Long Proofs of (Seemingly) Simple Formulas

Mladen Mikša and Jakob Nordström

School of Computer Science and Communication
KTH Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Abstract. In 2010, Spence and Van Gelder presented a family of CNF formulas based on combinatorial block designs. They showed empirically that this construction yielded small instances that were orders of magnitude harder for state-of-the-art SAT solvers than other benchmarks of comparable size, but left open the problem of proving theoretical lower bounds. We establish that these formulas are exponentially hard for resolution and even for polynomial calculus, which extends resolution with algebraic reasoning. We also present updated experimental data showing that these formulas are indeed still hard for current CDCL solvers, provided that these solvers do not also reason in terms of cardinality constraints (in which case the formulas can become very easy). Somewhat intriguingly, however, the very hardest instances in practice seem to arise from so-called fixed bandwidth matrices, which are provably easy for resolution and are also simple in practice if the solver is given a hint about the right branching order to use. This would seem to suggest that CDCL with current heuristics does not always search efficiently for short resolution proofs, despite the theoretical results of [Pipatsrisawat and Darwiche 2011] and [Atserias, Fichte, and Thurley 2011].

1 Introduction

Modern applied SAT solving is a true success story, with current state-of the art solvers based on *conflict-driven clause learning (CDCL)* [4, 21, 23] having delivered performance improvements of orders of magnitude larger than seemed possible just 15–20 years ago. From a theoretical perspective, however, the dominance of the CDCL paradigm is somewhat surprising in that it is ultimately based on the fairly weak *resolution* proof system [9]. Since it is possible in principle to extract a resolution refutation of an unsatisfiable formula from the execution trace of a CDCL solver running on it, lower bounds on resolution refutation length/size yield lower bounds on the running time of any CDCL solver trying to decide this formula.[1] By now, there is a fairly extensive literature on SAT instances for which exponential lower bounds are known, imposing firm restrictions on what kind of formulas the basic CDCL approach can hope to solve.

[1] Provided that the solver does not reason in terms of cardinality constraints or systems of linear equations and does not introduce new variables to apply extended resolution, in which case the theoretical lower bound guarantees no longer apply.

This suggests that an interesting question might be to turn the tables and ask for *maximally hard* instances. What are the smallest CNF formulas that are beyond reach of the currently best solvers? Pigeonhole principle (PHP) formulas were the first to be proven hard for resolution in the breakthrough result by Haken [15], but in terms of formula size N their hardness scales only as $\exp\bigl(\Omega\bigl(\sqrt[3]{N}\bigr)\bigr)$. Two formula families with refutation length $\exp(\Omega(N))$ are Tseitin formulas[2] over so-called expander graphs and random k-CNF formulas, as shown by Urquhart [27] and Chvátal and Szemerédi [11], respectively. The strongest lower bounds to date in terms of the explicit constant in the exponent were established recently by Beck and Impagliazzo [5].

Spence [26] instead focused on empirical hardness and exhibited a family of 3-CNF formulas that seem practically infeasible even for very small instances (around 100 variables). These formulas can be briefly described as follows. Fix a set of $4n+1$ variables. Randomly partition the variables into groups of 4 plus one group of 5. For each 4-group, write down the natural 3-CNF formula encoding the *positive cardinality constraint* that at least 2 variables must be true, and for the 5-group encode that a strict majority of 3 variables must be true. Do a second random variable partition into 4-groups plus one 5-group, but now encode *negative cardinality constraints* that the number of false variables is at least 2 and 3, respectively. By a counting argument, the CNF formula consisting of the conjunction of all these clauses must be unsatisfiable. Although [26] does not present any theoretical analysis, these formulas have a somewhat pigeonhole principle-like flavour and one can intuitively argue that they would seem likely to be hard provided that every moderately large set of positive cardinality constraints involves variables from many different negative constraints.

This construction was further developed by Van Gelder and Spence in [28], where the variable partitioning is done in terms of an $n \times n$ matrix with 4 non-zero entries in each row and column except that one extra non-zero entry is added to some empty cell. The variables in the formula correspond to the non-zero entries, each row is a positive cardinality constraint on its non-zero entries just as before, and each column provides a negative cardinality constraint. Equivalently, this formula can be constructed on a bipartite graph which is 4-regular on both sides except that one extra edge is added. In addition, there is a "no quadrangles" requirement in [28] that says that the graph contains no cycles of length 4. Just as above, it seems reasonable to believe that such formulas should be hard for resolution if the graph is a good expander. One such instance on 105 variables was issued by [28] as a "challenge formula" to be solved by any SAT solver in less than 24 hours, and in the concluding remarks the authors ask whether lower bounds on resolution length can be proven for formulas generated in this way.

1.1 Our Theoretical Results

We show that the formulas in [26, 28] are exponentially hard for resolution if the collection of constraints have a certain expansion property, and that random

[2] Encoding that the sum of the vertex indegrees in an undirected graph is even.

instances of these formulas are expanding with overwhelming probability. Let U denote the set of positive cardinality constraints and V the set of negative constraints. Then we can represent the formulas in [28] (and [26]) as bipartite (multi-)graphs $G = (U \dot\cup V, E)$, where edges are identified with variables and $x = (u, v) \in E$ if x occurs in both $u \in U$ and $v \in V$. Informally, we obtain the following lower bound for resolution (see Theorem 6 for the formal statement).

Theorem 1 (Informal). *If a 4-regular bipartite (multi-)graph G with one extra edge added is a sufficiently good expander, then the formula in [28] generated from G (or in [26] if G is a multigraph) requires resolution refutations of length $\exp\bigl(\Omega(n)\bigr)$. In particular, random instances of these formulas require resolution length $\exp\bigl(\Omega(n)\bigr)$ asymptotically almost surely.*

As a side note, we remark that the "no quadrangles" condition discussed above is not necessary (nor sufficient) for this theorem to hold—the more general notion of expansion is enough to guarantee that the formulas will be hard.

