Exponential Lower Bounds for the PPSZ k-SAT Algorithm*†

Shiteng Chen‡, Dominik Scheder§, Navid Talebanfard§, Bangsheng Tang‡

Abstract

In 1998, Paturi, Pudlák, Saks, and Zane presented PPSZ, an elegant randomized algorithm for k-SAT. Fourteen years on, this algorithm is still the fastest known worst-case algorithm. They proved that its expected running time on k-CNF formulas with n variables is at most $2^{(1-\epsilon_k)n}$, where $\epsilon_k \in \Omega(1/k)$. So far, no exponential lower bounds at all have been known.

In this paper, we construct hard instances for PPSZ. That is, we construct satisfiable k-CNF formulas over n variables on which the expected running time is at least $2^{(1-\epsilon_k)n}$, for $\epsilon_k \in O(\log^2 k/k)$.

1 Introduction

In 1998, Paturi, Pudlák, Saks, and Zane [PPSZ98, PPSZ05] presented an elegant randomized algorithm for the satisfiability problem of k-CNF formulas with n variables. Their algorithm, henceforth called PPSZ, is conceptually extremely simple to state:

> Pick a variable x randomly from the variables of F. Try to determine the correct value of x using some heuristic. If that fails, choose the value of x randomly. Fix it to this value. Repeat and good luck.

The heuristic they use checks whether the correct value can be determined by *resolution of bounded width*: that is, it builds all possible resolvents from the clauses of the formula, but throws away those resolvents whose size exceeds some parameter w. If at some point this leads to the unit clause $\{x\}$, the algorithm knows that it must set x to 1. If the unit clause $\{\bar{x}\}$ appears, surely x must be 0. This procedure runs in time $2^{o(n)}$ as long as $w \in o(n)$. Let us call this heuristic the *strong heuristic*, denoted by P^{strong}.

*The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and from the CFEM research center (supported by the Danish Strategic Research Council), within which this work was performed.

†This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174, 61150110582.

‡Tsinghua University

§Aarhus University

We also consider a weaker heuristic, P^{weak}. Instead of trying to derive $\{x\}$ or $\{\bar{x}\}$ using small-width resolution, it iterates over all subformulas G of F containing up to w clauses, and checks whether G implies x or $\bar{x}$ in the logical meaning: G *implies* x if all satisfying assignments of G set x to 1. By completeness of resolution, $\{x\}$ (or $\{\bar{x}\}$) follows from G by a resolution proof as well in which all clauses have size at most wk, simply because G cannot contain more than wk variables. This means that P^{strong} with clauses up to width wk is at least as strong as P^{weak}, considering subformulas up to size w. This justifies the terms weak and strong.

In general, a heuristic P takes as input a formula F and a variable x, and returns 0, 1, or "don't know". We require that P be *correct*: if $P(F, x) = 1$, then F implies x. If $P(F, x) = 0$, then F implies $\bar{x}$. Note that both P^{weak} and P^{strong} fullfil this requirement.

1.1 PPSZ as Encoding Satisfying Assignments

Now we discuss PPSZ from a particular perspective. The function `encode` below takes a satisfiable CNF formula F, a satisfying assignment $\mathbf{b}$ of it, a permutation π of the variables of F, and a heuristic P and outputs an encoding $\mathbf{c}$ of $\mathbf{b}$. $F^{[x_i \mapsto b_i]}$ is F simplified by replacing x_i by b_i (will be defined more formally later). It is not hard to see that one can reverse this procedure; that is, given $\mathbf{c}$, F, and π, one can reconstruct the original $\mathbf{b}$. The encoding $\mathbf{c}$ is never longer than the original $\mathbf{b}$. It might be much shorter.

ALGORITHM 1.1. `encode`($\mathbf{b}$, π, F, P)

```
1: c := the empty string
2: for x_i ∈ vbl(F) in the order according to π do
3:    if P(F, x_i) ="don't know" then
4:       append b_i to c
5:    end if
6:    F := F^[x_i ↦ b_i]
7: end for
8: return c
```

During a call to $\texttt{encode}(\mathbf{b}, \pi, F, P)$, some variables of F are encoded into the result $\mathbf{c}$, whereas some others are skipped, because the heuristic P was able to deduce their correct values. Let $\text{EncVbl}(\mathbf{b}, \pi, F, P)$ denote the set of variables that have been encoded. So the length of the string $\texttt{encode}(\mathbf{b}, \pi, F, P)$ equals the size of the set

$\mathrm{EncVbl}(\mathbf{b}, \pi, F, P)$. From $\mathbf{c}$, we can reconstruct $\mathbf{b}$: there is a procedure $\mathtt{decode}$ such that $\mathtt{decode}(\mathbf{c}, \pi, F, P) = \mathbf{b}$.

If we do not know $\mathbf{c}$, we might just take a random bit string and hope that $\mathbf{c}$ is a prefix of it. This happens with probability $2^{-|c|}$. This suggests the following probabilistic algorithm for SAT: let $\mathbf{c}$ be a random bit string, try to decode it, and hope that it works.

ALGORITHM 1.2. $\mathtt{ppsz}(F, \pi, P)$

1: $\mathbf{c} \leftarrow \{0,1\}^n$, uniformly at random
2: $\mathbf{b} := \mathtt{decode}(\mathbf{c}, \pi, F, P)$
3: **return b**

In most cases the algorithm will return junk, but if we are lucky, it will return a satisfying assignment $\mathbf{b}$. The following lemma is not difficult to prove:

LEMMA 1.1. ([PPSZ05]) *Let* $\mathbf{b}$ *be a satisfying assignment of* F. *Then* $\Pr[\mathtt{ppsz}(F, \pi, P) = \mathbf{b}] = 2^{-|\mathtt{encode}(\mathbf{b}, \pi, F, P)|}$.

We see that short encodings imply high success probability. We do not know, however, which permutation π leads to a short codelength. Our algorithm will simply choose a random permutation. Since $x \mapsto 2^x$ is convex, Jensen's inequality implies that

$$\mathop{\mathbb{E}}_{\pi}[\Pr[\mathtt{ppsz}(F, \pi, P) = \mathbf{b}]] \geq 2^{-\mathbb{E}_{\pi}|\mathtt{encode}(\mathbf{b}, \pi, F, P)|}.$$

The main theorem of Paturi, Pudlák, Saks, and Zane is:

THEOREM 1.1. ([PPSZ05]) *Let* F *be a satisfiable* k*-CNF formula and* $\mathbf{b}$ *be a satisfying assignment. Then* $\mathbb{E}_{\pi}[|\mathtt{encode}(\mathbf{b}, \pi, F, P)|] \leq (1 - \epsilon_k)n$, *where* $\epsilon_k \approx \frac{\pi^2}{6k}$ *for large* k.

For a 3-CNF formula F that has a *unique* satisfying assignment $\mathbf{b}$, they show that $\mathbb{E}_{\pi}[\mathtt{encode}(\mathbf{b}, \pi, F, P)] \leq (2\ln 2 - 1)n + o(1) \approx 0.38n$; this gives a randomized algorithm of running time 1.308^n for 3-SAT. Recently, Timon Hertli [Her11] proved that this algorithm achieves the same running time bound even if the 3-CNF formula has many satisfying assignments.

In Paturi, Pudlák, Saks, and Zane [PPSZ05] and Hertli [Her11], it suffices to consider the weak heuristic (even if [PPSZ05] formulates the algorithm using the strong heuristic). In fact, it suffices to check implication by subformulas of up to D clauses, where $D = w(n)$ can be an arbitrarily slowly growing function of n.

1.2 Results All known positive results about PPSZ, i.e., upper bounds on its expected running time, also hold for a very weak heuristic. We complement this with a lower bound on the running time that holds for the strong heuristic P^{strong}. This heuristic checks implication by resolution of width bounded by w for $w \leq \ln(k)n/k$.

THEOREM 1.2. (LOWER BOUND FOR PPSZ) *For every sufficiently large* $k \in \mathbb{N}$, *there exists a family* F_n *of satisfiable* k*-CNF formulas over* n *variables such that*

$$\Pr[\mathtt{ppsz}(F, P^{\mathrm{strong}}) \text{ is successful}] \leq 2^{-(1-\epsilon_k)n}$$

for $\epsilon_k \in O\left(\frac{\log^2 k}{k}\right)$.

In other words, the "savings" are $O(\log^2 k/k)$. Contrast this with the upper bound in [PPSZ05], stating that the savings are at least $\Omega(1/k)$.

