Improving PPSZ for 3-SAT using Critical Variables

Timon Hertli, Robin Moser, Dominik Scheder

Improving PPSZ for 3-SAT using Critical Variables

Timon Hertli[1,2], Robin A. Moser[1,3], and Dominik Scheder[1,4]

1 **Institute for Theoretical Computer Science**
Department of Computer Science
ETH Zürich, 8092 Zürich, Switzerland
2 **timon.hertli@inf.ethz.ch**
3 **robin.moser@inf.ethz.ch**
4 **dominik.scheder@inf.ethz.ch**

Abstract

A critical variable of a satisfiable CNF formula is a variable that has the same value in all satisfying assignments. Using a simple case distinction on the fraction of critical variables of a CNF formula, we improve the running time for 3-SAT from $\mathcal{O}(1.32216^n)$ by Rolf [10] to $\mathcal{O}(1.32153^n)$. Using a different approach, Iwama et al. [5] very recently achieved a running time of $\mathcal{O}(1.32113^n)$. Our method nicely combines with theirs, yielding the currently fastest known algorithm with running time $\mathcal{O}(1.32065^n)$. We also improve the bound for 4-SAT from $\mathcal{O}(1.47390^n)$ [6] to $\mathcal{O}(1.46928^n)$, where $\mathcal{O}(1.46981^n)$ can be obtained using the methods of [6] and [10].

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems; G.2.1 Combinatorics.

Keywords and phrases SAT, satisfiability, randomized, exponential time, algorithm, 3-SAT, 4-SAT

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.237

1 Introduction

The ideas behind the most successful algorithms for k-SAT are surprisingly simple. In 1999, Paturi, Pudlák, and Zane [9] proposed the following algorithm. Given a k-CNF formula F, we choose a variable x uniformly at random from the n variables in F, choose a truth value $b \in \{0,1\}$, and set x to b, thereby replacing F by $F^{[x \mapsto b]}$, and continue with $F^{[x \mapsto b]}$. The value b is chosen as follows: If the formula contains the unit clause (x), we choose $b = 1$. If it contains $(\bar{x})$, we choose $b = 0$. In these two cases, we say x was *forced*. If it contains neither, we choose b randomly and say x was *guessed*. Finally, if the formula contains both (x) and $(\bar{x})$, we can give up, since the formula is unsatisfiable. This algorithm is usually called PPZ after its three inventors.

Intuitively, if F is "strongly constrained", then the algorithm encounters many unit clauses, hence it needs to guess significantly fewer than n variables. On the other hand, if F is only "weakly constrained", it has multiple satisfying assignments, making it easier to find one. Paturi, Pudlák and Zane [9] make this intuition precise and show that PPZ finds a satisfying assignment for a k-CNF formula with probability at least $2^{-(1-1/k)n}$, provided there exists one.

A couple of years later, Paturi, Pudlák, Saks, and Zane [8] came up with a simple but powerful idea. In a preprocessing step, they apply a restricted version of resolution. This increases the number of unit clauses the algorithm encounters and therefore increases its success probability. This gives an algorithm called PPSZ. If F has a unique satisfying assignment, its success probability is quite good (for 3-SAT, it is $\Omega(1.308^{-n})$), and the

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 237–248

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


analysis is highly elegant. The case of multiple satisfying assignments appears to be much more difficult and has been the subject of several papers so far. Iwama and Tamaki [6] made a major step forward when they observed that while the success probability of PPSZ deteriorates as the number of satisfying assignments increases, that of Schöning's random walk algorithm [11] improves. They quantified this tradeoff and obtained an algorithm with a success probability of $\Omega(1.32373^{-n})$[1]. We denote this combined algorithm, consisting of one run of PPSZ and one run of Schöning's random walk algorithm, by Comb.

The PPSZ paper. There are two versions of [8], which we call the old version and the new version. For unique k-SAT, both are the same, but for general k-SAT, the old version of [8] gives a more complicated analysis. The old version gives a better bound for 3-SAT and the new version gives a better bound for 4-SAT.

Only the new version is published, but the old version is still available at the Citeseer cache[2]. However, we have found some minor errors in that version. There is also a conference version [7] stating the results of the old version of [8], but without most proofs. Rolf [10] improved the analysis of the old version to get a bound of $\Omega(1.32216^n)$. However [10] does not consider 4-SAT. We use the ideas of [10] for our improvement of 4-SAT. In Timon Hertli's master thesis [2], the old version of [8] with the result of [10] is presented in a self-contained way. We will reference that thesis for detailed proofs.

1.1 Our Contribution

Let F be a satisfiable CNF formula over n variables and x be a variable therein. We call x *critical* if all satisfying assignments of F agree on x. Equivalently, x is critical if exactly one of the formulas $F^{[x\mapsto 1]}$ and $F^{[x\mapsto 0]}$ is satisfiable. We denote by $c(F)$ the fraction of critical variables, i.e., the number of critical variables divided by n; if $n = 0$, we define $c(F) := 1$.