In one sentence, the proof works by reducing the formula to the pigeonhole principle on a 3-regular bipartite graph, which is then shown to be hard by a slight tweak of the techniques developed by Ben-Sasson and Wigderson [6]. A more detailed (if still incomplete) proof sketch is as follows. Start by fixing any complete matching in G (which can be shown to exist) and set all the matched edges plus the added extra edge to true. Also, set all remaining edges incident to the unique degree-5 vertex v^* on the right to false (this satisfies the negative constraint for v^*, which means that the corresponding clauses vanish). After this restriction, we are left with n constraints on the left which require that at least 1 out of the remaining 3 variables should be true, whereas on the right we have $n - 1$ constraints which all require that at most 1 remaining variable is true. But this is just a restricted PHP formula where each pigeon can go into one of three holes. Since we had a bipartite expander graph before restricting edges, and since not too many edges were removed, the restricted graph is still an expander. Now we can argue along the lines of [6] to obtain a linear lower bound on the refutation width from which an exponential length lower bound follows (and since restrictions can only make formulas easier, this lower bound must also hold for the original formula).

In fact, using tools from [2] one can show that the formulas are hard not only for resolution but also for *polynomial calculus resolution* [1, 12], which adds the power of Gröbner basis computations to resolution.

Theorem 2 (Informal). *For 4-regular bipartite (multi-)graphs with one extra edge that are sufficiently good expanders the formulas in [26, 28] require refutations of size $\exp\bigl(\Omega(n)\bigr)$ in polynomial calculus resolution (PCR). In particular, randomly sampled instances of these formulas require PCR refutation size $\exp\bigl(\Omega(n)\bigr)$ asymptotically almost surely.*

The technical details of this argument get substantially more involved, however. Thus, although Theorem 1 is strictly subsumed by Theorem 2, we focus mostly on the former theorem since it has a much cleaner and simpler proof that can be presented in full within the page limits of this extended abstract.

1.2 Our Empirical Results

We report results from running some current state-of-the-art SAT solvers on random instances of the formulas constructed by Spence [26] and Van Gelder and Spence [28], as well as on so-called *fixed bandwidth* versions of these formulas. The latter are formulas for which the non-zero entries on each row in the matrix appear on the diagonal and at some fixed (and small) horizontal offsets from the diagonal. Such matrices yield highly structured formulas, and as pointed out in [28] it is not hard to show that these formulas have polynomial-size refutations.

Our findings are that random instances of the formulas in [26, 28] are very hard, and become infeasible for slightly above 100 variables. As could be expected, the formulas in [28] are somewhat harder than the original formulas in [26], since the former are guaranteed not to have any multi-edges in the bipartite graph representing the constraints and thus "spread out" variables better among different constraints. However, to our surprise the formulas that are hardest in practice are actually the ones generated from fixed bandwidth matrices. A priori, one possible explanation could be that although the formulas are theoretically easy, the constants hidden in the asymptotic notation are so bad that the instances are hard for all practical purposes. This appears not to be the case, however—when the SAT solver is explicitly given a good variable branching order the fixed bandwidth formulas are solved much more quickly. Thus, this raises the question whether this could perhaps be an example of formulas for which CDCL with current state-of-the-art heuristics fails to search effectively for resolution proofs. This stands in intriguing contrast to the theoretical results in [3, 25], which are usually interpreted as saying that CDCL essentially harnesses the full power of resolution.

We have also done limited experiments with feeding the formulas in [26, 28] to Sat4j [7], the latest version of which can detect (small) cardinality constraints [8]. It is not hard to see that if the SAT solver is given the power to count, then it could potentially figure out quickly that it cannot possibly be the case that a strict majority of the variables is both true and false simultaneously. Indeed, this is also what happens, and in particular Sat4j solves the challenge formula from [28] in less than a second.

1.3 Organization of This Paper

After reviewing some preliminaries in Section 2, we state and prove formal versions of our proof complexity lower bounds in Section 3. In Section 4, we report our experimental results. Section 5 contains some concluding remarks.

2 Proof Complexity Preliminaries

In what follows, we give a brief overview of the relevant proof complexity background. This material is standard and we refer to, e.g., the survey [24] for more details. All formulas in this paper are in conjunctive normal form (CNF), i.e.,

consist of conjunctions of clauses, where a clause is a disjunction of positive literals (unnegated variables) and negative literals (negated variables, denoted by overline). It is convenient to view clauses as sets, so that there is no repetition of literals and order is irrelevant. A k-CNF formula has all clauses of size at most k, which is always assumed to be some fixed (and, in this paper, small) constant.

A *resolution refutation* $\pi : F \vdash \bot$ of a formula F (sometimes also referred to as a *resolution proof* for F) is an ordered sequence of clauses $\pi = (D_1, \ldots, D_\tau)$ such that $D_\tau = \bot$ is the empty clause without literals, and each line D_i, $1 \leq i \leq \tau$, is either one of the clauses in F (an *axiom* clause) or is derived from clauses D_j, D_k in π with $j, k < i$ by the *resolution rule* $\frac{B \vee x \quad C \vee \overline{x}}{B \vee C}$ (where $B \vee C$ is the *resolvent* of $B \vee x$ and $C \vee \overline{x}$ on x). It is also sometimes technically convenient to add a *weakening rule* $\frac{B}{B \vee C}$ that allows to add literals to a previously derived clause. The *length* (or *size*) $L(\pi)$ of a resolution refutation π is the number of clauses in π. The *width* $W(C)$ of a clause C is the number of literals $|C|$, and the width $W(\pi)$ of a refutation π is the width of a largest clause in π. Taking the minimum over all refutations of F, we obtain the length $L_{\mathcal{R}}(F \vdash \bot)$ and width $W_{\mathcal{R}}(F \vdash \bot)$ of refuting F, respectively. It is not hard to show that all use of weakening can be eliminated from refutations without increasing these measures.