Relation to ETH and Strong ETH Most complexity theorists believe that there is no polynomial time algorithm for k-SAT. But this does not rule out the possibility of a $2^{\sqrt{n}}$ time algorithm on formulas with n variables. Such an algorithm does not run in polynomial time, but it is far better than everything that is known. In fact, most researchers do not believe in such an algorithm either: The Exponential Time Hypothesis (ETH), a conjecture formulated by Impagliazzo, Paturi, and Zane [IPZ01] states that no algorithm solves 3-SAT in time $2^{o(n)}$. In fact, they prove that if 3-SAT can be solved in subexponential time, then so can k-SAT for every $k \geq 3$.

Far from being subexponential, the best known algorithms for k-SAT exhibit a running time of the form $2^{(1-\epsilon_k)n}$, where $\epsilon_k \to 0$ as k grows. In words, for large k they are not much better than exhaustive search. The conjecture that this is necessarily so is known as *Strong ETH*.

Looking into the running time in even more detail, we see that several k-SAT algorithms exhibit savings ϵ_k of order $\Omega(1/k)$: this is true for PPZ [PPZ99] and Schöning's algorithm [Sch99]. In fact, no algorithm, deterministic or randomized, of running time $2^{(1-\omega(1/k))n}$ is known. This means that when constructing hard instances, one should not only strive for exponential lower bounds: one should try to obtain bounds of the form $2^{(1-\epsilon_k)n}$, where the savings ϵ_k are not much larger than $1/k$. Providing such lower bounds for the PPSZ algorithm is the contribution of this paper.

It is also interesting to investigate the performance of PPSZ on well-structured instances, e.g. instances with small tree-widths. It is shown in [ACL+12] that satisfiability of a formula ϕ with tree-width $\mathsf{tw}(\phi)$ can be determined in $O^*(2^{\mathsf{tw}(\phi)})$ time by a specially tailored algorithm, and this is essentially optimal assuming ETH. Here we assume that $\mathsf{tw}(\phi)$ is asymptotically smaller than n, e.g. $\log^{O(1)} n$. Based on our hard instances for PPSZ, we can construct an instance with tree-width $\mathsf{tw}(\phi)$ on which PPSZ performs poorly: joining $n/\mathsf{tw}(\phi)$ disjointed hard instances for PPSZ on $\mathsf{tw}(\phi)$ variables. The success probability of PPSZ on this instance is the product of the success probabilities

of PPSZ on the disjointed hard instances. It is not difficult to see, on this instance, PPSZ needs $O(2^{\Omega(n)})$ time to answer with a constant success probability, which is worse than $O^*(2^{\mathrm{tw}(\phi)})$.

Methods and Challenges Resolution is a calculus for proving unsatisfiability of CNF formulas. It has been widely studied, and many exponential lower bounds are known. See Ben-Sasson and Wigderson [BSW01] for example. Resolution lower bounds are important because, among other things, they immediately translate into lower bounds for the so-called DPLL style algorithm (named after Davis, Putnam, Logemann, and Loveland [DP60, DLL62]) on unsatisfiable formulas. Alekhnovich, Hirsch, and Itsykson [AHI05] prove exponential lower bounds for DPLL algorithms on satisfiable formulas, using methods from resolution lower bounds. Unfortunately, the PPSZ algorithm does not fit into the framework of Alekhnovich, Hirsch, and Itsykson: it is not a DPLL algorithm. Therefore, resolution methods do not directly apply. Our proof is, however, to some extent inspired by their work.

A second challenge is to prove lower bounds of the form $2^{(1-\epsilon_k)n}$ for $\epsilon_k \to 0$. Existence of such lower bounds for DPLL algorithms is shown in [PI00]. In resolution lower bounds, a central notion is the *width* of a resolution derivation, which is the size of the largest clause appearing in it. Though there are *linear* lower bounds on the resolution width for k-CNF formulas (see again Ben-Sasson and Wigderson [BSW01]), no lower bound beyond $n/2 + o(n)$ is known. In fact, if the k-CNF formula is derived from a system $A \cdot \mathbf{x} = \mathbf{b}$ of linear equations, where A has at most k 1's in each row, then the resolution width is at most $n/2 + o(n)$: Ben-Sasson and Impagliazzo [BSI10] show that such formulas always have resolution refutations of width at most $n/2 + o(n)$ and size at most $2^{n/2+o(n)}$. Our hard instances are based on linear systems too; we therefore cannot use resolution width lower bounds directly if we want to prove bounds above $2^{n/2}$.

Following the framework of Ben-Sasson and Wigderson [BSW01], one proves width lower bounds by (i) defining a certain notion of *boundary expansion* on the formula, (ii) showing that a small-width resolution proof leads to a poorly expanding subformula, and (iii) showing that this contradicts the expansion properties of the formula. Unfortunately, this expansion method will not give us any width lower bounds close to n. Instead, we show that from a successful run of PPSZ, we obtain many subformulas $F_1, F_2, \ldots$ that are poorly expanding in a very particular way: there is a set S such that the "boundary" of each F_i falls almost completely into S. We can show that for a formula with good expansion properties, this is not possible, even if $|S|$ is very close to n.

Another challenge is more specific for PPSZ. Paturi, Pudlák, Saks, and Zane [PPSZ05] and Timon Hertli [Her11] use Jensen's inequality to bound $\mathbb{E}_\pi\left[2^{-|\mathtt{encode}|}\right] \geq 2^{-\mathbb{E}[|\mathtt{encode}|]}$. The latter expression is much more accessible. For us, the inequality goes in the wrong direction. If $|\mathtt{encode}(\mathbf{b}, \pi, F, P)|$ is $o(n)$ for even a single permutation π, then PPSZ's success probability is at least $2^{-o(n)}$. We need to make sure that $|\mathtt{encode}(\mathbf{b}, \pi, F, P)|$ is large for *every* $\mathbf{b}$ and *every* π. We define

$$\mathrm{codelength}(\mathbf{b}, F, P) := \min_\pi |\mathtt{encode}(\mathbf{b}, F, \pi, P)|$$

and

$$\mathrm{codelength}(F, P) := \min_{\mathbf{b}} \mathrm{codelength}(\mathbf{b}, F, P) \ ,$$

where the minimum is taken over all satisfying assignments $\mathbf{b}$.

THEOREM 1.3. *For every sufficiently large $k \in \mathbb{N}$, there is a family F_n of satisfiable k-CNF formulas over n variables such that*

$$\mathrm{codelength}(F, P^{\mathrm{strong}}) \geq (1 - \epsilon_k)n$$

for some $\epsilon_k \in O((\ln k)^2/k)$, and F_n has at most $2^{\epsilon_k n}$ satisfying assignments.

By Lemma 1.1, this means that the success probability of PPSZ on such a formula is at most $\sum_{\mathbf{b}\in \mathrm{sat}(F)} 2^{-\mathrm{codelength}(\mathbf{b})} \leq 2^{-(1-\epsilon_k)n}$. Thus, Theorem 1.3 implies Theorem 1.2.

1.3 Notation Here we introduce some notation that will be used throughout the remaining part of the paper. For a vector $\mathbf{v}$ over $\mathbb{F}_2$, we denote by $\mathrm{supp}(\mathbf{v})$ the set $\{i \mid \mathbf{v}_i = 1\}$. For a matrix M, $\mathrm{supp}(M)$ is the union of $\mathrm{supp}(\mathbf{r})$ over all rows $\mathbf{r}$ of M. For a CNF formula F, $\mathrm{vbl}(F)$ denotes the set of variables occurring there. For a literal u, $\mathrm{vbl}(u)$ denotes the one underlying variable. For a formula F, a variable $x \in \mathrm{vbl}(F)$ and $b \in \{0, 1\}$, $F^{[x\mapsto b]}$ denotes the simplied F after setting the value of x to the constant b. Furthermore, for a subset S of variables and an assigment $\mathbf{b}$, $F^{[x_S\mapsto \mathbf{b}_S]}$ is the simplified formula F after setting the value of variables in S according to $\mathbf{b}$. The binomial distribution of n experiments with success probability p will be denoted by $\mathcal{B}in(n, p)$.

2 The Hard Instances

We base our construction of hard instances on linear equations of the form $M \cdot \mathbf{x} = 0$. Form $M \in \mathbb{F}_2^{n\times n}$ by

setting each entry to 1 with probability k/n. One can write the system $M \cdot x = 0$ as a CNF formula by writing each constraint $\mathbf{m} \cdot \mathbf{x} = 0$ as an equivalent CNF formula. The following theorem holds.

THEOREM 2.1. *Let F denote the CNF formula encoding the system $M \cdot \mathbf{x} = 0$. Then with high probability over the choice of M,*

$$\text{codelength}(F, P^{\text{strong}}) \geq (1 - \epsilon_k)n$$

for some $\epsilon_k \in O((\ln k)^2/k)$.