Our contribution consists of two statements: Theorem 1 shows that for our purposes we only need to consider formulas with many critical variables. Point 3 of Lemma 9 then implies that the success probability of PPSZ increases if F has many critical variables. This is obtained by slightly modifying the existing analysis of [8] and [10] by taking critical variables into account. However, Lemma 9 is somewhat technical and we need to embed it into a review of the existing analysis. Theorem 1 is very simple, so we state it here:

▶ **Theorem 1.** *Let $p, q, c^* \in [0,1]$ and $a, b \geq 1$ such that $\frac{q}{b} = \left(1 - \frac{c^*}{2}\right) =: r$. Suppose algorithm $\mathcal{A}$ runs in time $a^n 2^{o(n)}$ and for every satisfiable $(\leq k)$-CNF formula F with $c(F) \geq c^*$ finds a satisfying assignment with probability at least $p^n \left(\frac{1}{2}\right)^{o(n)}$. Then there exists an algrotihm $\mathcal{A}'$ that runs in time $\max\{a,b\}^n 2^{o(n)}$ and for every satisfiable $(\leq k)$-CNF formula finds a satisfying assignment with probability at least $\min\{p,q\}^n \left(\frac{1}{2}\right)^{o(n)}$.*

Obviously we can turn $\mathcal{A}'$ into a algorithm that finds a satisfying assignment in expected time $\left(\frac{\max\{a,b\}}{\min\{p,q\}}\right)^n 2^{o(n)}$.

Proof. By *guessing* j variables we mean fixing in F j variables chosen uniformly at random to values chosen uniformly at random, obtaining the formula F' over at most $n - j$ variables. $\mathcal{A}'$ for each $j \in \{0, \ldots, n\}$ repeats the following b^j times: Guess j variables and then run $\mathcal{A}$ on F'; the running time bound is trivial. To bound the probability, we first claim that there

[1] Using the new version of [8] immediately gives the bound $\Omega(1.32267^{-n})$, as stated in [10].
[2] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1134

exists a j such that $a_j \geq \frac{r^j}{n+1}$ where a_j is the probability that after guessing j variables F' is satisfiable and $c(F') \geq c^*$. Suppose this is not the case: Let b_j be the probability that after guessing j variables F' is satisfiable and $c(F') < c^*$. Clearly $a_0 + b_0 = 1$ since F is satisfiable, and $a_{i+1} + b_{i+1} \geq b_i \cdot r$, as guessing one variable preserves satisfiability with probability at least $\left(1 - \frac{c^*}{2}\right) = r$. By the assumption, $b_i \cdot r \geq \left(a_i + b_i - \frac{r^i}{n+1}\right) \cdot r$; from this it is easy to show that $a_n + b_n \geq r^n - n\frac{r^n}{n+1} = \frac{r^n}{n+1}$. If $j = n$, we have $c(F') = 1$ by definition; hence $b_n = 0$ and $a_n \geq \frac{r^n}{n+1}$, a contradiction. Now let j^* be the j given by the claim; we repeat b^{j^*} times an algorithm that has success probability at least $\frac{r^{j^*}}{n+1} p^{n-j^*} \left(\frac{1}{2}\right)^{o(n)}$; as $r \cdot b = q$ this gives by a routine argument an algorithm with success probability at least $p^{n-j^*} q^{j^*} \left(\frac{1}{2}\right)^{o(n)}$. ◀

We improve the analysis for PPSZ for formulas with many critical variables. In combination with Theorem 1, this gives a success probability of $\Omega(1.32153^{-n})$ for 3-SAT and $\Omega(1.46928^{-n})$ for 4-SAT. Very recently, Iwama, Seto, Takai, and Tamaki [5] showed how to combine an improved version of Schöning's algorithm [4, 1] with PPSZ and achieved expected running time of $O(1.32113^n)$. We combine our improvement with theirs to obtain a bound of $O(1.32065^n)$. Due to page limitations, we were not able to use the full power of [5] in this version. We show a bound $O(1.321^n)$ that still improves on the bound of [5]. For a proof of the better bound, see the full version of this paper [3]. The only change is we use a better result of [5] which has different parameters; however these are not not stated explicitly so we needed to derive and prove them.

We analyze the algorithm Comb(F), where F is a CNF formula. Comb consists essentially of a call to PPSZ [8] and to Schoening [11]. In [6] it was shown that Comb has a better success probability than what the analysis of PPSZ and Schoening gives. Let ISTT be the algorithm of [5] that improves Comb.

▶ **Theorem 2.** *There exists an algorithm that for every satisfiable* 3*-CNF formula finds a satisfying assignment with probability* $\Omega(1.32153^{-n})$ *and runs in subexponential time.*

▶ **Theorem 3.** *There exists an algorithm that for every satisfiable* 3*-CNF formula finds a satisfying assignment with expected running time* $O(1.32065^{-n})$*.*

Due to page limitations, we prove the following weaker theorem instead. For the proof of the previous theorem, see the full version of this paper [3].

▶ **Theorem 4.** *There exists an algorithm that for every satisfiable* 3*-CNF formula finds a satisfying assignment with expected running time* $O(1.321^{-n})$*.*

▶ **Theorem 5.** *There exists an algorithm that for every satisfiable* 3*-CNF formula finds a satisfying assignment with probability* $\Omega(1.46928^{-n})$ *and runs in subexponential time.*

This is already very close to unique 4-SAT, which has a success probability of $\Omega(1.46899^{-n})$. The benefit of Theorem 1 is that when proving Theorems 2 and 5, we only need to consider formulas with many critical variables. For example, to prove Theorem 2, we choose c^* such that $1 - c^*/2 = 1/1.32153$, i.e., $c^* \approx 0.4866$. Then we have to bound from below the success probability of Comb for 3-CNF formulas F with $c(F) \geq c^*$.

