Breaking the PPSZ Barrier for Unique 3-SAT

Timon Hertli[*]

February 18, 2014

Abstract

The PPSZ algorithm by Paturi, Pudlák, Saks, and Zane (FOCS 1998) is the fastest known algorithm for (Promise) Unique k-SAT. We give an improved algorithm with exponentially faster bounds for Unique 3-SAT.

For uniquely satisfiable 3-CNF formulas, we do the following case distinction: We call a clause critical if exactly one literal is satisfied by the unique satisfying assignment. If a formula has many critical clauses, we observe that PPSZ by itself is already faster. If there are only few clauses allover, we use an algorithm by Wahlström (ESA 2005) that is faster than PPSZ in this case. Otherwise we have a formula with few critical and many non-critical clauses. Non-critical clauses have at least two literals satisfied; we show how to exploit this to improve PPSZ.

1 Introduction

The well-known problem k-SAT is NP-complete for $k \geq 3$. If P$\neq$NP, k-SAT does not have a polynomial time algorithm. For a CNF formula F over n variables, the naive approach of trying all satisfying assignments takes time $O(2^n \cdot \text{poly}(|F|))$. Especially for $k = 3$ much work has been put into finding so-called "moderately exponential time" algorithms running in time $O(2^{cn})$ for some $c < 1$. In 1998, Paturi, Pudlák, Saks, and Zane presented a randomized algorithm for 3-SAT that runs in time $O(1.364^n)$. Given the promise that the formula has at most one satisfying assignment (that problem is called Unique 3-SAT), a running time of $O(1.308^n)$ was shown. Both bounds were the best known when published. The running time of general 3-SAT has been improved repeatedly (e.g. [8, 5]), until PPSZ was shown to run in time $O(1.308^n)$ for general 3-SAT [3].

Any further improvement of 3-SAT further also improves Unique 3-SAT; however that bound has not been improved upon since publication of the PPSZ algorithm. In this paper, we present a randomized algorithm for Unique 3-SAT with exponentially better bounds than what could be shown for PPSZ. Our algorithm builds on PPSZ and improves it by treating sparse and dense formulas differently.

A key concept of the PPSZ analysis is the so-called critical clause: We call a clause critical for a variable x if exactly one literal is satisfied by this unique satisfying assignment, and that literal is over x. It is not hard to see that the uniqueness of the satisfying assignment implies that every variable has at least one critical clause. If some variables have strictly *more* than one critical clause, then we will give a straightforward proof that PPSZ by itself is faster already. Hence the bottleneck of PPSZ is when every variable has *exactly* one critical clause, and in total there are exactly n critical clauses.

Given a formula with exactly n critical clauses, consider how many other (non-crtical) clauses there are. If there are few, we use an algorithm by Wahlström [9] that is faster than PPSZ for formulas with few clauses allover. If there are many non-critical clauses we use the following fact: A non-critical clause has

---

[*]Institute for Theoretical Computer Science, Department of Computer Science, ETH Zürich, 8092 Zürich, Switzerland (`timon.hertli@inf.ethz.ch`)

two or more satisfied literals (w.r.t. unique satisfying assignment); so after removing a literal, the remaining 2-clause is still satisfied. We will exploit this to improve PPSZ.

An remaining problem is the case if only very few (i.e. sublinearly many) variables have more than one critical clause or appear in many (non-critical) clauses. In this case, we would get only a subexponential improvement. A significant part of our algorithm deals with this problem.