2.1 Transformation into a k-CNF formula The bulk of the work is to prove Theorem 2.1. In addition, we have to show (i) that $M \cdot \mathbf{x} = 0$ does not have too many solutions and (ii) how to transform $M \cdot \mathbf{x} = 0$ into a k-CNF formula. Both (i) and (ii) are rather straightforward. We state the needed key lemmas and give the proofs in the appendix.

PROPOSITION 2.1. *With high probability over the choice of M, the system $M \cdot \mathbf{x} = 0$ has at most $2^{\epsilon_k n}$ solutions, where $\epsilon_k \in O(\ln^2(k)/k)$.*

PROPOSITION 2.2. *Let F be a CNF formula over n variables, $T \subseteq [n]$ a set of variables, and $\mathbf{b}^*$ be a satisfying assignment. Set $F' := F^{[x_T \mapsto \mathbf{b}^*_T]}$. Then* $\text{codelength}(F', P^{\text{strong}}) \geq \text{codelength}(F, P^{\text{strong}}) - |T|$.

PROPOSITION 2.3. *Choose $M \in \mathbb{F}_2^{n \times n}$ by setting every entry to 1 with probability k/n. Call a row* excessive *if it has more than $2ek$ 1's. The expected total number of 1's in the excessive rows is $O(nk/2^k)$.*

Combining these two lemmas, we take our formula F, which is not a k-CNF formula, and set to 0 all the variables that occur in clauses of size bigger than $2ek$. The resulting formula F' is a $2ek$-CNF formula, and by Lemma **??**,

$$\begin{aligned} \text{codelength}(\mathbf{b}', F') &\geq (1 - O(\ln^2 k/k) - O(k/2^k))n \\ &\geq (1 - O(\ln^2 k/k))n \end{aligned}$$

for all satisfying assignments $\mathbf{b}'$ of F'. Thus, Theorem 1.3 follows from Theorem 2.1.

3 Proof of Theorem 2.1

3.1 Overview Choose $M \in \mathbb{F}_2^{n \times n}$ by setting every entry to 1 with probability k/n. Let F denote the CNF formula encoding the system $M \cdot x = 0$. Suppose the conclusion of Theorem 2.1 does not hold. That is, there is a satisfying assignment $\mathbf{b}$ and a permutation π such that the set $S = \text{EncVbl}(\mathbf{b}, \pi, F, P^{\text{strong}})$ is too small: $|S| \leq (1 - c \ln^2 k/k)n$ for some c to be determined later. We show that with high probability, this does not happen.

DEFINITION 3.1. (STEP MATRIX) *Let $s \in \mathbb{N}$. A matrix L is an s-step matrix if the rows $\mathbf{a}_1, \mathbf{a}_2, \ldots$ of A satisfy*

$$|\text{supp}(\mathbf{a}_i) \setminus (\text{supp}(\mathbf{a}_1) \cup \ldots \cup \text{supp}(\mathbf{a}_{i-1}))| \in [s, 4s] \ . \tag{3.1}$$

In words, each new row introduces at least s and at most $4s$ new 1's.

We need some parameters. Define $\tau := 129 \ln^2 k/k$, $\ell := 4 \ln k$ and $w := \ln(k)n/k$.

LEMMA 3.1. (EXISTENCE OF A STEP MATRIX) *Suppose $|\text{EncVbl}(\mathbf{b}, \pi, F, P^{\text{strong}})| \leq (1 - 2\tau)n$. Then there exists (1) a set $U \subset [n]$ of τn variables, (2) an n/k-step matrix L of dimension $\ell \times n$ and (3) an $(\ell \times \tau n)$-matrix Z in which every row has at most w 1's such that*

$$L \cdot M_U = Z \tag{3.2}$$

where M_U is the $n \times |U|$-matrix M restricted to the variables in U.

This lemma is the main technical challenge in the proof of Theorem 2.1, and its detailed proof will be given in Section 3.3. From the lemma and next propositions, the theorem follows immediately:

PROPOSITION 3.1. *There are at most $\binom{n}{\tau n}$ ways to choose U, at most $\binom{n}{4\ell n/k}^{\ell}$ ways to choose L, and at most $\binom{\tau n}{\leq w}^{\ell}$ ways to choose Z.*

Here, we use $\binom{a}{\leq b}$ as a short-hand for $\binom{a}{0} + \binom{a}{1} + \ldots + \binom{a}{b}$. The proof of this proposition and the following propositions are rather straightforward. We give them in the appendix.

PROPOSITION 3.2. *For each fixed U, L, and Z,*

$$\Pr[L \cdot M_U = Z] \leq e^{-\ell \tau n/2} \ ,$$

where the probability is taken over the choice of M.

We combine these two propositions:

PROPOSITION 3.3. *The probability that there exists a set U and matrices L and Z as described in Lemma 3.1 is at most*

$$\binom{n}{\tau n} \cdot \binom{n}{4\ell n/k}^{\ell} \cdot \binom{\tau n}{\leq w}^{\ell} \cdot e^{-2\frac{\ln k}{k}\tau n} \ .$$

For our choice of parameters, this is $o(1)$.

Thus, it is very unlikely that there exists a satisfying assignment $\mathbf{b}$ and a permutation π such that $|\text{EncVbl}(\mathbf{b}, \pi, F, P^{\text{strong}})| \leq (1 - 2\tau)n$. This finishes the proof of Theorem 2.1.

3.2 The Connection Between PPSZ and Linear Algebra Since our formula F is derived from a system of linear equations, there is a connection between resolution proofs and Gaussian row operations. Ben-Sasson and Impagliazzo [BSI10] call this concept *Gaussian refutation* and attribute it to personal communication with Mikhail Alekhnovich. Since we could not find a concise, ready-to-use lemma in the literature, we will develop the concept ourselves here, and give complete proofs in the appendix.

Suppose F is a CNF formula and C is a clause. We say F implies C if every satisfying assignment of F also satisfies C. A literal $u \in C$ is *critical* if F implies C, but not $C \setminus \{u\}$. That means there is some satisfying assignmnet $\mathbf{b}$ that satisfies F and C, but not $C \setminus \{u\}$.

LEMMA 3.2. (GAUSSIAN RESOLUTION) *Let F be a CNF formula encoding the system of linear equations $M \cdot \mathbf{x} = \mathbf{b}$. Suppose there is a resolution derivation*

$$C_1, C_2, \ldots, C_m \ ,$$

where each C_i is either a clause of F or the resolvent of two clauses C_{i_1}, C_{i_2} for $i_1, i_2 < i$. Then there exist linear constraints of the form

$$\mathbf{c}_1 \cdot \mathbf{x} = z_1, \ \mathbf{c}_2 \cdot \mathbf{x} = z_2, \ \ldots, \ \mathbf{c}_m \cdot \mathbf{x} = z_m \ ,$$

where each constraint is either a row of the system $M \cdot \mathbf{x} = \mathbf{b}$, or $\mathbf{c}_i = \mathbf{c}_{i_1} + \mathbf{c}_{i_2}$ and $z_i = z_{i_1} + z_{i_2}$ for some $i_1, i_2 < i$. In particular, all $\mathbf{c}_i$ are linear combinations of rows of M. Furthermore, for all $1 \le i \le m$ it holds that (i) $\mathrm{supp}(\mathbf{c}_i) \subseteq \mathrm{vbl}(C_i)$, (ii) every $\mathbf{x}$ with $\mathbf{c}_i \cdot \mathbf{x} = z_i$ satisfies C_i, (iii) if a literal $u \in C_i$ is critical, then $\mathrm{vbl}(u) \in \mathrm{supp}(\mathbf{c}_i)$.

3.3 Proof of Lemma 3.1 In this section we prove Lemma 3.1. A proof of this lemma for P^{weak} (namely, replacing P^{strong} with P^{weak} in the statement), which is similar in structure but technically simpler, is given in the appendix.