Resolution can be extended with algebraic reasoning to yield the proof system *polynomial calculus resolution (PCR)*, or more briefly just *polynomial calculus*.[3] For some fixed field $\mathbb{F}$ (which would be GF(2) in practical applications but can be any field in theory) we consider the polynomial ring $\mathbb{F}[x, \overline{x}, y, \overline{y}, \ldots]$ with x and $\overline{x}$ as distinct formal variables, and translate clauses $\bigvee_{x \in L^+} x \vee \bigvee_{y \in L^-} \overline{y}$ to monomials $\prod_{x \in L^+} x \cdot \prod_{y \in L^-} \overline{y}$. A *PCR refutation* π of F is then an ordered sequence of polynomials $\pi = (P_1, \ldots, P_\tau)$, expanded out as linear combinations of monomials, such that $P_\tau = 1$ and each line P_i, $1 \leq i \leq \tau$, is one of the following:

- a monomial encoding a clause in F;
- a *Boolean axiom* $x^2 - x$ or *complementarity axiom* $x + \overline{x} - 1$ for any variable x;
- a polynomial obtained from one or two previous polynomials by *linear combination* $\frac{Q \quad R}{\alpha Q + \beta R}$ or *multiplication* $\frac{Q}{xQ}$ for any $\alpha, \beta \in \mathbb{F}$ and any variable x.

The *size* $S(\pi)$ of a PCR refutation π is the number of monomials in π (counted with repetitions) and the *degree* $Deg(\pi)$ is the maximal degree of any monomial appearing in π. Taking the minimum over all PCR refutations, we define the size $S_{\mathcal{PCR}}(F \vdash \bot)$ and degree $Deg_{\mathcal{PCR}}(F \vdash \bot)$ of refuting F in PCR. When the proof system is clear from context, we will drop the subindices denoting resolution or PCR, respectively. It is straightforward to show that PCR can simulate resolution efficiently by simply mimicking the resolution steps in a refutation, and this simulation can be done without any noticeable blow up in size/length or degree/width. There are formulas, however, for which PCR can be exponentially stronger than resolution with respect to size/length.

[3] Strictly speaking, PCR as defined in [1] is a slight generalization of polynomial calculus [12], but here we will not be too careful in distinguishing between the two and the term "polynomial calculus" will refer to PCR unless specified otherwise.

A *restriction* ρ on F is a partial assignment to the variables of F. In a restricted formula $F\restriction_\rho$ (or refutation $\pi\restriction_\rho$) all clauses satisfied by ρ are removed and all other clauses have falsified literals removed. It is a well-known fact that restrictions preserve resolution refutations, so that if π is a resolution refutation of F, then $\pi\restriction_\rho$ is a refutation of $F\restriction_\rho$ (possibly using weakening) in at most the same length and width. For polynomials, we think of 0 as true and 1 as false. Thus, if a restriction satisfies a literal in a monomial that monomial vanishes, and all falsified literals in a monomial get replaced by 1 and vanish. Again it holds that if π is a PCR refutation of F, then $\pi\restriction_\rho$ is a PCR refutation of $F\restriction_\rho$ (after a simple postprocessing step to take care of cancelling monomials and to adjust for that multiplication can only be done one variable at a time). This restricted refutation will have at most the same size and degree (except possibly for a constant factor in size due to postprocessing multiplications).

3 Theoretical Hardness Results

In this section, we present our proof complexity lower bounds. We will focus below on the formulas in [28]. The proof for the formulas in [26] is very similar in spirit but contains some further technical complications, and we defer the discussion of this to the end of this section. Let us start by giving an explicit, formal definition of these formulas, which we will refer to as *subset cardinality formulas*.

Definition 3 (Subset cardinality formula). *Suppose that $G = (U \dot\cup V, E)$ is a 4-regular bipartite (multi-)graph except that one extra edge has been added. Then the* subset cardinality formula *$SC(G)$ over G has variables $x_e, e \in E$, and clauses:*

- $x_{e_1} \vee x_{e_2} \vee x_{e_3}$ *for every triple e_1, e_2, e_3 of edges incident to $u \in U$,*
- $\overline{x}_{e_1} \vee \overline{x}_{e_2} \vee \overline{x}_{e_3}$ *for every triple e_1, e_2, e_3 of edges incident to $v \in V$.*

As noted before, an easy counting argument shows that these formulas are unsatisfiable. Intuitively, the hardness of proving this unsatisfiability should depend on the structure of the underlying graph G. We remind the reader that compared to [28], the "no quadrangles" condition mentioned in Section 1 is missing in Definition 3. This is because this condition is neither necessary nor sufficient to obtain lower bounds. Expressed in terms of the graph G, what quadrangle-freeness means is that there are no 4-cycles, which is essentially saying that no constraints in G have a very "localized structure." However, the fixed bandwidth formulas already discussed in Section 1 can be constructed to be quadrangle-free, but are still guaranteed to be easy for resolution. Therefore, in order for our lower bound proof to go through we need the more general condition that the graph G should be an expander as defined next.

Definition 4 (Expander). *A bipartite graph $G = (U \dot\cup V, E)$ is an (s, δ)-*expander *if for each vertex set $U' \subseteq U, |U'| \leq s$, it holds that $|N(U')| \geq \delta|U'|$, where $N(U') = \{v \in V \mid \exists (u, v) \in E \text{ for } u \in U'\}$ is the set of neighbours of U'.*

The key idea in our lower bound proof is to apply a suitably chosen restriction to reduce subset cardinality formulas to so-called *graph pigeonhole principle formulas* $PHP(G)$. These formulas are also defined in terms of bipartite graphs $G = (U \dot\cup V, E)$ and encode that every "pigeon" vertex on the left, i.e., in U, needs to have at least one of the edges incident to it set to true, while every "hole" vertex on the right, i.e., in V, must have at most one edge incident to it set to true. Ben-Sasson and Wigderson [6] showed that random instances of such formulas are hard for resolution if the left degree is at least 5, and modifying their techniques slightly we prove that left degree 3 is sufficient provided that the graphs have good enough expansion. The proof is by showing a resolution width lower bound and then applying the lower bound on length in terms of width in [6]. An analogous result can be proven also for polynomial calculus by using techniques from Alekhnovich and Razborov [2] to obtain a degree lower bound and then applying the lower bound on size in terms of degree in Impagliazzo et al. [17], which yields the following lemma.