1.2 Notation

We use the notational framework introduced in [12]. We assume an infinite supply of propositional *variables*. A *literal* u is a variable x or a complemented variable $\bar{x}$. A finite set

C of literals over pairwise distinct variables is called a *clause* and a finite set of clauses is a formula in *CNF* (Conjunctive Normal Form). We say that a variable x *occurs* in a clause C if either x or $\bar{x}$ are contained in it and that x occurs in the formula F if there is any clause where it occurs. We write $\mathrm{vbl}(C)$ or $\mathrm{vbl}(F)$ to denote the set of variables that occur in C or in F, respectively. A clause containing exactly one literal is called a *unit clause*. We say that F is a $(\leq k)$-*CNF* formula if every clause has size at most k. Let such an F be given and write $V := \mathrm{vbl}(F)$ and $n := |V|$.

A *assignment* is a function $\alpha : V \to \{0,1\}$ which assigns a Boolean value to each variable. A literal $u = x$ (or $u = \bar{x}$) is *satisfied by* α if $\alpha(x) = 1$ (or $\alpha(x) = 0$). A clause is *satisfied by* α if it contains a satisfied literal and a formula is *satisfied by* α if all of its clauses are. A formula is *satisfiable* if there exists a satisfying truth assignment to its variables.

For an assignment α on V and a set $W \subseteq V$, we denote by $\alpha \oplus W$ the assignment that corresponds to α on variables of $V \setminus W$ and is flipped on variables of W.

Given a CNF formula F, we denote by $\mathrm{sat}(F)$ the set of assignments that satisfy F.

Formulas can be manipulated by permanently assigning values to variables. If F is a given CNF formula and $x \in \mathrm{vbl}(F)$ then assigning $x \mapsto 1$ satisfies all clauses containing x (irrespective of what values the other variables in those closes are possibly assigned later) whilst it truncates all clauses containing $\bar{x}$ to their remaining literals.

We will write $F^{[x \mapsto 1]}$ (and analogously $F^{[x \mapsto 0]}$) to denote the formula arising from doing just this.

We say that two clauses C_1 and C_2 conflict on a variable x if one of them contains x and the other $\overline{x}$. We call C_1 and C_2 a resolvable pair if they conflict in exactly one variable x, and we define their *resolvent* by $R(C_1, C_2) := (C_1 \cup C_2) \setminus \{x, \overline{x}\}$. It is easy to see that if F contains a resolvable pair C_1, C_2, then $\mathrm{sat}(F) = \mathrm{sat}(F \cup \{R(C_1, C_2)\})$. A resolvable pair C_1, C_2 is s-bounded if $|C_1| \leq s$, $|C_2| \leq s$, and $|R(C_1, C_2)| \leq s$.

By $\textsc{Resolve}(F, s)$, we denote the set of clauses C that have an s-bounded resolution deduction from F. By a straightforward algorithm, we can compute $\textsc{Resolve}(F, s)$ in time $O\left(n^{3s}\mathrm{poly}\,(n)\right)$ [8].

By choosing an element u.a.r. from a finite set, we mean choosing it uniformly at random. By choosing an element u.a.r. from an closed real interval, we mean choosing it according to the continuous uniform distribution over this interval. Unless otherwise stated, all random choices are mutually independent.

We denote by log the logarithm to the base 2. For the logarithm to the base e, we write ln. We define $0 \log 0 := 0$.

2 Proof of the Main Theorems

In the following let $k \geq 3$ be a fixed integer. Let F be a satisfiable $(\leq k)$-CNF formula, $V := \mathrm{vbl}(F)$ and $n := |V|$. We first give the concepts from [8] needed to understand Lemma 9. Then we state the lemma and use it to improve the bounds on the success probability of Comb and ISTT given sufficiently many critical variables. In Section 3, we prove Lemma 9 and also consider 4-SAT. Most concepts used in the proof are from [8, 10]. Our contribution is to exploit what these concepts yield for critical variables.

Subcubes. For $D \subseteq V$ and $\alpha \in \{0,1\}^V$, the set $B(D, \alpha) := \{\beta \in \{0,1\}^V \mid \alpha(x) = \beta(x)\ \forall x \in D\}$ is called a *subcube*. The variables in D are called *defining* variables and those in $V \setminus D$ *nondefining* variables. The subcube $B(D, \beta)$ has dimension $|V \setminus D|$. For example, if $V = \{x_1, x_2, x_3\}$, $D = \{x_1, x_3\}$ and $\alpha = (1, 0, 0)$, then $B(D, \alpha)$ contains exactly the two

Algorithm 1 PPSZ(CNF formula F, assignment β, permutation π)

Let α be a partial assignment over $\mathrm{vbl}(F)$, initially the empty assignment.
$G \leftarrow \textsc{Resolve}(F, \log(|\mathrm{vbl}(F)|))$
for all $x \in \mathrm{vbl}(G)$, according to π **do**
  if $\{x\} \in G$ **then**
    $\alpha(x) \leftarrow 1$
  else if $\{\overline{x}\} \in G$ **then**
    $\alpha(x) \leftarrow 0$
  else
    $\alpha(x) \leftarrow \beta(x)$
  end if
  $G \leftarrow G^{[x \mapsto \alpha(x)]}$
end for
return α

Algorithm 2 PPSZ(CNF formula F)