Consider a run of PPSZ. Suppose there is a satisfying assignment $\mathbf{b}$ and a permutation π such that $S := \mathrm{EncVbl}(\mathbf{b}, \pi, F, P^{\mathrm{weak}})$ has size $s := |S| \le (1-2\tau)n$. From $F^{[\mathbf{x}_S \mapsto \mathbf{b}_S]}$, one can successively derive all the remaining variables by resolution of width at most w. For simplicity, assume that $S = \{1, \ldots, s\}$ and $\pi = (1, \ldots, n)$. For each variable x_j, $s+1 \le j \le n$, there exists a resolution proof

$$\begin{aligned} C_1^{(j)}, \ldots, C_{m_j}^{(j)} &= (\mathbf{x}_{<j} \neq \mathbf{b}_{<j} \vee x_j = b_j) \\ &=: C^{(j)} \ , \end{aligned} \tag{3.3}$$

such that $|\mathrm{vbl}(C_i^{(j)}) \setminus \{1, \ldots, j-1\}| \le w$. This follows from the fact that one can derive $x_j = b_j$ from $F^{[\mathbf{x}_{<j} = \mathbf{b}_{<j}]}$ using width-w-resolution. Note that the literal $x_j = b_j$ is critical in the clause $C^{(j)}$: the satisfying assignment $\mathbf{b}$ does not satisfy $(\mathbf{x}_{<j} \neq \mathbf{b}_{<j})$. We apply Lemma 3.2 to the sequence (3.3) and obtain a sequence of vectors:

$$\mathbf{c}_1^{(j)}, \mathbf{c}_2^{(j)}, \ldots, \mathbf{c}_{m_j}^{(j)} =: \mathbf{c}^{(j)} \ . \tag{3.4}$$

Since the literal $x_j = b_j$ is critical in $C^{(j)}$, we conclude that $j \in \mathrm{supp}(\mathbf{c}^{(j)})$. Since $\mathrm{supp}(\mathbf{c}^{(j)}) \subseteq \mathrm{vbl}(C^{(j)}) = \{1, \ldots, j\}$, it follows that $\max \mathrm{supp}(\mathbf{c}^{(j)}) = j$. Since every $\mathbf{c}^{(j)}$ is a linear combination of rows of M, there are vectors

$$\mathbf{r}_1^{(j)}, \mathbf{r}_2^{(j)}, \ldots, \mathbf{r}_{m_j}^{(j)} =: \mathbf{r}^{(j)} \tag{3.5}$$

such that $\mathbf{r}_i^{(j)} \cdot M = \mathbf{c}_i^{(j)}$ and each $\mathbf{r}_i^{(j)}$ in the sequence either contains exactly one 1 or is the sum of two previous vectors in the sequence. Writing the $\mathbf{r}^{(j)}$'s for $s+1 \le j \le n$ as an $(n-s) \times n$ matrix R gives the following equation:

$$\begin{pmatrix} \overset{n}{\overbrace{\qquad}} \\ \Big\downarrow \tfrac{s}{n-s} \quad R \end{pmatrix} \cdot \begin{pmatrix} \overset{n}{\overbrace{\qquad}} \\ \Big\downarrow n \quad M \end{pmatrix} = \begin{pmatrix} \overset{|S|}{\overbrace{\qquad}} \ \overset{n-|S|}{\overbrace{\qquad}} \\ \Big\downarrow \tfrac{s}{n-s} \quad * \quad \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} \quad 0 \end{pmatrix}$$

The right side has full rank $n - s(\ge 2\tau n)$, and so does R. Take the τn first rows of R to be R' and let U be the last τn variables. Let M_U be the restriction of M to these variables. This gives:

$$\begin{pmatrix} \overset{n}{\overbrace{\qquad}} \\ \Big\downarrow \tau n \quad R' \end{pmatrix} \cdot \begin{pmatrix} \overset{\tau n}{\overbrace{\qquad}} \\ \Big\downarrow n \quad M_U \end{pmatrix} = \begin{pmatrix} \overset{\tau n}{\overbrace{\qquad}} \\ \Big\downarrow \tau n \quad 0 \end{pmatrix}$$

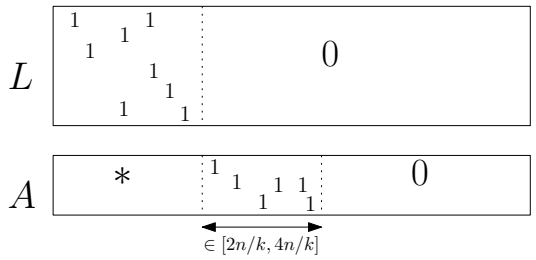
Matrix R' has full rank τn, so $|\mathrm{supp}(R')| \ge \tau n$. For general S and π, we can do the same: Let T be the *first* τn variables of $[n] \setminus S$ processed by PPSZ, and U the *last* τn variables. Form R' by restricting R to the rows corresponding to T and restrict M to the columns corresponding to U.

OBSERVATION 3.1. *Let $\mathbf{c}_i^{(j)}$ be a row vector occurring in the Gaussian resolution derivation (3.4) for some $j \in T$. Then $|\mathrm{supp}(\mathbf{c}_i^{(j)}) \cap U| \le w$.*

This is because all but w variables occurring in each clause in the resolution proof must come *before* j; however, the variables of U come after T in the permutation π.

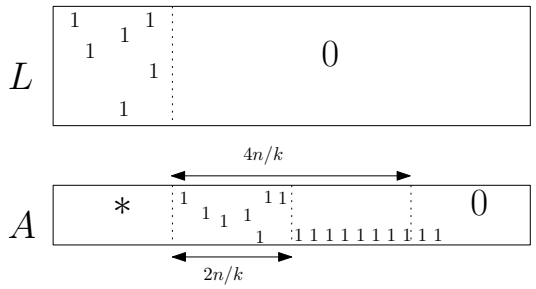
We build an $\ell \times n$-matrix L as follows by successively adding rows to it until $|\mathrm{supp}(L)| > \tau n - 2n/k$. Start with L being a $0 \times n$-matrix, i.e., not containing any rows. We build a temporary matrix A, starting with A being $0 \times n$-matrix and iteratively adding rows of R' to it. Since R' is full rank, $|\mathrm{supp}(R')| \geq \tau n$. Note that as long as $|\mathrm{supp}(L)| \leq \tau n - 2n/k$, there is some point in time where $|\mathrm{supp}(A) \setminus \mathrm{supp}(L)|$ first exceeds $2n/k$.

At this point, there are two cases. First, suppose that we have arrived at a matrix A with $2n/k \leq |\mathrm{supp}(A) \setminus \mathrm{supp}(L)| \leq 4n/k$:



Let $\mathbf{r}$ denote a random linear combination of the rows of A. Note that $\mathbb{E}[|\mathrm{supp}(\mathbf{r}) \setminus \mathrm{supp}(L)|] \in [n/k, 2n/k]$. Thus there is some linear combination $\mathbf{r}$ providing at least n/k and at most $4n/k$ new 1's. Add this row to L. Note that $\mathbf{r} \cdot M_U = 0$.

In the second case, assume $|\mathrm{supp}(A) \setminus \mathrm{supp}(L)| > 4n/k$. This means that the last row we added to A increased $\mathrm{supp}(A) \setminus \mathrm{supp}(L)$ from below $2n/k$ to above $4n/k$.



Thus, the last row $\mathbf{r}$ of R' we added to A contains at least $2n/k$ 1's that are not in $\mathrm{supp}(L)$. At this point, recall that $\mathbf{c} := \mathbf{r} \cdot M$ has been derived by Gaussian resolution

$$\mathbf{c}_1^{(j)}, \ldots, \mathbf{c}_{m_j}^{(j)} =: \mathbf{c} \tag{3.6}$$

of some variable x_j with $j \in T$. We need the following proposition:

PROPOSITION 3.4. *Let $U \subseteq [n]$ and $a \in \mathbb{N}$, $a \geq 1$. Let $\mathbf{r}_1, \ldots, \mathbf{r}_t \in \mathbb{F}_2^n$ be a sequence of vectors as in (3.5). Then either $|\mathrm{supp}(\mathbf{r}_t) \setminus U| \leq a$, or there is an $\mathbf{r}_i$ in the sequence for which $|\mathrm{supp}(\mathbf{r}_i) \setminus U| \in [a/2, a]$.*

Proof. Assume $|\mathrm{supp}(\mathbf{r}_t) \setminus U| > a$. Note that the sequence $\mathbf{r}_1, \ldots, \mathbf{r}_t$ is obtained from a resolution proof and thus it follows from Lemma 3.2 that for each $1 \leq i \leq t$, $|\mathrm{supp}(\mathbf{r}_i)| = 1$ or $\mathbf{r}_i = \mathbf{r}_{i_1} + \mathbf{r}_{i_2}$ for some $i_1, i_2 < i$ which implies $\mathrm{supp}(\mathbf{r}_i) \subseteq \mathrm{supp}(\mathbf{r}_{i_1}) \cup \mathrm{supp}(\mathbf{r}_{i_2})$. But this gives $\mathrm{supp}(\mathbf{r}_i) \setminus U \subseteq (\mathrm{supp}(\mathbf{r}_{i_1}) \setminus U) \cup (\mathrm{supp}(\mathbf{r}_{i_2}) \setminus U)$ and consequently $|\mathrm{supp}(\mathbf{r}_i) \setminus U| \leq |\mathrm{supp}(\mathbf{r}_{i_1}) \setminus U| + |\mathrm{supp}(\mathbf{r}_{i_2}) \setminus U|$. Combining this sub-additivity with the fact that $|\mathrm{supp}(\mathbf{r}_1) \setminus U| = 1$ and $|\mathrm{supp}(\mathbf{r}_t) \setminus U| > a$, it follows that there must be some $\mathbf{r}_i$ satisfying $|\mathrm{supp}(\mathbf{r}_i) \setminus U| \in [a/2, a]$. □