1.1 Notation

We use the notational framework introduced in [11]. Let V be a finite set of propositional *variables*. A *literal u* over $x \in V$ is a variable x or a negated variable $\bar{x}$. If $u = \bar{x}$, then $\bar{u}$, the negation of u, is defined as x. We mostly use x, y, z for variables and u, v, w for literals. We assume that all literals are distinct. A *clause* over V is a finite set of literals over pairwise distinct variables from V. By $\text{vbl}(C)$ we denote the set of variables that occur in C, i.e. $\{x \in V \mid x \in C \vee \bar{x} \in C\}$. C is a k-clause if $|C| = k$ and it is a $(\leq k)$-clause if $|C| \leq k$. A formula in *CNF* (Conjunctive Normal Form) F over V is a finite set of clauses over V. We define $\text{vbl}(F) := \bigcup_{C \in F} \text{vbl}(C)$. F is a k-CNF formula (a $(\leq k)$-CNF formula) if all clauses of F are k-clauses $((\leq k)$-clauses). A (truth) *assignment* on V is a function $\alpha : V \to \{0, 1\}$ which assigns a Boolean value to each variable. α extends to negated variables by letting $\alpha(\bar{x}) := 1 - \alpha(x)$. A literal u is *satisfied by α* if $\alpha(u) = 1$. 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. A formula that is not satisfiable is called *unsatisfiable*. Given a CNF formula F, we denote by $\text{sat}(F)$ the set of assignments on $\text{vbl}(F)$ that satisfy F. *k-SAT* is the decision problem of deciding if a $(\leq k)$-CNF formula has a satisfying assignment.

If F is a CNF formula and $x \in \text{vbl}(F)$, we write $F^{[x \to 1]}$ (analogously $F^{[x \to 0]}$) for the formula arising from removing all clauses containing x and truncating all clauses containing $\bar{x}$ to their remaining literals. This corresponds to assigning x to 1 (or 0) in F and removing trivially satifsied clauses. We call assignments α on V and β and W *consistent* if $\alpha(x) = \beta(x)$ for all $x \in V \cap W$. If α is an assignment on V and $W \subseteq V$, we denote by $\alpha|_W$ the assignment on W with $\alpha|_W(x) = \alpha(x)$ for $x \in W$. If $\gamma = \{x \mapsto 0, y \mapsto 1, \dots\}$, we write $F^{[\gamma]}$ as a shorthand for $F^{[x \to 0][y \to 1] \dots}$, the *restriction* of F to γ.

For a set W, we denote by $x \leftarrow_{\text{u.a.r.}} W$ choosing an element x u.a.r. (uniformly at random). 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. By $\text{poly}(n)$ we denote a polynomial factor depending on n. We use the following convention if no confusion arises: When F is a CNF formula, we denote by V its variables and by n the number of variables of F, i.e. $V := \text{vbl}(F)$ and $n := |\text{vbl}(F)|$. By $o(1)$ we denote a quantity dependent on n going to 0 with $n \to \infty$.

1.2 Previous Work

Definition 1.1. *(Promise) Unique 3-SAT is the following promise problem: Given a (≤ 3)-CNF with at most one satisfying assignment, decide if it is satisfiable or unsatisfiable.*

A randomized algorithm for Unique 3-SAT is an algorithm that, for a uniquely satisfiable (≤ 3)-CNF formula returns the satisfying assignment with probability $\frac{1}{2}$.

Note that if the formula is not satisfiable, there is no satisfying assignment, and the algorithm cannot erroneously find one. Hence the error is one-sided and we don't have to care about unsatisfiable formulas.

The PPSZ algorithm [7] is a randomized algorithm for Unique 3-SAT running in time $O(1.308^n)$. The precise bound is as follows:

Definition 1.2. *Let $S := \int_0^1 \left(1 - \min\{1, \frac{r^2}{(1-r)^2}\} \right) dr = 2\ln 2 - 1$.*

2

Algorithm 1 PPSZ(CNF formula F)

$V \leftarrow \text{vbl}(F); n \leftarrow |V|$
Choose β u.a.r. from all assignments on V
Choose $\pi : V \rightarrow [0,1]$ as a random placement of V
Let α be a partial assignment on V, initially empty
for $x \in V$, in ascending order of $\pi(x)$ **do**
   if F $(\log n)$-implies x or $\bar{x}$, set $\alpha(x)$ to satisfy this literal
   otherwise $\alpha(x) \leftarrow \beta(x)$ {guess $\alpha(x)$ u.a.r.}
   $F \leftarrow F^{[x \mapsto \alpha(x)]}$
end for
return α

Theorem 1.3 ([7])**.** *There exists a randomized algorithm (called* PPSZ*) for Unique 3-SAT running in time* $2^{(S+o(1))n}$.