{this algorithm is used for 4-SAT}
Choose $\beta(x)$ u.a.r. from all assignments on $\mathrm{vbl}(F)$
Choose π u.a.r. from all permutations of $\mathrm{vbl}(F)$
return $\mathrm{PPSZ}(F, \beta, \pi)$

assignments $(1,0,0)$ and $(1,1,0)$. Given a nonempty set $S \subseteq \{0,1\}^V$, there is a partition

$$\{0,1\}^V = \bigcup_{\alpha \in S} B_\alpha$$

where the B_α are pairwise disjoint subcubes, and $\alpha \in B_\alpha$ for all $\alpha \in S$. See [8] for a proof. For the rest of the paper, we fix such a partition for S being the set of satisfying assignments. To estimate the success probability of Comb, consider the assignment β that Comb chooses uniformly at random from $\{0,1\}^V$.

$$\begin{aligned} \Pr[\textsc{Comb}(F) \in \mathrm{sat}(F)] &= \sum_{\alpha \in \mathrm{sat}(F)} \Pr[\textsc{Comb}(F) \in \mathrm{sat}(F) |\ \beta \in B_\alpha] \cdot \Pr[\beta \in B_\alpha] \\ &\geq \min_{\alpha \in \mathrm{sat}(F)} \Pr[\textsc{Comb}(F) \in \mathrm{sat}(F)\ |\ \beta \in B_\alpha]. \end{aligned}$$

Hence instead of analyzing Comb for an assignment β sampled uniformly at random from all assignments, we fix $\alpha \in \mathrm{sat}(F)$ arbitrarily and we think of β as being sampled from the subcube B_α. Let N_α be the set of non-defining variables of this cube, and D_α the set of defining variables. Intuitively, if B_α has small dimension, then β is likely to be close to α, thus Schoening has a better success probability:

▶ **Lemma 6** ([6]). $\Pr[\textsc{Schoening}(F, \beta) \in \mathrm{sat}(F)\ |\ \beta \in B_\alpha] \geq (2 - 2/k)^{-|N_\alpha|}$.

Placements. As a next step, we analyze $\mathrm{PPSZ}(F, \beta, \pi)$ with β chosen uniformly at random from B_α and the permutation also chosen from some subset of permutations. A *placement* of the variables V is a function $\sigma : V \to [0,1]$, and a *uniform random placement* is defined by chosing $\sigma(x)$ uniformly at random from $[0,1]$ independently for each $x \in V$. With probability 1, a uniform random placement is injective and gives rise to a uniformly distributed permutation via the natural ordering $<$ on $[0,1]$. For the rest of the paper, we will

Algorithm 3 SCHOENING(CNF formula F, assignment β)

for $3|\mathrm{vbl}(F)|$ steps **do**
  if β satisfies F **then**
    return β
  end if
  Select an arbitrary $C \in F$ not satisfied by β
  Select a variable x u.a.r. from $\mathrm{vbl}(C)$ and flip x in β
end for
return β

Algorithm 4 COMB(CNF formula F)

{this algorithm is used for 3-SAT}
Choose $\beta(x)$ u.a.r. from all assignments on $\mathrm{vbl}(F)$
$\alpha \leftarrow \mathrm{PPSZ}(F, \beta)$
if $\alpha \notin \mathrm{sat}(F)$ **then**
  $\alpha \leftarrow \textsc{Schoening}(F, \beta)$
end if
return α

view π as a placement rather than a permutation. Let Γ be a measurable set of placements. Then

$$\Pr[\mathrm{PPSZ}(F,\beta,\pi) \in \mathrm{sat}(F) \mid \beta \in B_\alpha] \geq \Pr[\mathrm{PPSZ}(F,\beta,\pi) \in \mathrm{sat}(F) \mid \beta \in B_\alpha, \pi \in \Gamma] \cdot \Pr[\pi \in \Gamma].$$

The benefit of this is that we can tailor Γ towards our needs, i.e., making the conditional probability $\Pr[\mathrm{PPSZ}(F,\beta,\pi) \in \mathrm{sat}(F) \mid \beta \in B_\alpha, \pi \in \Gamma]$ fairly large. This may come at the cost of making $\Pr[\pi \in \Gamma]$ small.

Forced variables. Suppose the permutation π orders the variables V as $(x_1, \ldots, x_n)$. Let α be a satisfying assignment of F. Imagine we call $\mathrm{PPSZ}(F, \alpha, \pi)$. The algorithm applies bounded resolution to F, obtaining $G = \textsc{Resolve}(F, \log(n))$ and sets the variables $x_1, \ldots, x_n$ step by step to their respective values under α, creating a sequence of formulas by $G = G_0, G_1, \ldots, G_n$, where $G_i = G_{i-1}^{[x_i \mapsto \alpha(x_i)]}$ for $1 \leq i \leq n$. Since α is a satisfying assignment, G_n is the empty formula. We say x_i is *forced* with respect to α and π if G_{i-1} contains the unit clause $\{x_i\}$ or $\{\bar{x}_i\}$. By $\mathrm{forced}(\alpha, \pi)$ we denote the set of variables x that are forced with respect to α and π. If x is not forced, we say it is *guessed*. We denote by $\mathrm{guessed}(\alpha, \pi)$ the set of guessed variables. Note that $\mathrm{PPSZ}(F, \beta, \pi)$ returns α if and only if $\alpha(x) = \beta(x)$ for all $x \in \mathrm{guessed}(\alpha, \pi)$. Furthermore, since β is chosen uniformly at random from B_α, we already have $\alpha(x) = \beta(x)$ for all $x \in D_\alpha$. Therefore

$$\begin{aligned} \Pr[\mathrm{PPSZ}(F,\beta,\pi) \in \mathrm{sat}(F)] &\geq \Pr[\mathrm{PPSZ}(F,\beta,\pi) = \alpha] && (1) \\ &= \mathbf{E}\left[2^{-|N_\alpha \cap \mathrm{guessed}(\alpha,\pi)|}\right] \\ &\geq 2^{-\mathbf{E}[|N_\alpha \cap \mathrm{guessed}(\alpha,\pi)|]}, && (2) \end{aligned}$$

where the inequality comes from Jensen's inequality applied to the convex function $t \mapsto 2^{-t}$. Note that (2) holds when taking π uniformly at random as well as when sampling it from

some set Γ. Using linearity of expectation, we see that

$$\mathbf{E}[|N_\alpha \cap \text{guessed}(\alpha,\pi)|] = \sum_{x\in N_\alpha} \Pr[x \in \text{guessed}(\alpha,\pi)]. \tag{3}$$