Lemma 5. *Suppose that $G = (U \dot\cup V, E)$ is a 3-regular $\left(\epsilon n, \frac{3}{2} + \delta\right)$-expander for some constant $\epsilon, \delta > 0$ and $|U| = |V| = n$, and let G' be the graph obtained by removing any vertex from V in G and its incident edges. Then the resolution refutation length of the graph pigeonhole principle $PHP(G')$ is $\exp(\Omega(n))$, and the same bound holds for PCR size.*

Let us first show how Lemma 5 can be used to establish the lower bound for subset cardinality formulas and then present a proof of the lemma for the case of resolution. The argument for polynomial calculus is more involved and we will only be able to sketch it due to space constraints.

Theorem 6. *Suppose that $G = (U \dot\cup V, E)$ is a 4-regular $\left(\epsilon n, \frac{5}{2} + \delta\right)$-expander for $|U| = |V| = n$ and some constants $\epsilon, \delta > 0$, and let G' be obtained from G by adding an arbitrary edge from U to V. Then any polynomial calculus refutation of $SC(G')$ must have size $\exp\big(\Omega(n)\big)$ (and hence the same lower bound holds for resolution length).*

Proof. We want to restrict the subset cardinality formula $SC(G')$ to get a graph pigeonhole principle formula. By a standard argument, which can be found for instance in [10], we know that any 4-regular bipartite graph G has a perfect matching. Fix such a matching M and let $M' = M \cup \{(u', v')\}$, where (u', v') denotes the edge added to G. We apply the following restriction ρ to $SC(G')$:

$$\rho\left(x_{(u,v)}\right) = \begin{cases} \top & \text{if } (u,v) \in M', \\ \bot & \text{if } v = v' \text{ and } (u,v') \notin M', \\ * & \text{otherwise (i.e., the variable is unassigned).} \end{cases} \tag{1}$$

This reduces the original formula $SC(G')$ to $PHP(G'')$ on the graph G'' obtained by removing the matching M and also the vertex v' with incident edges from G. To see this, consider what happens with the clauses encoding the constraints.

For every vertex $u \in U \setminus \{u'\}$, which has four edges $e_i, 1 \leq i \leq 4$, incident to it, we have the clauses $\{x_{e_1} \vee x_{e_2} \vee x_{e_3}, x_{e_1} \vee x_{e_2} \vee x_{e_4}, x_{e_1} \vee x_{e_3} \vee x_{e_4}, x_{e_2} \vee x_{e_3} \vee x_{e_4}\}$ in $SC(G')$. After applying ρ, the one edge that is in the matching M will be set to true, satisfying all of these clauses but one. If in addition u is one of the vertices neighbouring v', the remaining constraint will shrink to a 2-clause. The constraint corresponding to u' is similarly reduced. In this case, we have five incident edges $e_i, 1 \leq i \leq 5$, and two of them are set to true. If, for instance, we have $e_4 \in M$ and $e_5 = (u', v')$, then the only clause that is not satisfied is $x_{e_1} \vee x_{e_2} \vee x_{e_3}$, which corresponds to the pigeon axiom for the vertex u' in G''.

For a constraint $v \in V \setminus \{v'\}$ with neighbours $e_i, 1 \leq i \leq 4$, the clause set is the same as for $U \setminus \{u'\}$ except that every variable is negated. If $e_4 \in M$, then after the restriction we are left with the set of clauses $\{\overline{x}_{e_1} \vee \overline{x}_{e_2} \vee \overline{x}_{e_3}, \overline{x}_{e_1} \vee \overline{x}_{e_2}, \overline{x}_{e_1} \vee \overline{x}_{e_3}, \overline{x}_{e_2} \vee \overline{x}_{e_3}\}$, where the last three clauses are the hole axioms for the vertex v in G'' and the first clause can be ignored since it follows by weakening of any of the other clauses. Since ρ satisfies the constraint v' the clauses encoding this constraint vanish. This shows that $SC(G')\restriction_\rho$ is indeed equal to $PHP(G'')$.

Now all that remains is to observe that G'' can be obtained by removing a right vertex from a 3-regular bipartite $(\epsilon n, \frac{3}{2} + \delta)$-expander. This is so since deleting the matching M from G decreases all vertex degrees from 4 to 3 and lowers the expansion factor by at most an additive 1. Applying Lemma 5 we conclude that $PHP(G'')$ requires polynomial calculus size (and hence resolution length) $\exp\big(\Omega(n)\big)$. As restrictions do not increase the length/size of refutations, the same lower bound must hold also for $SC(G')$.

It remains to prove Lemma 5. We give a full proof of the lemma for resolution below, but due to space constraints we can only outline the argument for polynomial calculus. For both resolution and polynomial calculus we need a stronger notion of expansion as defined next.

Definition 7 (Boundary expander). *A bipartite graph $G = (U \dot\cup V, E)$ is an* (s, δ)-boundary expander *if for every set of vertices $U' \subseteq U, |U'| \leq s$, it holds that $|\partial(U')| \geq \delta|U'|$, where $v \in \partial(U')$ if there is exactly one vertex $u \in U'$ that is a neighbour of v.*

Using the following connection between usual expansion and boundary expansion (which is straightforward to show and is stated here without proof) we can prove Lemma 5.

Proposition 8. *Every d-regular (s, δ)-expander is also an $(s, 2\delta - d)$-boundary expander.*

Proof (of Lemma 5 for resolution). Since G is an $(\epsilon n, 2\delta)$-boundary expander by Proposition 8, even after removing a vertex in V it must hold for G' that every set of vertices $U'' \subseteq U$, $|U''| \leq \epsilon n$ satisfies $|\partial_{G'}(U'')| \geq 2\delta|U'| - 1$.

Let us also observe that the connected component $G^c = (U^c \dot\cup V^c, E^c)$ of G to which the vertex v' belongs must be a 3-regular graph with $|U^c| > \epsilon n$. This is so since if $|U^c| \leq \epsilon n$, it would follow from the expansion of G that $|V^c| =$

$|N_{G^c}(U^c)| \geq \left(\frac{3}{2} + \delta\right)|U^c| > |U^c|$. But $|U^c| \neq |V^c|$ implies that G^c cannot be a 3-regular bipartite graph, which is a contradiction. Furthermore, for every proper subset $U'' \subsetneq U^c$ it must hold that $|N(U'')| > |U''|$, since otherwise U'' and its neighbours $N(U'')$ would form a disconnected component in G^c. Hence, when we remove the vertex v' from G^c we have $|N(U'')| \geq |U''|$ for every proper subset $U'' \subsetneq U^c$. By Hall's theorem, this implies that every proper subset $U'' \subsetneq U^c$ has a matching in G^c. This shows that any refutation of $PHP(G')$ must use all the pigeons in G^c, i.e., at least ϵn pigeon axiom clauses, to show that $PHP(G')$ is unsatisfiable, since the formula becomes satisfiable if just one of these pigeon axioms is removed.