By Proposition 3.4, there is some $\mathbf{r}'$ in the sequence such $|\mathrm{supp}(\mathbf{r}') \setminus \mathrm{supp}(L)| \in [n/k, 2n/k]$. We add $\mathbf{r}'$ to L. Note that $\mathbf{r}' \cdot M =: \mathbf{c}'$ is a row vector in the sequence (3.6). By Observation 3.1, this means that $|\mathrm{supp}(\mathbf{c}) \cap U| \leq w$. Consequently, $\mathbf{r}' \cdot M_U$ has at most w 1's.

Let us summarize: Every row in L introduces at least n/k and at most $4n/k$ 1's. This means, we can repeat this process at least $\frac{\tau n}{4n/k} - 1 = \tau k/4 - 1 \geq 4 \ln k$ times. This finishes the proof of Lemma 3.1.

Acknowledgements

We are grateful to Periklis Papakonstantinou, Pavel Pudlák, and Rahul Santhanam for inspiring discussions.

References

[ACL+12] E. Allender, S. Chen, T. Lou, P. Papakonstantinou, and B. Tang. Width-parameterized SAT: time-space tradeoffs. *ECCC TR12-027*, 2012.

[AHI05] Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. *J. Autom. Reasoning*, 35(1-3):51–72, 2005.

[BSI10] Eli Ben-Sasson and Russell Impagliazzo. Random CNF's are hard for the polynomial calculus. *Computational Complexity*, 19(4):501–519, 2010.

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Comm. ACM*, 5:394–397, 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. Assoc. Comput. Mach.*, 7:201–215, 1960.

[Her11] Timon Hertli. 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. In Rafail Ostrovsky, editor, *FOCS*, pages 277–284. IEEE, 2011.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *J. Comput. System Sci.*, 63(4):512–530, 2001. Special issue on FOCS 98 (Palo Alto, CA).

[PI00] P. Pudlák and R. Impagliazzo. A lower bound for dll algorithms for k-sat (preliminary version). In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 128–136. Society for Industrial and Applied Mathematics, 2000.

[PPSZ98] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. In *FOCS*, pages 628–637. IEEE Computer Society, 1998.

[PPSZ05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005.

[PPZ99] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theoret. Comput. Sci.*, pages Article 11, 19 pp. (electronic), 1999.

[Sch99] Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 410, Washington, DC, USA, 1999. IEEE Computer Society.

In this appendix we state the proofs that are missing in the main part of the paper. For convenience, we do not only give the proofs, but also repeat the statements of the propositions and lemmas. Therefore, we also repeat the original numbering.

A Remaining Proofs

PROPOSITION A.1. (PROPOSITION 2.1) *With high probability over the choice of M, the system $M \cdot \mathbf{x} = 0$ has at most $2^{\epsilon_k n}$ solutions, where $\epsilon_k \in O(\ln^2(k)/k)$.*

Proof. We compute the expected number of solutions to $M \cdot \mathbf{x} = 0$. This is

$$\sum_{\mathbf{x} \in \mathbb{F}_2^n} \Pr[M \cdot \mathbf{x} = 0] \ . \tag{A.1}$$

Each summand depends only on the number of 1's in the vector $\mathbf{x}$. If $\mathbf{x}$ has exactly w 1's, then

$$\Pr[M \cdot \mathbf{x} = 0] = (\Pr[\mathcal{B}in(w, k/n) \equiv 0 \bmod 2])^n =: p_w^n \ .$$

Thus, (A.1) is

$$\sum_{w=0}^{n} \binom{n}{w} p_w^n \ .$$

We divide this sum into two parts: for w above a certain threshold, $\mathcal{B}in(w, k/n)$ is even with probability close to $1/2$. For w below this threshold, that p_w might be close to 1. But there are few such $\mathbf{x}$ in the first place. We can compute p_w as follows.

FACT A.1. $p_w = \frac{1+(1-2k/n)^w}{2}$.

Proof. Set $a = k/n$. We write p_w explicitely:

$$p_w = \sum_{0 \le i \le w,\ \text{even}} \binom{w}{i} a^i (1-a)^{w-i} \ .$$

Define and observe:

$$\begin{aligned} A &:= \sum_{i=0}^{w} \binom{w}{i} a^i (1-a)^{w-i} = 1 \ , \\ B &:= \sum_{i=0}^{w} \binom{w}{i} (-a)^i (1-a)^{w-i} = (1-2a)^w \ . \end{aligned}$$

In $A + B$, all terms for odd i cancel out:

$$A + B = \sum_{i\ \text{even}} \binom{w}{i} 2a^i (1-a)^{w-i} = 2p_w \ .$$

Therefore, $p_w = \frac{A+B}{2} = \frac{1+(1-2a)^w}{2}$. □

Fixing $w_0 := \ln(k)n/k$, above the threshold we now have

$$\begin{aligned}\sum_{w=w_0}^{n}\binom{n}{w}p_w^n &= \sum_{w=w_0}^{n}\binom{n}{w}\left(\frac{1+(1-2k/n)^w}{2}\right)^n \\ &\le \sum_{w=w_0}^{n}\binom{n}{w}\left(\frac{1+(1-2k/n)^{w_0}}{2}\right)^n \\ &\le (1+(1-2k/n)^{w_0})^n \\ &< (1+e^{-2kw_0/n})^n \\ &= (1+e^{-2\ln(k)})^n \\ &= (1+1/k^2)^n \le e^{n/k^2}\ .\end{aligned}$$

Below the threshold, there are at most

$$\sum_{w=0}^{w_0}\binom{n}{w} \le \left(\frac{en}{w_0}\right)^{w_0} \le \left(\frac{ek}{\ln(k)}\right)^{\ln(k)n/k} \le e^{2\ln^2(k)n/k}\ .$$

Thus both above and below the threshold the expected number of $\mathbf{x}$ such that $M\cdot x = 0$ is at most $2^{O(\ln^2(k)n/k)}$. □

Proposition A.2. (Proposition 2.2) *Let F be a CNF formula over n variables, $T \subseteq [n]$ a set of variables, and $\mathbf{b}^*$ be a satisfying assignment. Set $F' := F^{[x_T \mapsto \mathbf{b}^*_T]}$. Then* $\text{codelength}(F', P^{\text{strong}}) \ge \text{codelength}(F, P^{\text{strong}}) - |T|$.

Proof. Let $\mathbf{b}'$ be a satisfying assignment of F' and π' be a permutation of $\text{vbl}(F')$ achieving $|S| = \text{codelength}(F', P^{\text{strong}})$, where $S = \texttt{encode}(\mathbf{b}', \pi', F', P^{\text{strong}})$. Let $\mathbf{b}$ be the assignment to the variables $\text{vbl}(F)$ that agrees with $\mathbf{b}'$ on $\text{vbl}(F')$ and with $\mathbf{b}^*$ on the rest. Note that $\mathbf{b}$ satisfies F. Define a permutation π of $\text{vbl}(F)$ as follows: first take the variables of T, in an arbitrary order, then the variables of $\text{vbl}(F')$, according to their order in π'. Should there be leftover variables, insert them at any point. What is $|\texttt{encode}(\mathbf{b}, \pi, F, P^{\text{strong}})|$? Note that `encode` first processes the variables in T. Thus, it will arrive at the formula F' after that. From then on, it will encode exactly $|\texttt{encode}(\mathbf{b}', \pi', F', P^{\text{strong}})|$ variables. Therefore, $|\texttt{encode}(\mathbf{b}, \pi, F, P^{\text{strong}})| \le |T| + |\texttt{encode}(\mathbf{b}', \pi', F', P^{\text{strong}})|$. □

Proposition A.3. (Proposition 2.3) *Choose $M \in \mathbb{F}_2^{n\times n}$ by setting every entry to 1 with probability k/n. Call a row* excessive *if it has more than $2ek$ 1's. The total number of 1's in the excessive rows is $O(nk/2^k)$.*