Now if α is the unique satisfying assignment, then $N_\alpha = V$. For 3-SAT, one central result of [8] is that

▶ **Lemma 7** ([8]). *Let F be a satisfiable 3-CNF formula with a unique satisfying assignment α. Then for every $x \in \text{vbl}(F)$, it holds that $\Pr[x \in \text{guessed}(\alpha,\pi)] \leq 2\ln(2) - 1 + o(1) < 0.3863$.*

Combining the lemma with (2) shows that PPSZ on 3-CNF formulas with a unique satisfying assignment has a success probability of at least $2^{-(2\ln(2)-1+o(1))n} \in \Omega(1.308^{-n})$. For the case of multiple satisfying assignments, the lemma does not hold anymore.

Critical variables. Let F be a satisfiable CNF formula and x a variable. Recall that we call x *critical* if all satisfying assignments of F agree on x. The following observation is not difficult to show:

▶ **Observation 8.** *Let F be a satisfiable CNF formula and let V_C be the set of critical variables. Let B_α be the subcube as defined above. For a satisfying assignment α, let N_α be the set of nondefining variables. Then $V_C \subseteq N_\alpha$.*

▶ **Lemma 9.** *Let F be a satisfiable 3-CNF formula and α be a satisfying assignment. There is a measurable set $\Gamma \subseteq [0,1]^V$ of placements such that for $\beta = 0.8022563838$ and $\gamma = 0.6073995502$, we have*

1. $\Pr[\pi \in \Gamma] \geq 2^{-\beta|D_\alpha| - o(n)} \approx 0.57345159^{|D_\alpha| - o(n)}$,
2. $\Pr[x \in \text{forced}(\alpha,\pi) \mid \pi \in \Gamma] \geq \gamma - o(1) \approx 0.6073995502 - o(1)$ *for all* $x \in N_\alpha$,
3. $\Pr[x \in \text{forced}(\alpha,\pi) \mid \pi \in \Gamma] \geq 2 - 2\ln(2) - o(1) \approx 0.6137056$ *for all critical* $x \in V$.

The important part of the lemma is point 3, namely that critical variables are forced with a larger probability than non-critical ones.

Proof of Theorem 2. Using Theorem 1, we can assume $c(F) \geq 0.48659459$. Let $\Delta := |D_\alpha|/|V| = 1 - |N_\alpha|/|V|$ be the fraction of defining variables. Combining (3) with Lemma 9, we obtain

$$\begin{aligned}
\mathbf{E}[|N_\alpha \cap \text{guessed}(\alpha,\pi)| \mid \pi \in \Gamma] &= \sum_{x\in N_\alpha} \Pr[x \in \text{guessed}(\alpha,\pi)] \\
&\leq (2\ln 2 - 1)|V_C| + (1-\gamma)|N_\alpha \setminus V_C| + o(n) \\
&\leq (2\ln 2 - 1)c^* n + (1-\gamma)(1-\Delta-c^*)n + o(n) \\
&= 0.389532n - 0.3926004498\Delta n + o(n).
\end{aligned}$$

The expected fraction of nondefining variables we have to guess is thus a little bit larger than in the case of a unique satisfying assignment, where it is ≈ 0.3863. Together with (2), we conclude that the success probability of PPSZ is at least

$$\begin{aligned}
\Pr[\text{PPSZ}(F,\beta,\pi) = \alpha \mid \beta \in B_\alpha] &\geq \Pr[\text{PPSZ}(F,\beta,\pi) = \alpha \mid \beta \in B_\alpha, \pi \in \Gamma] \cdot \Pr[\pi \in \Gamma] \\
&\geq 2^{-\mathbf{E}[|N_\alpha \cap \text{guessed}(\alpha,\pi)| \mid \pi\in\Gamma]} \cdot \Pr[\pi \in \Gamma] \\
&\geq 2^{-0.389532n + 0.3926004498\Delta n} \cdot 0.57345159^{\Delta n} \cdot 2^{-o(n)} \\
&\geq 1.3099684^{-n} \cdot 1.328369^{-\Delta n} \cdot 2^{-o(n)}. \qquad (4)
\end{aligned}$$

Our bound on the success probability of PPSZ thus deteriorates with the number of defining variables. A bigger subcube B_α is better for PPSZ. We combine this with the bound for Schöning's algorithm from Iwama and Tamaki [6], stated above in Lemma 6

$$\Pr[\textsc{Schoening}(F,\beta) \in \mathrm{sat}(F) \mid \beta \in B_\alpha] \geq (2-2/k)^{-(1-\Delta)n}. \tag{5}$$

