [3] formulas similar to $\mathit{Clique}(G, k)$ and $h(G)$ have been studied for the dual problem of independent sets. They study the case of $k = \Omega(n)$, so the former encoding has a lower bound because it contains clauses of a non-trivial pigeonhole principle. In the parameterized framework this is not necessarily true, since k is small and PHP^k_{k-1} is feasible here.

We will now show that for a random graph $G \in \mathcal{G}_\epsilon$ any decision tree which proves unsatisfiability of k-clique has size $n^{\Omega(k(1-\epsilon))}$ with high probability. To show that k-clique requires refutations of size $n^{\Omega(k(1-\epsilon))}$ it suffices to exhibit two score functions c_0 and c_1 and a Delayer strategy such that the Delayer is guaranteed to score $\Omega(k(1-\epsilon)\log n)$ points in any game played against any Prover.

Theorem 7. *For any $0 < \epsilon < 1$. For a random $G \in \mathcal{G}_\epsilon$ the k-clique CNF requires tree-like Parameterized Resolution refutations of size $n^{\Omega(k(1-\epsilon))}$ with high probability.*

Proof. Let G be a random graph distributed according to $\mathcal{G}_\epsilon$. For a set S of vertices, let $\Gamma^c(S)$ be the set of common neighbors of S. We first show that with high probability the following properties hold:

1. G has no clique of size k;
2. For any set S of less than $\frac{k}{4}$ vertices in distinct blocks, $|\Gamma^c(S) \cap V_b| \geq n^{\Omega(1-\epsilon)}$ for any block V_b disjoint from S.

For item 1: the expected number of k-cliques in G is $n^k p^{\binom{k}{2}} = n^{-k\epsilon}$. By Markov inequality, the probability of the existence of a single k-clique is at most the expected value.

For item 2: it is sufficient to show the statement for sets of size exactly $\frac{k}{4} - 1$. Fix any such set S, and fix any block V_b which does not contain vertices in this set. We denote by X_i the random variable which is 1 when $i \in \Gamma^c(S)$, and 0 otherwise. Thus the size of $V_b \cap \Gamma^c(S)$ is the sum of n independent variables. Notice that X_i is 1 with probability $p^{\frac{k}{4}-1} \geq n^{-\frac{1+\epsilon}{2}}$. Thus the expected value is at least $n^{\frac{1-\epsilon}{2}}$. We define $T = \frac{n^{\frac{1-\epsilon}{2}}}{2}$. Since $T = n^{\Omega(1-\epsilon)}$ and T is a constant fraction of the expected value, by the Chernoff bound we obtain that $V_b \cap \Gamma(S)$ has size less than T with probability $e^{-n^{\Omega(1-\epsilon)}}$. By the union bound on the choices of block V_b and of set S of size $\frac{k}{4} - 1$ we get item 2.

We now define functions c_0 and c_1 which are legal cost functions for an asymmetric Prover-Delayer game played on the k-clique formula of the graph G. We also show a Delayer strategy which is guaranteed to score $\Omega(k \log T)$ points. This, together with Theorem 3, implies the main statement.

For any partial assignment α we consider the set of vertices "chosen by α", which is $\{u \mid \alpha(x_u) = 1\}$; any vertex which is the common neighbor of the chosen set is called "good for α". Notice that a good vertex for α can be set to 1 without causing an immediate contradiction. Notice also that α may set to 0 some good vertices. In particular we denote by $R_b(\alpha)$ the vertices of the block V_b which are good for α, but are nevertheless set to 0 in α.

When asked for a variable x_v, for some $v \in V_b$, the Delayer behaves according to the following strategy:

- If α contains at least $\frac{k}{4}$ variables set to 1, the Delayer surrenders;
- if there is u such that $\alpha(x_u) = 1$ and $\{u, v\} \notin E(G)$, the Delayer answers 0;
- if $R_b(\alpha)$ has size at least $T - 1$, then the Delayer answers 1;
- otherwise the Delayer leaves the answer to the Prover.

During the game the invariant $|R_b(\alpha)| < T$ holds for every $b \in [k]$: the only way such a set can increase in size is when Prover sets a good vertex in V_b to 0. Thus the size of $R_b(\alpha)$ can only increase one by one. When it reaches $T - 1$ and the Delayer is asked for a variable in that block, she will reply 1, so the size of $R_b(\alpha)$ won't increase any more.

Another important property of the Delayer strategy is that her decision to answer 1 never falsifies a clause, since all blocks contain at least T good vertices at any moment during the game. This follows from item 2 and from the fact that the Delayer surrenders after $\frac{k}{4}$ vertices are set in α. This proves that no clause in (8) can be falsified during the game.

Neither clauses in (9) can be falsified during the game: the Delayer imposes answer 0 whenever a vertex is not good for α, which means that, if chosen, it would not form a clique with the ones chosen before. It is also not possible that the game ends by violating a parameterized clause as these are just weakenings of the clauses (9). Therefore, the game only ends when the Delayer gives up.

For an assignment α and a vertex $v \in V_b$, let

$$c_0 = \frac{T - |R_b(\alpha)|}{T - |R_b(\alpha)| - 1} \qquad \text{and} \qquad c_1 = T - |R_b(\alpha)|.$$

Because of the previous observations the values of c_0 and c_1 are always non-negative. Furthermore notice that when $|R_b(\alpha)| = T - 1$ Delayer never leaves the choice to Prover, thus c_0 is always well defined when the Delayer scores.