Proof. Fix $w \ge 2ek$. What is the probability that a row has at least w 1's? This is at most

$$\binom{n}{w}\left(\frac{k}{n}\right)^w \le \left(\frac{en}{w}\right)^w\left(\frac{k}{n}\right)^w = \left(\frac{ek}{w}\right)^w \le 2^{-w}\ .$$

Note that M has n rows. The expected total number of 1's in excessive rows is at most

$$\begin{aligned}n\sum_{w=2ek}^{\infty} w\cdot 2^{-w} &= n\sum_{i=0}^{\infty}(i+2ek)2^{-i-2ek} \\ &= n2^{-2ek}\left(\sum_{i=0}^{\infty} i2^{-i} + \sum_{i=0}^{\infty}2ek2^{-i}\right) \\ &= n2^{-2ek}(2+4ek) \in O(nk/2^k).\end{aligned}$$

This completes the proof. □

Proposition A.4. (Proposition 3.1) *There are at most $\binom{n}{\tau n}$ ways to choose U, at most $\binom{n}{4\ell n/k}^{\ell}$ ways to choose L, and at most $\binom{\tau n}{\le w}^{\ell}$ ways to choose Z.*

Proof. U is a set of τn variables, so there are $\binom{n}{\tau n}$ ways to choose U. As for L, it is an n/k-step matrix of dimensions $\ell \times n$. This means that every row introduces at most $4n/k$ new 1's. Thus, every row has at most $4\ell n/k$ 1's. There are $\binom{n}{4\ell n/k}$ ways to choose such a row, and ℓ such rows to choose. Finally, Z is an $\ell\times\tau n$-matrix in which each row contains at most w 1's. Thus, there are $\binom{\tau n}{\le w}^{\ell}$ such matrices. □

Proposition A.5. (Proposition 3.2) *For each fixed U, L, and Z,*

$$\Pr[L\cdot M_U = Z] \le e^{-\ell\tau n/2}\ ,$$

where the probability is taken over the choice of M.

Proof. M_U has τn columns, each of which is chosen independently. Thus, it suffices to show that

$$\Pr[L\cdot\mathbf{m} = \mathbf{z}] \le e^{-\ell/2} \tag{A.2}$$

for every column $\mathbf{m}$ of M and the corresponding column $\mathbf{z}$ of Z. Let $\mathbf{r}_1, \ldots, \mathbf{r}_\ell$ be the rows of L. We prove that for $1 \le i \le \ell$ it holds that

$$\Pr[\mathbf{r}_i\cdot\mathbf{m} = z_i \mid \forall\ i' < i,\ \mathbf{r}_{i'}\cdot\mathbf{m} = z_{i'}] \le e^{-1/2}\ . \tag{A.3}$$

From here, (A.2) follows by the chain rule of conditional probabilities.

To show (A.3), write $\mathbf{r}_i = \mathbf{r}_{\text{old}} + \mathbf{r}_{\text{new}}$, where $\mathbf{r}_{\text{new}}$ contains exactly those 1's that are not in $\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}$. Since L is an n/k-step matrix, $|\text{supp}(\mathbf{r}_{\text{new}})| \ge n/k$. Denote by $\mathcal{E}$ the condition that $\mathbf{r}_{i'}\cdot\mathbf{m} = z_i$ for all $i' < i$. Then

$$\begin{aligned}\Pr[\mathbf{r}_i\cdot\mathbf{m} = z_i\ |\mathcal{E}] &= \Pr[\mathbf{r}_{\text{new}}\cdot\mathbf{m} + \mathbf{r}_{\text{old}}\cdot\mathbf{m} = z_i \mid \mathcal{E}] \\ &= \Pr[\mathbf{r}_{\text{new}}\cdot\mathbf{m} = z_i + \mathbf{r}_{\text{old}}\cdot\mathbf{m} \mid \mathcal{E}].\end{aligned} \tag{A.4}$$

Since $\mathbf{r}_{\text{new}}$ only contains 1's that are not present in any previous $\mathbf{r}_{i'}$, the random variable $\mathbf{r}_{\text{new}} \cdot \mathbf{m}$ is independent of $\mathbf{r}_{\text{old}} \cdot \mathbf{m}$ and $\mathcal{E}$. Define $y := z_i + \mathbf{r}_{\text{old}} \cdot \mathbf{m}$ and estimate

$$\Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} = y] \ .$$

We write $m = |\text{supp}(\mathbf{r}_{\text{new}})|$ and $p = k/n$. Note that

$$\Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} = y] = \begin{cases} \Pr[\mathcal{B}in(m,p) \text{ is even}] & \text{when } y = 0 \\ \Pr[\mathcal{B}in(m,p) \text{ is odd}] & \text{when } y = 1 \end{cases}$$

The probability that $\mathcal{B}in(m,p)$ is even is:

$$\begin{aligned} \Pr[\mathcal{B}in(n/k, k/n) \text{ is even}] &= \frac{1 + (1-2k/n)^{n/k}}{2} \\ &\leq \frac{1 + e^{-2}}{2} \leq e^{-1/2} \ . \end{aligned}$$

The probability that $\mathcal{B}in(m,p)$ is odd is at most $1/2$. Thus, we see that $\Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} = y] \leq e^{-1/2}$, no matter what y is. This concludes the proof. □

Proposition A.6. (Proposition 3.3) *The probability that there exists a set U and matrices L and Z as described in Lemma 3.1 is at most*

$$\text{(A.5)} \quad \binom{n}{\tau n} \cdot \binom{n}{4\ell n/k}^{\ell} \cdot \binom{\tau n}{\leq w}^{\ell} \cdot k^{-2\tau n} \ .$$

For our choice of parameters this is $o(1)$.

Proof. The expression in (A.5) follows from the two previous propositions via a union bound. Let us evaluate it for our choice of parameters. Recall that $\tau = 129\ln^2(k)/k$, $w = \ln(k)n/k$ and $\ell = 4\ln(k)$. Thus, $4\ell n/k = 16w$.

$$\begin{aligned} (A.5) &\leq \binom{n}{\tau n} \cdot \binom{n}{16w}^{\ell} \cdot \binom{\tau n}{\leq w}^{\ell} \cdot k^{-2\tau n} \\ &\leq \binom{n}{\tau n} \cdot \binom{n}{16w}^{2\ell} \cdot w^{l} \cdot k^{-2\tau n} \\ &\leq \left(\frac{ek}{129\ln^2(k)}\right)^{\tau n} \cdot \left(\frac{k}{16\ln(k)}\right)^{32w\ell} \cdot w^{l} \cdot k^{-2\tau n} \\ &\leq k^{\tau n} k^{32w\ell} \cdot k^{4\ln n} \cdot k^{-2\tau n} \\ &\leq k^{32w\ell - \tau n + 4\ln n} \end{aligned}$$

Focus on the exponent: $32w\ell - \tau n = 128\ln^2(k)n/k - 129\ln^2(k)n/k = -\ln^2(k)n/k$. Thus, $(A.5) \leq k^{-\ln^2(k)n/k + \ln n} = o(1)$. In fact, it is exponentially small. □

B Gaussian Resolution

Lemma B.1. (Gaussian Resolution, Lemma 3.2) *Let F be a CNF formula encoding the system of linear equations $M \cdot \mathbf{x} = \mathbf{b}$. Suppose there is a resolution derivation*

$$C_1, C_2, \ldots, C_m \ ,$$

where each C_i is either a clause of F or the resolvent of two clauses C_{i_1}, C_{i_2} for $i_1, i_2 < i$. Then there exist linear constraints of the form

$$\mathbf{c}_1 \cdot \mathbf{x} = z_1, \ \mathbf{c}_2 \cdot \mathbf{x} = z_2, \ \ldots, \ \mathbf{c}_m \cdot \mathbf{x} = z_m \ ,$$

where each constraint is either a row of the system $M \cdot \mathbf{x} = \mathbf{b}$, or $\mathbf{c}_i = \mathbf{c}_{i_1} + \mathbf{c}_{i_2}$ and $z_i = z_{i_1} + z_{i+2}$ for some $i_1, i_2 < i$. In particular, all $\mathbf{c}_i$ are linear combinations of rows of M. Furthermore, for all $1 \leq i \leq m$ it holds that (i) $\text{supp}(\mathbf{c}_i) \subseteq \text{vbl}(C_i)$, (ii) every $\mathbf{x}$ with $\mathbf{c}_i \cdot \mathbf{x} = z_i$ satisfies C_i, (iii) if a literal $u \in C_i$ is critical, then $\text{vbl}(u) \in \text{supp}(\mathbf{c}_i)$.