The combined worst case is with $\Delta \approx 0.0309273$, in which case both (4) and (5) evaluate to $\Omega(1.32153^{-n})$. Therefore for any Δ, at least one of Schoening and PPSZ has a success probability of $\Omega(1.32153^{-n})$. ◀

Proof of Theorem 4. Lemma 6 from [5] tells us that there is an algorithm ISTTSch that improves Schoening such that for all $m^* \in [0, \frac{1}{3}]$ we have, after preprocessing time $6^{m^* n}$,

$$\Pr[\textsc{ISTTSch}(F,\beta) \in \mathrm{sat}(F) \mid \beta \in B_\alpha] \geq 1.012795^{m^* \cdot n} \cdot 1.2845745^{\Delta n} \cdot (3/4)^n.$$

We want to prove that by replacing Schoening with ISTTSch in Comb, we obtain expected running time of $O(1.321^n)$. Setting $c^* := 0.48599$ and $m^* := 0.155371873$ gives $1 - c^*/2 \geq 1/1.321$ and $6^{m^*} \geq 1.321$. With this choice of c^*, we have the following bound for PPSZ (obtained as in the previous proof, but with a different constant c^*):

$$\Pr[\mathrm{PPSZ}(F,\beta,\pi) = \alpha \mid \beta \in B_\alpha] \geq 1.31^{-n} \cdot 1.3312^{-\Delta n} \cdot 2^{-o(n)}.$$

The combined worst case is at $\Delta \approx 0.029225$ where $1.31^{-n} \cdot 1.3312^{-\Delta n} > 1.321^{-n}$ and $1.012795^{m^* \cdot n} \cdot 1.2845745^{\Delta n} \cdot (3/4)^n > 1.321^{-n}$, proving that the combined success probability is $\Omega(1.321^{-n})$ (after preprocessing time $O(1.321^n)$). ◀

3 Proof of Lemma 9

3.1 Critical Clause Trees

Let $G := \textsc{Resolve}(F, \log(n))$. Note that $\mathrm{vbl}(F) = \mathrm{vbl}(G)$ and $\mathrm{sat}(F) = \mathrm{sat}(G)$. A *critical clause* for $x \in V$ w.r.t. α is a clause where α satisfies exactly one literal and this literal is over x. It can be easily seen that if the output of PPSZ should be α, then exactly the critical clauses of G are the clauses that might turn into unit clauses. Note that the *defining* variables are assumed to be set correctly, so we only need to consider critical clauses for *nondefining* variables here.

We now define critical clause trees, a concept that tells us which critical clauses we can expect in a CNF formula after bounded resolution. Let T be a rooted tree in which every node is either labeled with a variable from V or is unlabeled. A *cut* in a rooted tree is a set of nodes A such that the root is not in A and every path from the root to a leaf contains at least one node in A. The *depth* of a node is the distance to the root. For a set A of nodes, $\mathrm{vbl}(A)$ denotes the set of variables occurring as labels in A. We say T is a *critical clause tree* for x w.r.t. G and α if the following properties hold:

1. The root is labeled by x.
2. On any path from the root to a leaf, no two nodes have the same label.
3. For any cut A of the tree, there is a critical clause $C \in G$ w.r.t. α where the satisfied literal is over x and every unsatisfied literal is over some variable in $\mathrm{vbl}(A)$.

It is shown in [8] that we can construct a critical clause tree for $x \in N_\alpha$ as follows: Start with the root labeled x. Now we can repeatedly extend a leaf node v. Let L be the set of labels that occur on the path from v to the root. If $\alpha \oplus L$ does not satisfy F, then we can extend the tree at that node: There is a clause C in F (not in G) not satisfied by $\alpha \oplus L$. For each literal in C that is not satisfied by α, we add a child to v labeled with the variable of that literal. If there are no such literals, we add an unlabeled node. As clauses of F have at most k literals, each node has at most $k-1$ children. If the constructed tree has at most $\log(n)$ nodes (as we do $\log(n)$-bounded resolution), then it is a critical clause tree for x w.r.t. G and α.

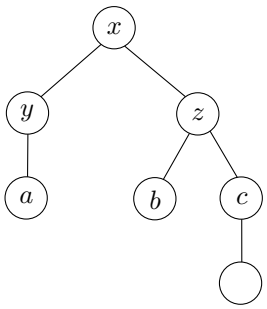

Figure 1 Example Critical Clause Tree

We give a simple example: Let

$$F := \{\{x, \overline{y}, \overline{z}\}, \{x, y, \overline{a}\}, \{z, \overline{b}, \overline{c}\}, \{x, z, c\}\}.$$

For the all-one assignment and x, we can get the tree shown in Figure 1 by the described procedure. $\{a, b\}$ is a cut in this tree. We have $R(\{z, \overline{b}, \overline{c}\}, \{x, z, c\}) = \{x, z, \overline{b}\}$, $R(\{x, \overline{y}, \overline{z}\}, \{x, y, \overline{a}\}) = \{x, \overline{z}, \overline{a}\}$ and $R(\{x, z, \overline{b}\}, \{x, \overline{z}, \overline{a}\}) = \{x, \overline{a}, \overline{b}\}$, giving the required critical clause.

If α is the only satisfying assignment of F, $\alpha \oplus L$ never satisfies F, and we can build a tree where all leafs are at depth $d := \lfloor \log_k \log(n) \rfloor$. We call this a *full tree*. The important observation is now that this also works if x is a *critical variable*, as in that case $\alpha \oplus L$ also never satisfies F, as $x \in L$.