Now we can employ the progress measure on refutations developed in [6] to show that the width of refuting $PHP(G')$ is lower bounded by $\epsilon\delta n - 1$. This follows by a straightforward adaptation of the argument in Sections 5 and 6.2 of [6], which yields a width lower bound analogous to that in Theorem 4.15. By appealing to the lower bound on length in terms of width in Corollary 3.6 in [6] we obtain a lower bound on the resolution refutation length of $\exp\left(\Omega(n)\right)$.

The proof of Lemma 5 for polynomial calculus is similar in that we prove a degree lower bound and then use the lower bound on size in terms of degree in [17] (which is an exact analogue of the result in [6] for resolution). We closely follow the proof of Theorem 4.14 in [2], from which one can derive that any refutation of the graph pigeonhole principle on an (s, δ)-boundary expander requires degree $\delta s/2$. This is almost what we need, except that we lose an additive 1 when we remove a vertex from the right. Nevertheless, the proof still goes through if we subtract 1 everywhere, yielding a degree lower bound of $\delta s/2 - 1$. The only point where we argue a bit differently than [2] is when we need to show that at least s pigeons have to be used to prove unsatisfiability. But we have already shown this claim in the proof of the resolution width lower bound and we can reuse the same argument in the proof of Lemma 5 for polynomial calculus.

This proves that the formulas in [28] are hard for polynomial calculus (and hence also for resolution) if the underlying graph is an expander. In order to establish that randomly sampled instances of such formulas are hard, we just need the fact that randomly sampled graphs are likely to be expanders. The following theorem tells us what we need to know.

Theorem 9 ([16]). *Let $d \geq 3$ be a fixed integer. Then for every $\delta, 0 < \delta < \frac{1}{2}$, there exists an $\epsilon > 0$ such that almost all d-regular bipartite graphs G with n vertices on each side are $\left(\epsilon n, d - \frac{3}{2} + \delta\right)$-expanders.*

Corollary 10. *The formula $SC(G)$ for a random 4-regular bipartite graph G with an arbitrary extra edge added requires polynomial calculus refutations (and hence also resolution refutations) of exponential size asymptotically almost surely.*

Proof. Use Theorem 9 with $d = 4$ together with Theorem 6.

Let us conclude this section by discussing how the lower bound proof above for the formulas in [28], i.e., subset cardinality formulas $SC(G)$ for ordinary graphs G, can be made to work also for the formulas in [26].

Following the description in Section 1, the formulas in [26] can be defined in terms of permutations of $[4n+1]$. We can construct a bipartite multigraph $G(\sigma)$ from a permutation σ of $[4n+1]$ as follows. We first partition $[4n+1]$ into subsets $\{1,2,3,4\}, \{5,6,7,8\} \ldots, \{4n-3, 4n-2, 4n-1, 4n, 4n+1\}$ and identify the vertices in V with these subsets. Second, by using the partition into subsets $\{\sigma(1), \sigma(2), \sigma(3), \sigma(4)\}, \ldots, \{\sigma(4n-3), \sigma(4n-2), \sigma(4n-1), \sigma(4n), \sigma(4n+1)\}$ we obtain the vertices in U. Then, for every number that is in both in $u \in U$ and $v \in V$ we add an edge between u and v. In this way, we obtain a subset cardinality formula on the multigraph $G(\sigma)$, which we will denote by $SC^*(G(\sigma))$ (or more briefly just $SC^*(\sigma)$) to highlight that the formula is generated from a multigraph obtained from a permutation.

In order to show that the formula $SC^*(\sigma)$ requires polynomial calculus refutations of exponential size if the multigraph $G(\sigma)$ is an expander, there are three issues we need to address in the proof for standard graphs above:

- Firstly, our graph theoretic claims should now be made for multigraphs instead of ordinary graphs. This is not a problem, however, since all the claims we need are still true in this setting and since multiple copies of an edge can be eliminated by setting the corresponding variables to false.
- Secondly, the degree-5 vertices are not necessarily connected in $G(\sigma)$. Because of this, we need to modify the restriction used to reduce the formula to a graph pigeonhole principle formula. We still find a matching and set all the edges in it to true, but now we choose two special edges incident to the degree-5 vertices u' and v' and set their values to true and false, respectively. In this way the graph we get has a vertex on the right with two of the edges incident to it set to true, one from the matching and one from the special edge incident to u'. This forces the values of the remaining two edges to be false, which satisfies the constraint v' and gives us the graph pigeonhole principle formula required by Lemma 5.
- Thirdly, a slightly subtle point is that we do not require $G(\sigma)$ to be expanding, but rather a slightly modified multigraph $G^M(\sigma)$. To form $G^M(\sigma)$, we start by removing from $G(\sigma)$ one of the edges incident to the degree-5 vertex in U and one of the edges incident to the degree-5 vertex in V. The resulting multigraph has two vertices of degree 3, which we connect with an edge in order to form the 4-regular multigraph $G^M(\sigma)$.

In order to show that randomly sampled instances of the formulas $SC^*(\sigma))$ are hard, we note that the model of a random graph used to prove Theorem 9 is actually based on random permutations. Hence, the claim that random 4-regular graphs are good expanders holds for random permutations as well, which implies that almost all instances of the formulas in [26] are hard for polynomial calculus.