Proof. We apply induction over the length of the sequence. It suffices to prove the lemma for $i = m$. For all smaller $i < m$, just apply the lemma to the subsequence ending at i. So consider C_m. If it is a clause of F, then it comes from some linear constraint $\mathbf{c} \cdot \mathbf{x} = b$. Clearly, $\text{supp}(\mathbf{c}) = \text{vbl}(C_m)$, and every x with $\mathbf{c} \cdot \mathbf{x} = b$ satisfies C_m.

So suppose C_m is the resolvent of two previous clauses C_k, C_ℓ. Without loss of generality, $C_k = x_j \vee C'_k$ and $C_\ell = \bar{x}_j \vee C'_\ell$. By induction, there are corresponding linear constraints $\mathbf{c}_k \cdot \mathbf{x} = z_k$ and $\mathbf{c}_\ell \cdot \mathbf{x} = z_\ell$. We consider two cases.

Case 1. $j \in \text{supp}(\mathbf{c}_k) \cap \text{supp}(\mathbf{c}_\ell)$: We define $\mathbf{c}_m := \mathbf{c}_j + \mathbf{c}_k$ and $z_m := z_j + z_k$. Note that $j \notin \text{supp}(\mathbf{c}_m)$, and therefore $\text{supp}(\mathbf{c}_m) \subseteq (\text{supp}(\mathbf{c}_k) \cup \text{supp}(\mathbf{c}_\ell)) \setminus \{j\} \subseteq \text{vbl}(C'_k) \cup \text{vbl}(C'_\ell) = \text{vbl}(C_m)$, by induction. This shows (i). For (ii), suppose $\mathbf{x}$ does not satisfy C_m. We show that $\mathbf{c}_m \cdot \mathbf{x} = z_m + 1$. Since $\mathbf{x}$ does not satisfy C_m, it satisfies neither C'_k nor C'_ℓ. Without loss of generality, $x_j = 0$. So $\mathbf{x}$ does not satisfy C_k and $\mathbf{x} + \mathbf{e}_j$ does not satisfy C_ℓ. By induction, this means that

$$\begin{aligned} \mathbf{c}_k \cdot \mathbf{x} &= z_k + 1 \\ \mathbf{c}_\ell \cdot (\mathbf{x} + \mathbf{e}_j) &= z_\ell + 1 \ . \end{aligned}$$

Adding these two equations gives

$$\begin{aligned} \mathbf{c}_k \cdot \mathbf{x} + \mathbf{c}_\ell \cdot \mathbf{x} + \mathbf{c}_\ell \cdot \mathbf{e}_j &= z_k + z_\ell \ \Leftrightarrow \\ (\mathbf{c}_k + \mathbf{c}_\ell) \cdot \mathbf{x} + \mathbf{c}_\ell \cdot \mathbf{e}_j &= z_m \ \Leftrightarrow \\ \mathbf{c}_m \cdot \mathbf{x} + 1 &= z_m \ . \end{aligned}$$

The last equation follows since $\mathbf{c}_\ell$ has a 1 at position j: by assumption $j \in \text{supp}(\mathbf{c}_\ell)$. This shows (ii).

Case 2. $j \notin \text{supp}(\mathbf{c}_k)$ or $j \notin \text{supp}(\mathbf{c}_\ell)$: Without loss of generality, $j \notin \mathbf{c}_k$. In this case, simply set

$\mathbf{c}_m := \mathbf{c}_k$ and $z_m = z_k$. Obviously, (i) holds. For (ii) suppose $\mathbf{x}$ does not satisfy C_m. If $x_j = 0$ then $\mathbf{x}$ does not satisfy C_k either, so $\mathbf{c}_k \cdot \mathbf{x} = z_k + 1$ by induction, and we are done. If $x_j = 1$, define $\mathbf{y} = \mathbf{x} + \mathbf{e}_j$. Note that $\mathbf{y}$ does not satisfy C_k, and therefore $\mathbf{c}_k \cdot \mathbf{y} = z_k + 1$. But $\mathbf{c}_k$ has a 0 at position j, therefore $\mathbf{c}_k \cdot \mathbf{y} = \mathbf{c}_k \cdot \mathbf{x}$ and $\mathbf{c}_k \cdot \mathbf{x} = z_k + 1$. Again, we are done.

It remains to prove (iii). Let $u \in C_m$ be a critical literal. We have to show that $\mathrm{vbl}(u) \in \mathrm{supp}(\mathbf{c}_m)$. Since u is critical, F does not imply $C'_m := C_m \setminus \{u\}$, thus there is an assignment $\mathbf{x}$ satisfying F but not C'_m. Since F implies C_m, $\mathbf{x}$ satisfies C_m. Thus, u is the unique satisfied literal in C_m. Let x_j be the underlying variable of u. Define $\mathbf{y} := \mathbf{x} + \mathbf{e}_j$. Observe that $\mathbf{y}$ does not satisfy u, and thus does not satisfy C either. By (ii), this means that $\mathbf{c}_m \cdot \mathbf{y} = z_m + 1$. However, $\mathbf{x}$ is a satisfying assignment, thus $M \cdot \mathbf{x} = \mathbf{b}$ and $\mathbf{c}_m \cdot \mathbf{x} = z_m$, since every constraint is a linear combination of constraints of the original system. We summarize:

$$\begin{aligned} \mathbf{c}_m \cdot \mathbf{y} &= z_m + 1 \\ \mathbf{c}_m \cdot \mathbf{x} &= z_m \ . \end{aligned}$$

Adding these two equations gives

$$1 = \mathbf{c}_m \cdot (\mathbf{y} + \mathbf{x}) = \mathbf{c}_m \cdot \mathbf{e}_j \ .$$

This means that $\mathbf{c}_m$ has a 1 at position j, i.e., $j \in \mathrm{supp}(\mathbf{c}_m)$. This shows (iii) and concludes the proof. □

C Proof of Lemma 3.1 for P^{weak}

Recall that P^{weak} is the heuristic checking implication by small subformulas. Here, small means at most $w := n/k$. We will prove Lemma 3.1 for P^{weak} (namely, replacing P^{strong} with P^{weak} in the statement). We do this because the proof is much simpler and clearer than for P^{strong}, especially for a reader who is unfamiliar with resolution.

Let $\mathbf{b}$ be a satisfying assignment of F and π a permutation such that $S := \mathrm{EncVbl}(\mathbf{b}, \pi, F, P^{\mathrm{weak}})$ has size $s := |S| \leq (1 - 2\tau)n$. By the assumption of Lemma 3.1, such $\mathbf{b}$, π and S exist. For the moment suppose $S = \{1, \ldots, s\}$ and $\pi = (1, 2, \ldots, n)$. From $F^{[\mathbf{x}_S \mapsto \mathbf{b}_S]}$, ppsz can successively infer $x_{s+1}, x_{s+2}, \ldots, x_n$. This means, there are subformulas $G_{s+1}, \ldots, G_n$ of F from which P^{strong} infers the variables $x_{s+1}, \ldots, x_n$: $\forall j$, $|G_j| \leq w$, and every satisfying assignment $\mathbf{x}$ of G_j with $\mathbf{x}_{<j} = \mathbf{b}_{<j}$ also satisfies $x_j = b_j$. Every clause of F comes from a row of M. Thus, the subformulas $G_{s+1}, \ldots, G_n$ correspond to sets $R_{s+1}, \ldots, R_n$ of rows of M where $|R_j| \leq w = n/k$, and every $\mathbf{x}$ with $R_j \cdot \mathbf{x} = 0$ and $\mathbf{x}_{<j} = \mathbf{b}_{<j}$ satisfies $x_j = b_j$.