In the general case, however, the assignment $\alpha \oplus L$ might satisfy F so that we cannot extend the tree. However if L consists only of *nondefining* variables, then we know that $\alpha \oplus L$ does not satisfy F. Hence we can get a tree where every leaf not at depth d is labeled by a *defining* variable. We define the trees T_x we will use in the analysis:

▶ **Definition 10.** For $x \in N_\alpha$, construct the critical clause tree for x as follows: If x is a critical variable, then construct T_x such that all leaves are at depth d, i.e., construct a full tree. Otherwise, construct T_x such that all leaves not labeled by defining variables are at depth d.

This means that a tree might just consist of a root where all children are labeled with defining variables, which essentially nullifies the benefits from resolution. To cope with this, we have to make defining variables more likely to occur at the beginning. We achieve this by choosing the set Γ of placements whose existence we claim in Lemma 9 in a way such that exactly that happens.

▶ **Definition 11.** A function $H : [0, 1] \to [0, 1]$ is called a *nice distribution function* if H is non-decreasing, uniformly continuous, $H(0) = 0$, $H(1) = 1$, H is differentiable except for finitely many points and $H(r) \geq r$.

Compared with [8], we added the requirement $H(r) \geq r$. This will mean that defining variables cannot be less likely to occur at the beginning than nondefining variables. We now define a random placement where defining variables are placed with distribution function H:

▶ **Definition 12.** Let H be a nice distribution function. By π_H, we define the random placement on V s.t. $\pi(x)$ for $x \in N_\alpha$ is u.a.r. $\in [0, 1]$, and for $x \in D_\alpha$ and $r \in [0, 1]$, $\Pr(\pi(x) \leq r) = H(r)$.

Assume that the variables are processed according to some placement π. Consider T_x. If there is a cut A such that $\pi(y) < \pi(x)$ for every $y \in \mathrm{vbl}(A)$, then x is forced, as the corresponding critical clause has turned into a unit clause for x. Denote the probability that $S_x(\pi)$ is a cut in T_x by $Q(T_x, \pi)$.

For $r \in [0,1]$, let $R_k(r)$ be the smallest non-negative x that satisfies $x = (r + (1-r)x)^{k-1}$ and $R_k := \int_0^1 R_k(r)dr$. It was shown in [8] that if T_x is a full tree, then

$$Q(T_x, \pi_U) \geq R_k - o(1).$$

$R_k(r)$ can be understood as follows: Take an infinite $(k-1)$-ary tree and mark each node as "dead" with probability r, except the root. $R_k(r)$ is the probability that this tree contains an infinite path that starts at the root and contains only "alive" nodes.

We have $R_3 = 2 - 2\ln 2 \approx 0.6137$ and $R_4 \approx 0.4451$. For $r \in [0, \frac{1}{2}]$, we have $R_3(r) = \left(\frac{r}{1-r}\right)^2$ and for $r \in [\frac{1}{2}, 1]$, we have $R_3(r) = 1$. As $H(r) \geq r$, and by definition of π_H and of a cut, it is obvious that

$$Q(T_x, \pi_H) \geq R_k - o(1), \tag{6}$$

if T_x is a full tree. If T_x is not a full tree, we do not have any good bounds on $Q(T_x, \pi_U)$. In [10] it is shown that if T_x is not necessarily a full tree, but a tree in which every leaf not at depth d is labeled by a defining variable, then

$$Q(T_x, \pi_H) \geq \gamma_H - o(1), \tag{7}$$

where

$$\gamma_H = \int_0^1 \min\{H(r)^{k-1}, R_k(r)\}dr.$$

Obviously $\gamma_H \leq R_k$, which means that the bound (6) for full trees is at least as strong as the bound (7) for general trees. The $H(r)^{k-1}$ term corresponds to the tree that consists of a root where all children are labeled with defining variables and are thus leaves (remember that there are at most $k-1$ children). It takes a small lemma to show that this tree and the full tree are the worst cases. See [2] for details. The following observation summarizes this:

▶ **Observation 13.** *If x is a* critical *variable, then $Q(T_x, \pi_H) \geq R_k - o(1)$. If x is a noncritical* nondefining *variable, then $Q(T_x, \pi_H) \geq \gamma_H - o(1)$.*

We want to find a set Γ of placements such that a placement chosen uniformly at random from Γ behaves more or less like π_H.

▶ **Lemma 14** (old version of [8])**.** *Let H be a nice distribution function. If $|D_\alpha| \geq \sqrt{n}$, there is a set of placements Γ depending on n with the following properties: Let π_Γ be the placement choosen uniformly at random from Γ. Then for any tree T with at most $\log(n)$ nodes we have*

$$Q(T, \pi_\Gamma) \geq Q(T, \pi_H) - o(1)$$

and

$$Pr(\pi_U \in \Gamma) \geq 2^{-\beta_H |D_\alpha| - o(n)}$$

with

$$\beta_H := \int_0^1 h(r) \log\left(h(r)\right) dr$$

where $h(r)$ is the derivative of $H(r)$.

The proof of this lemma is long and complicated, see Sections 4.2 and 4.3 in [2]. The case $|D_\alpha| < \sqrt{n}$ is easy to handle: The probability that all defining variables come at the beginning is substantial, and we are essentially in the (good) unique case.