Theorem 11. *Let σ be a permutation of $[4n+1]$ and let $SC^*(\sigma)$ be the corresponding subset cardinality formula. If the multigraph $G^M(\sigma)$ is an $\left(\epsilon n, \frac{5}{2}+\delta\right)$-expander for some constants $\epsilon, \delta > 0$, then any polynomial calculus refutation (and resolution refutation) of $SC^*(\sigma)$ has size $\exp\left(\Omega(n)\right)$.*

In particular, for a random permutation σ the formula $SC^*(\sigma)$ requires polynomial calculus refutation of exponential size asymptotically almost surely.

4 Empirical Results on SAT Solver Performance

For our experiments we used the SAT solvers Glucose 2.2 [14], March-rw [19], and Lingeling-ala [18]. The experiments were run under Linux on a computer with two quad-core AMD Opteron 2.2 GHz CPUs (2374 HE) and 16 GB of memory, where only one solver was running on the computer at any given time. We limited the solver running time to 1 hour per instance . For the experiments with fixed variable ordering we used a version of MiniSat 2.2.0 [22] modified so that the solver always branches on unset variable in fixed order.

The CNF formula instances were obtained as follows:

1. The formulas $SC^*(\sigma)$ from [26] were generated by taking one fixed partition of $[4n+1]$ into $\{1,2,3,4\}$, $\{5,6,7,8\}$, ..., $\{4n-3,4n-2,4n-1,4n,4n+1\}$ and one random partition into 4-groups plus one 5-group, and then encoding positive and negative cardinality constraints, respectively, on these two partitions.
2. For the formulas $SC(G)$ from [28] we started with a random (non-bipartite) 4-regular graph, took the bipartite double cover (with two copies v_L, v_R of each vertex v and edges (u_L, v_R) for all edges (u,v) in the original graph), and finally added a random edge.[4]
3. The fixed bandwidth formulas were constructed from an $n \times n$ matrix with ones in the first row on positions $1,2,4,8$ and zeroes everywhere else, and with every subsequent row being a cyclic shift one step to the right of the preceding row. Finally, an extra one was added to the top right cell of the matrix if this was a zero, and otherwise to the nearest cell containing a zero.[5]

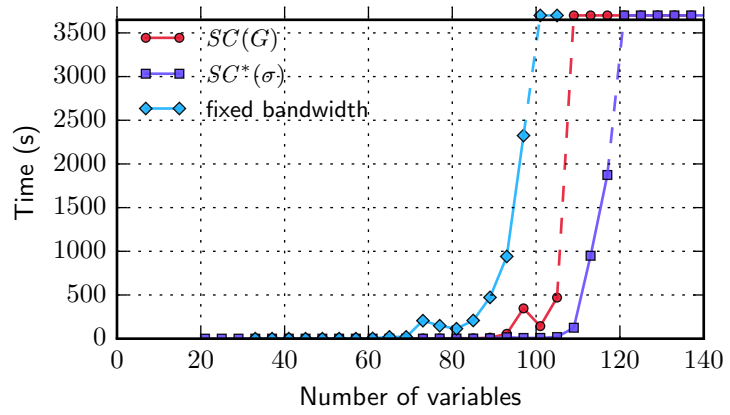
For each CNF formula we ran each SAT solver three times (with different random seeds), and for randomly generated formulas we ran on three different CNF formulas for each parameter value. The values in the plots are the medians of these values. We also performed exactly the same set of experiments on randomly shuffled version of the formulas (with randomly permuted clauses, variables, and polarities), but this random shuffling did not affect the results in any significant way and so we do not display these plots.

We present the results of our experiments in Figure 1 with one subplot per solver.[6] As can be seen from these plots, all three versions of the formulas become infeasible for around 100–120 variables. Comparing to our experiments on random 3-CNF formulas and Tseitin formulas on random 3-regular graphs in Figure 2, it should be clear that all three flavours of the formulas from [26, 28] that we investigated were substantially harder than random formulas. Notice

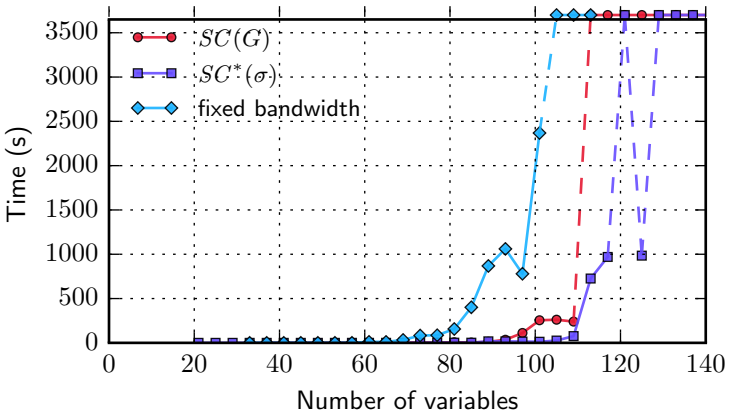
[4] We remark that, strictly speaking, this does not yield uniformly random instances but we just wanted to obtain some instances with "good enough" randomness (and hence expansion) on which we could run experiments.

[5] Note that this construction yields quadrangle-free instances for large enough n, except possibly for quadrangles involving the added extra top-right entry.

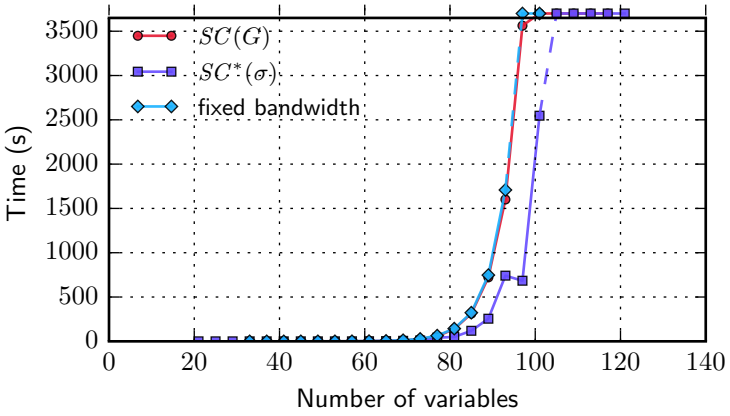
[6] The code for generating the CNF instances and complete data for the experiments can be found at `http://www.csc.kth.se/~jakobn/publications/sat14/`.