Lemma C.1. *Let M be a matrix. The unit vector $\mathbf{e}_j$ is in the row span of M if and only all solutions $\mathbf{x}$ of $M \cdot \mathbf{x} = 0$ satisfy $x_j = 0$.*

Proof. The "only if" direction is easy: If $\mathbf{e}_j \in \mathrm{rowspan}(M)$, then there exists a row vector $\mathbf{r}$ such that $\mathbf{r} \cdot M = \mathbf{e}_j$. Now $x_j = \mathbf{e}_j \cdot x = \mathbf{r} \cdot M \cdot x = 0$. For the other direction, note that the set of solutions x is the kernel of M, which in turn is $\mathrm{rowspan}(M)^\perp$. Every $x \in \mathrm{rowspan}(M)^\perp$ satisfying $x_j = 0$ means that $\mathbf{e}_j \perp \mathrm{rowspan}(M)^\perp$, which in turn means $\mathbf{e}_j \in \left(\mathrm{rowspan}(M)^\perp\right)^\perp = \mathrm{rowspan}(M)$. □

Corollary C.1. *Let M be a matrix with n columns, $S \subset [n]$, $j \in [n] \setminus S$, and $\mathbf{b}$ be a solution to $M \cdot \mathbf{x} = 0$. If all $\mathbf{x}$ with $M \cdot \mathbf{x} = 0$ and $\mathbf{x}_S = \mathbf{b}_S$ satisfy $x_j = b_j$, then there is a row vector $\mathbf{m} \in \mathrm{rowspan}(M)$ such that $\mathrm{supp}(\mathbf{m}) \setminus S = \{j\}$.*

Proof. Take the system $M \cdot \mathbf{x} = 0$ and replace x_i by the constant b_i for each $i \in S$. This gives a new system

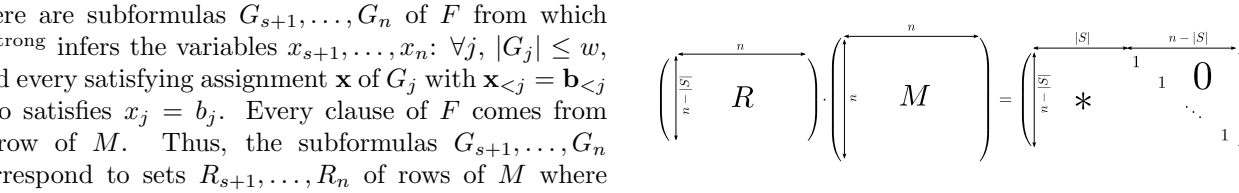
$$M' \cdot \mathbf{x}' = \mathbf{c} \tag{C.6}$$

Every solution to (C.6) satisfies $x'_j = b_j$. The restriction $\mathbf{b}' := \mathbf{b}_{[n] \setminus S}$ is a solution to (C.6). If $\mathbf{y}$ is a solution to $M' \cdot \mathbf{x}' = 0$, then $\mathbf{y} + \mathbf{b}'$ is a solution to (C.6), and therefore $y_j + b_j = b_j$. So every solution $\mathbf{y}$ of $M' \cdot \mathbf{x}' = \mathbf{c}$ satisfies $y_j = 0$. By Lemma C.1, $\mathbf{e}_j \in \mathrm{rowspan}(M')$. So there exists a row vector $\mathbf{r}$ such that $\mathbf{r} \cdot M' = \mathbf{e}_j$. How does $\mathbf{m} := \mathbf{r} \cdot M$ compare to $\mathbf{r} \cdot M' = \mathbf{e}_j$? It has additional coordinates: $i \in S$. However, the coordinates $i \in [n] \setminus S \setminus \{j\}$ are still 0. Thus, $\mathrm{supp}(m) \setminus S = \{j\}$. □

The corollary implies that there are sets $R_{s+1}, \ldots, R_n$ of rows of M such that $|R_j| \leq w$ and

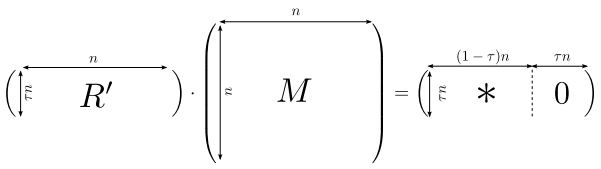
$$\sum_{i \in R_j} \mathbf{m}_i = (*, \ldots, *, 1, 0, \ldots, 0) \ , \forall j : s + 1 \leq j \leq n$$

where the 1 is at the j^{th} position. Here, $\mathbf{m}_i$ is the i^{th} row of M. If we let $\mathbf{r}_i$ denote the characteristic vector of R_j, then $\sum_{i \in R_j} \mathbf{m}_i = \mathbf{r}_j \cdot M$. Let R denote the $(n - s) \times n$-matrix with rows $\mathbf{r}_j$, $s + 1 \leq j \leq n$. We see that the matrix M satisfies the following equation:

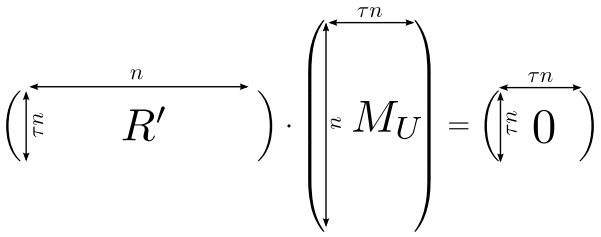


Every row of R contains at most w 1's. Note that the upper triangle at the end of $R \cdot M$ implies that $R \cdot M$

is full rank, and R has full rank $n - s \geq 2\tau n, too$. Let us look at the "upper half" of that equation, that is, let R' consist of the τn first rows of R. R' has rank τn and satisfies the following equation:



Finally, let $U \subseteq [n]$ be the last τn variables, and M_U the matrix M restricted to the variables in U. We get:



Of course, we can come up with these matrices for any π and S. Simply let T be the *first* τn variables of $[n] \setminus S$ processed by PPSZ, and U the *last* τn variables. Form R' by restricting R to the rows corresponding to T and restrict M to the columns corresponding to U.

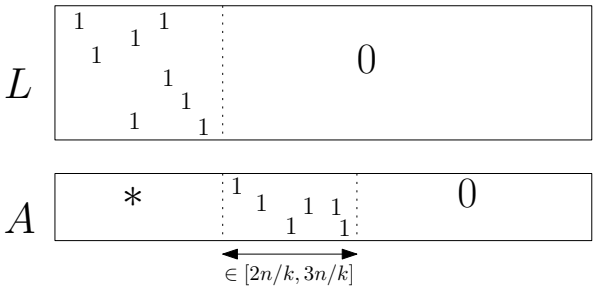
A naïve union bound. We would like to proceed as follows: first, estimate the probability for given R' that $R' \cdot M_U = 0$. Second, perform a union bound over (i) all choices of R', (ii) all choices of S, (iii) all choices of T. A direct union bound is bound to fail, though: note that with probability $(1 - k/n)^{\tau n^2}$, M_U is all-0 and thus satisfies the equation. This probability is small, but only singly exponentially small. How many choices of R' are there? There are τn rows, and in each row we choose w entries to be 1. Thus, there are $\binom{n}{w}^{\tau n}$ choices, which is of order $2^{\Omega(n \log n)}$. There is no way of surviving this union bound.

A cleverer union bound. Instead of performing a union bound over all R', we argue as follows. If $R' \cdot M_U = 0$, then there is a very structured matrix L, the step matrix, such that $L \cdot M_U = 0$, too. Indeed L is so structured that there are only exponentially many choices of L. This is the reason why we want to prove the existence of such a step matrix.

We are now ready to explain how to construct the step matrix L.

Proof. [Proof of Lemma 3.1 for P^{weak}] We successively build L by adding rows to it. Each such row will be a linear combination of rows of R'. This makes sure that $L \cdot M_U = 0$. Suppose we have built up parts of L and want to find a new row we can add to it. For this we build a temporary matrix A.

Grow a matrix A with n columns by adding rows of R' to it. Note that R' has full rank, therefore $|\text{supp}(R')| \geq \tau n$. As long as $|\text{supp}(L)| \leq \tau n - 2n/k$, since $|\text{supp}(R')| \geq \tau n$ there is a point in time where $|\text{supp}(A) \setminus \text{supp}(L)| \geq 2n/k$ for the first time. At this point, $|\text{supp}(A) \setminus \text{supp}(L)| \leq 3n/k$, since the last row adds at most $w = n/k$ 1's.



Take a random linear combination of the rows in A. This has at most $3n/k$ 1's outside $\text{supp}(L)$ and, on expectation, at least n/k 1s outside $\text{supp}(L)$. Thus, there is some linear combination $\mathbf{r}$ with $n/k \leq |\text{supp}(\mathbf{r}) \setminus \text{supp}(L)| \leq 3n/k$. Add this row to L.

Since $|\text{supp}(L)|$ is 0 at the beginning and grows by at most $3n/k$ in each step, we can grow L for at least

$$\frac{|\text{supp}(R')|}{3n/k} \geq \frac{\tau k}{3} \geq \frac{129 \ln(k)}{3} \geq 4 \ln(k) \; .$$

steps. This final matrix L satisfies $L \cdot M_U = 0$, since every row of L is a linear combination of rows of R'. This is even better then required: Lemma 3.1 just states that $L \cdot M_U$ has at most $\frac{\ln k}{k} n$ 1's per row. The dimensions of L are $4 \ln(k) \times n$, and it is an n/k-step matrix. This finishes the proof of the Lemma 3.1 for P^{weak}. □