Below we will show how to choose a good function H for the case $k = 3$ and $k = 4$. To get an intuition, see Figure 2 for a plot of H for $k = 3$. With this function, one obtains $\gamma_H \approx 0.6073995502$ and $\beta_H \approx 0.8022563838$. Together with Lemma 14 and Observation 13, we conclude that for a *critical* variable x

$$\Pr[x \in \text{forced}(\alpha, \pi)] \geq Q(T_x, \pi_H) - o(1) \geq R_k - o(1) \geq 0.61371,$$

and for a non-critical non-defining variable x

$$\Pr[x \in \text{forced}(\alpha, \pi)] \geq Q(T_x, \pi_H) \geq \gamma_H - o(1) \geq 0.6073995502 - o(1).$$

3.2 Choosing a good H

3-SAT. Let now $k = 3$. We choose H as in [10]: Let $\theta \in [0.5, 1]$ be a parameter. With some appropriate parameters a and $b > 1$, we define $H(r)$ as follows:

$$H(r) := \begin{cases} r/\theta & \text{if } r \in [0, 1-\theta) \\ 1 - (-a \ln(r))^b & \text{if } r \in [1-\theta, 1] \end{cases}$$

To determine a and b, we set the constraints

$$H(1-\theta) = R_3(1-\theta)^{1/2}$$

(as $\theta \geq 1/2$, this right-hand side is equal to $\frac{1-\theta}{\theta}$) and

$$h(1-\theta) = 1/\theta.$$

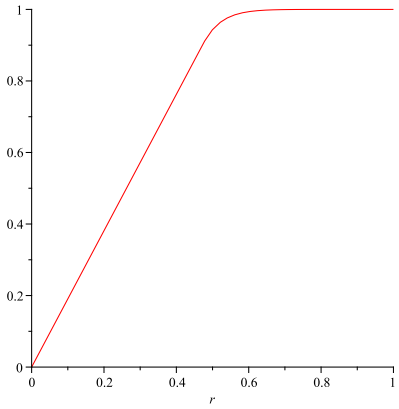

Figure 2 $H(r)$ for 3-SAT

If these constraints are satisfied, $H(r)$ is a nice distribution function that is differentiable on $[0, 1]$. Figure 2 gives a plot of the $H(r)$ we use. Numerical optimization gives $\theta \approx 0.52455825$ and as before $c^* \approx 0.48659459$. See Section 4.6 in [2] for details of the computation. This gives

$$a \approx 0.96782885577,$$

$$b \approx 7.19709520894,$$

$$\beta_H \leq 0.8022563838,$$

$$\gamma_H \geq 0.6073995502.$$

This concludes the proof of Lemma 9.

4-SAT. For 4-SAT, we use the H corresponding to the new version of [8]. For some parameter $\theta \in [\frac{2}{3}, 1]$, we let $H(r) := \min\{\frac{r}{\theta}, 1\}$. It turns out that the optimum is when $\beta_H = 1 - \gamma_H$. In that case it is easily seen that the bound for PPSZ does not depend on $|D_\alpha|$, and hence we do not need Schoening. Numerical optimization gives $\theta \approx 0.6803639$ and $c^* \approx 0.63878808$. This implies the success probability $\Omega(1.46928^{-n})$, proving Theorem 5.

4 Conclusion

We have shown how to improve PPSZ by a preprocessing step that guarantees that a substantial fraction of variables will be critical. With this, we were able to improve the bound for 3-SAT and 4-SAT from [10]. We have also shown that our approach nicely combines with the improvement by [5] by giving an even better bound. In 4-SAT, we are already very close to the unique case. We do not know if a more refined choice of H (similar to [10]), possibly depending on Δ, allows us to close that gap.

It is interesting to see that we could make use of multiple assignments in the guessing step before considering just one assignment using the subcube partition.

Acknowledgments

We thank Emo Welzl for many fruitful discussions and continous support and Konstantin Kutzkov for pointing us to [5].

References

1 Sven Baumer and Rainer Schuler. Improving a probabilistic 3-SAT algorithm by dynamic search and independent clause pairs. In *Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pages 150–161. Springer Berlin / Heidelberg, 2004.

2 Timon Hertli. Investigating and improving the PPSZ algorithm for SAT, master's thesis. ETH Zürich, 2010. doi: `http://dx.doi.org/10.3929/ethz-a-006206989`.

3 Timon Hertli, Robin A. Moser, and Dominik Scheder. Improving PPSZ for 3-SAT using critical variables. *CoRR*, abs/1009.4830, 2010.

4 Thomas Hofmeister, Uwe Schöning, Rainer Schuler, and Osamu Watanabe. A probabilistic 3-SAT algorithm further improved. In *STACS 2002*, volume 2285 of *Lecture Notes in Comput. Sci.*, pages 192–202. Springer, Berlin, 2002.

5 Kazuo Iwama, Kazuhisa Seto, Tadashi Takai, and Suguru Tamaki. Improved randomized algorithms for 3-SAT. In *Algorithms and Computation*, volume 6506 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin / Heidelberg, 2010.

6 Kazuo Iwama and Suguru Tamaki. Improved upper bounds for 3-SAT. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 328–329 (electronic), New York, 2004. ACM.

7 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An Improved Exponential-Time Algorithm for k-SAT. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 628–637. IEEE Computer Society, 1998.

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

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

10 Daniel Rolf. Improved Bound for the PPSZ/Schöning-Algorithm for 3-SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:111–122, 2006.

11 Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science (New York, 1999)*, pages 410–414. IEEE Computer Soc., Los Alamitos, CA, 1999.

12 Emo Welzl. Boolean satisfiability – combinatorics and algorithms (lecture notes), 2005. `http://www.inf.ethz.ch/~emo/SmallPieces/SAT.ps`.