(a) Glucose

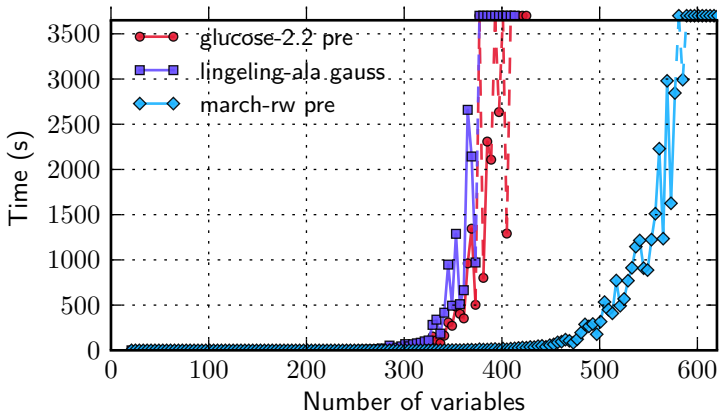


(b) Lingeling

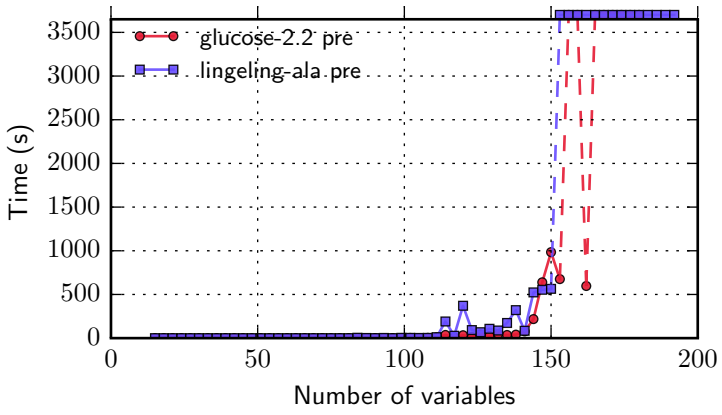


(c) March

Fig. 1. SAT solver performance on variants of the formulas in [26, 28].



(a) Random 3-CNF formulas



(b) Tseitin formulas on random graphs

Fig. 2. SAT solver performance on two well-known hard formula families.

that for Tseitin formulas we do not present results for March and that Lingeling was run without Gaussian elimination. The reason is that March and Lingeling with Gaussian elimination solve Tseitin formulas in less than a second for even the largest instances we have tried.

Comparing random instances of formulas $SC^*(\sigma)$ and $SC(G)$ with fixed bandwidth instances, we can see that the easiest ones are $SC^*(\sigma)$ while $SC(G)$ are somewhat harder. This is as expected—by construction, for $SC(G)$ we are guaranteed that no pair of positive and negative constraints share more than one variable, whereas for $SC^*(\sigma)$ it could happen in principle that a positive and a negative constraint act on two, three, or even four common variables. Some-

what counter-intuitively, however, the instances that are hardest in our practical experiments are the theoretically easy fixed bandwidth formulas.

In order to investigate whether the hardness of fixed bandwidth formulas could be attributed to hidden constants in the asymptotics—i.e., that the polynomial upper bounds on resolution length are so large in practice that the fixed bandwidth formulas are infeasible for all practical purposes—we ran a modified version of MiniSat on these formulas which always branched on variables row by row and in every row column by column. Intuitively, this seems to be the appropriate variable ordering if one is to recover the polynomial-length resolution refutation presented in [28]. And indeed, MiniSat run on fixed bandwidth formulas with fixed variable ordering performed much better than any of the other solvers on random instances of $SC^*(\sigma)$ and $SC(G)$ formulas. (We also verified that fixed variable ordering is not a good idea in general—as expected, MiniSat with fixed variable ordering performs poorly on random instances of $SC^*(\sigma)$ and $SC(G)$ formulas.)

Given the latest advances in SAT solving technology, with solvers going beyond resolution by incorporating elements of algebraic reasoning (Gröbner bases) and geometric reasoning (pseudo-Boolean solvers), a natural question is whether the formulas in [26, 28] remain hard for such solvers.

Regarding algebraic solvers, we are not aware of any general-purpose solvers that can compete with CDCL solvers, but as mentioned the theoretical lower bounds that we prove for resolution hold also for polynomial calculus, which is a proof system for formalizing the reasoning in solvers based on Gröbner basis computations. Also, one can note that the algebraic reasoning in terms of Gaussian elimination in Lingeling does not seem to help.

For pseudo-Boolean solvers, which can be seen to search for proofs in (more or less restricted version of) the cutting planes proof system [13], the story could potentially be very different. As noted multiple times already, the formulas $SC^*(\sigma)$ and $SC(G)$ are just encodings of a fairly simple counting principle, and in contrast to resolution and polynomial calculus the cutting planes proof system knows how to count. Thus, pseudo-Boolean solvers with enough well-developed methods of cardinality constraints reasoning should have the potential to solve these formulas quickly. This indeed appears to be the case as reported in [8], and our own (albeit limited) experiments also show this.

5 Concluding Remarks

In this work, we establish that the formulas constructed by Spence [26] and Van Gelder and Spence [28] are exponentially hard for resolution and also for polynomial calculus resolution (PCR), which extends resolution with Gröbner basis computations. Formally, we prove that if the bipartite (multi-)graph describing the constraints encoded by the formula is expanding, then this implies exponential lower bounds on proof size in resolution and PCR. Furthermore, we show that random instances of these formulas are almost surely expanding, meaning that the exponential lower bound applies with high probability.

We also investigate the performance of some current state-of-the-art SAT solvers on these formulas, and find that small instances are indeed much harder than, e.g., random 3-CNF formulas with the same number of variables. Somewhat surprisingly, however, the very hardest formulas in our experiments are versions of the formulas in [26, 28] generated from fixed bandwidth matrices. This is intriguing, since such formulas are easy for resolution, and since the current conventional wisdom (based on [3, 25]) seems to be that CDCL solvers can search efficiently for short resolution proofs. In view of this, an interesting (albeit very speculative) question is whether perhaps these fixed bandwidth matrix formulas could be used to show formally that CDCL with VSIDS, 1UIP, and phase saving, say, does *not* polynomially simulate resolution.