Consider a game play and the set of $\frac{k}{4}$ vertices chosen by the final partial assignment α. We show that for any chosen vertex, the Delayer scores $\log T$ points for queries in the corresponding block.

Fix the block b of a chosen vertex u. Consider the assignment α which corresponds to the game step when x_u is set to 1. Consider $R = R_b(\alpha)$. We identify partial assignments $\alpha_0 \subset \alpha_1 \subset \ldots \subset \alpha_{|R|-1} \subset \alpha$ corresponding to the moments in the game when Prover sets to 0 one of the variables indexed by R. For such rounds the Delayer gets at least

$$\sum_{i=0}^{|R|-1} \log \frac{T - |R_b(\alpha_i|)}{T - |R_b(\alpha_i)| - 1} \geq \sum_{i=0}^{|R|-1} \log \frac{T-i}{T-i-1} = \log(T) - \log(T - |R|)$$

points. Here the first inequality follows from the fact that any vertex which is good at some stage of the game is also good in all previous stages. Thus $|R_b(\alpha_i)| \geq i$.

Now we must consider two cases: either $x_u = 1$ is set by Prover, or it is set by Delayer. In the former case Delayer gets $\log(T - |R|)$ points for Prover setting $x_u = 1$. Together with the points for the previous zeros this yields $\log T$ points. In the latter case Delayer gets 0 points as she set $x_u = 1$ by herself, but now $|R| = T - 1$ and she got already $\log T$ points for all the zeros assigned by Prover. In both cases the total score of the Delayer is $\log T = \frac{1-\epsilon}{2} \log n$.

Since this score is obtained in at least $\frac{k}{4}$ blocks, we are done. □

Acknowledgments. We thank the anonymous referees for their insightful suggestions which helped to improve the paper.

References

1. M. Alekhnovich and A. A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM Journal on Computing*, 38(4):1347–1363, 2008. 1
2. K. Amano. Subgraph isomorphism on AC^0 circuits. *Computational Complexity*, 19(2):183–210, 2010. 10
3. P. Beame, R. Impagliazzo, and A. Sabharwal. The resolution complexity of independent sets and vertex covers in random graphs. *Comput. Complex.*, 16(3):245–297, 2007. 3, 11
4. P. Beame, R. M. Karp, T. Pitassi, and M. E. Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM J. Comput.*, 31(4):1048–1075, 2002. 1
5. P. Beame, H. A. Kautz, and A. Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res.*, 22:319–351, 2004. 1
6. P. Beame and T. Pitassi. Simplified and improved resolution lower bounds. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science*, pages 274–282, 1996. 1
7. P. W. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Mathematical Society*, 73(3):1–26, 1996. 1

8. E. Ben-Sasson and A. Wigderson. Short proofs are narrow - resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001. 1
9. O. Beyersdorff, N. Galesi, and M. Lauria. Hardness of parameterized resolution. Technical Report TR10-059, Electronic Colloquium on Computational Complexity, 2010. 2, 9
10. O. Beyersdorff, N. Galesi, and M. Lauria. A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games. *Information Processing Letters*, 110(23):1074–1077, 2010. 2
11. O. Beyersdorff, N. Galesi, M. Lauria, and A. Razborov. Parameterized bounded-depth Frege is not optimal. Technical Report TR10-198, Electronic Colloquium on Computational Complexity, 2010. 1, 2, 3, 4, 9, 10
12. A. Blake. *Canonical expressions in boolean algebra*. PhD thesis, University of Chicago, 1937. 1
13. M. L. Bonet, J. L. Esteban, N. Galesi, and J. Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000. 1
14. M. L. Bonet and N. Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001. 9
15. Y. Chen and J. Flum. The parameterized complexity of maximality and minimality problems. *Annals of Pure and Applied Logic*, 151(1):22–61, 2008. 2
16. V. Chvátal and E. Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988. 1
17. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979. 2, 4
18. S. S. Dantchev, B. Martin, and S. Szeider. Parameterized proof complexity. In *Proc. 48th IEEE Symposium on the Foundations of Computer Science*, pages 150–160, 2007. 1, 2, 4, 9
19. M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962. 1
20. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:210–215, 1960. 1
21. Y. Gao. Data reductions, fixed parameter tractability, and random weighted d-CNF satisfiability. *Artificial Intelligence*, 173(14):1343–1366, 2009. 2
22. A. Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985. 1
23. S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs*. Wiley, 2000. 3
24. P. Pudlák and R. Impagliazzo. A lower bound for DLL algorithms for SAT. In *Proc. 11th Symposium on Discrete Algorithms*, pages 128–136, 2000. 5
25. J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965. 1
26. B. Rossman. On the constant-depth complexity of k-clique. In *Proc. 40th ACM Symposium on Theory of Computing*, pages 721–730, 2008. 10
27. B. Rossman. The monotone complexity of k-clique on random graphs. In *Proc. 51th IEEE Symposium on the Foundations of Computer Science*, pages 193–201. IEEE Computer Society, 2010. 10
28. N. Segerlind, S. R. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004. 1
29. G. Stalmark. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33:277–280, 1996. 9
30. A. Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. 1