Since the formulas in [26, 28] encode what is in essence a fairly simple counting argument, SAT solvers that can reason efficiently with cardinality constraints could potentially solve these formulas fast. This indeed turns out to be the case for the latest version of Sat4j [8]. It would be interesting to investigate whether the formulas in [26, 28] could be slightly obfuscated to make them hard also for solvers with cardinality constraints. If so, this could yield small benchmark formulas that are hard not only for standard CDCL solvers but also for solvers extended with algebraic and/or geometric reasoning.

Another candidate construction of small but very hard CNF formulas is the one presented by Markström [20]. It would be interesting to investigate what theoretical hardness results can be established for these formulas (for resolution and proof systems stronger than resolution) and how the practical hardness scales compared to the constructions by Spence and Van Gelder [26, 28]. In particular, an interesting question is whether these formulas, too, become easy for CDCL solvers enhanced with cardinality constraints reasoning.

Acknowledgements

The authors are very grateful to Massimo Lauria and Marc Vinyals for stimulating discussions and for invaluable practical help with setting up and evaluating the experiments. We wish to thank Niklas Sörensson for explaining how to fix the variable decision order in MiniSat, and Daniel Le Berre for sharing data about the performance of the latest version of Sat4j on our benchmark formulas. We are also grateful to Allen Van Gelder for comments on a preliminary write-up of some of the results in this paper, as well as for introducing us to this problem in the first place. Finally, we thank several participants of the workshop *Theoretical Foundations of Applied SAT Solving (14w5101)* at the Banff International Research Station in January 2014 for interesting conversations on themes related to this work.

The authors were funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611. The second author was also supported by the Swedish Research Council grants 621-2010-4797 and 621-2012-5645.

References

1. Alekhnovich, M., Ben-Sasson, E., Razborov, A.A., Wigderson, A.: Space complexity in propositional calculus. SIAM Journal on Computing 31(4), 1184–1211 (2002), preliminary version appeared in *STOC '00*
2. Alekhnovich, M., Razborov, A.A.: Lower bounds for polynomial calculus: Non-binomial case. Proceedings of the Steklov Institute of Mathematics 242, 18–35 (2003), available at `http://people.cs.uchicago.edu/~razborov/files/misha.pdf`. Preliminary version appeared in *FOCS '01*.
3. Atserias, A., Fichte, J.K., Thurley, M.: Clause-learning algorithms with many restarts and bounded-width resolution. Journal of Artificial Intelligence Research 40, 353–373 (Jan 2011), preliminary version appeared in *SAT '09*
4. Bayardo Jr., R.J., Schrag, R.: Using CSP look-back techniques to solve real-world SAT instances. In: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97). pp. 203–208 (Jul 1997)
5. Beck, C., Impagliazzo, R.: Strong ETH holds for regular resolution. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13). pp. 487–494 (May 2013)
6. Ben-Sasson, E., Wigderson, A.: Short proofs are narrow—resolution made simple. Journal of the ACM 48(2), 149–169 (Mar 2001), preliminary version appeared in *STOC '99*
7. Berre, D.L., Parrain, A.: The Sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation 7, 59–64 (2010), system description
8. Biere, A., Le Berre, D., Lonca, E., Manthey, N.: Detecting cardinality constraints in CNF. In: Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14) (Jul 2014), to appear
9. Blake, A.: Canonical Expressions in Boolean Algebra. Ph.D. thesis, University of Chicago (1937)
10. Bondy, J.A., Murty, U.S.R.: Graph Theory. Springer (2008)
11. Chvátal, V., Szemerédi, E.: Many hard examples for resolution. Journal of the ACM 35(4), 759–768 (Oct 1988)
12. Clegg, M., Edmonds, J., Impagliazzo, R.: Using the Groebner basis algorithm to find proofs of unsatisfiability. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96). pp. 174–183 (May 1996)
13. Cook, W., Coullard, C.R., Turán, G.: On the complexity of cutting-plane proofs. Discrete Applied Mathematics 18(1), 25–38 (Nov 1987)
14. The Glucose SAT solver. `http://www.labri.fr/perso/lsimon/glucose/`
15. Haken, A.: The intractability of resolution. Theoretical Computer Science 39(2-3), 297–308 (Aug 1985)
16. Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bulletin of the American Mathematical Society 43(4), 439–561 (Oct 2006)
17. Impagliazzo, R., Pudlák, P., Sgall, J.: Lower bounds for the polynomial calculus and the Gröbner basis algorithm. Computational Complexity 8(2), 127–144 (1999)
18. Lingeling and Plingeling. `http://fmv.jku.at/lingeling/`
19. March. `http://www.st.ewi.tudelft.nl/~marijn/sat/march_dl.php`
20. Markström, K.: Locality and hard SAT-instances. Journal on Satisfiability, Boolean Modeling and Computation 2(1-4), 221–227 (2006)
21. Marques-Silva, J.P., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Transactions on Computers 48(5), 506–521 (May 1999), preliminary version appeared in *ICCAD '96*

22. The MiniSat page. `http://minisat.se/`
23. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: Proceedings of the 38th Design Automation Conference (DAC '01). pp. 530–535 (Jun 2001)
24. Nordström, J.: Pebble games, proof complexity and time-space trade-offs. Logical Methods in Computer Science 9, 15:1–15:63 (Sep 2013)
25. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning SAT solvers as resolution engines. Artificial Intelligence 175, 512–525 (Feb 2011), preliminary version appeared in *CP '09*
26. Spence, I.: sgen1: A generator of small but difficult satisfiability benchmarks. Journal of Experimental Algorithmics 15, 1.2:1.1–1.2:1.15 (Mar 2010)
27. Urquhart, A.: Hard examples for resolution. Journal of the ACM 34(1), 209–219 (Jan 1987)
28. Van Gelder, A., Spence, I.: Zero-one designs produce small hard SAT instances. In: Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT '10). Lecture Notes in Computer Science, vol. 6175, pp. 388–397. Springer (Jul 2010)