Note that $0.3862 < S < 0.3863$ and $2^S < 1.308$.

1.3 Our Contribution

For Unqiue 3-SAT, we get time bounds exponentially better than PPSZ:

Theorem 1.4. *There exists a randomized algorithm for Unique 3-SAT running in time* $2^{(S-\varepsilon_2+o(1))n}$ *where* $\varepsilon_2 = 10^{-24}$.

In Section 2, we review the PPSZ algorithm. In Section 3, we show that the worst case for PPSZ occurs when every variable has exactly one critical 3-clause; this case we improve in Section 4. In Section 5, we pose open problems that arise.

2 The PPSZ Algorithm

In this section we review the PPSZ algorithm [7], summarized in Algorithm 1. We need to adapt some statements slightly. For the straightforward but technical proofs we refer the reader to the appendix. The following two definitions are used to state the PPSZ algorithm.

Definition 2.1. *A CNF formula F D-implies a literal u if there exists a subformula* $G \subseteq F$ *with* $|G| \leq D$ *and all satisfying assignments of G set u to* 1.

In a random permutation, the positions of two elements are not independent. To overcome this, placements were defined. They can be seen as continuous permutations with the nice property that the places of different elements are independent.

Definition 2.2 ([7])**.** *A placement on V is a mapping* $V \rightarrow [0,1]$. *A random placement is obtained by choosing for every* $x \in V$ $\pi(x)$ *uniformly at random from* $[0,1]$, *independently.*

Observation 2.3. *By symmetry and as ties happen with probability* 0, *ordering V according to a random placement gives a permutation distributed the same as a permutation drawn uniformly at random from the set of all permutations on V.*

The analysis of PPSZ builds on the concept of forced and guessed variables:

Definition 2.4. *If in PPSZ, $\alpha(x)$ is assigned 0 or 1 because of D-implication, we call x* forced*. Otherwise (if $\alpha(x)$ is set to $\beta(x)$), we call x* guessed.

The following lemma from [7] relates the expected number of guessed variables to the success probability (the proof is by an induction argument and Jensen's inequality).

Lemma 2.5 ([7])**.** *Let F be a satisfiable (≤ 3)-CNF, let α^* be a satisfying assignment. Let $G(\pi)$ be the expected number of guessed variables conditioned on $\beta = \alpha^*$ depending on π. Then $\mathrm{PPSZ}(F)$ returns α^* with probability at least $\mathbf{E}_\pi[2^{-G(\pi)}] \geq 2^{\mathbf{E}_\pi[-G(\pi)]}$.*

Remember that $S := \int_0^1 \left(1 - \min\{1, \frac{r^2}{(1-r)^2}\}\right) dr = 2\ln 2 - 1$, which corresponds to the probability that a variable is guessed. We define S_p where the integral starts from p instead of 0; this corresponds to the probability that a variable has place at least p and is guessed.

Definition 2.6. *Let $S_p := \int_p^1 \left(1 - \min\{1, \frac{r^2}{(1-r)^2}\}\right) dr$.*

Observation 2.7. *For $p \leq \frac{1}{2}$, $S_p = S - p + \int_0^p \frac{r^2}{(1-r)^2} dr$.*

In the appendix, we derive from [7] the following:

Corollary A.3. *Let F a (≤ 3)-CNF with unique satisfying assignment α. Then in PPSZ(F) conditioned on $\beta = \alpha$, the expected number of guessed variables is at most $(S + o(1))n$.*

Furthermore, suppose we pick every variable of F with probability p, independently, and let V_p be the resulting set. Then in PPSZ(F) conditioned on $\beta = \alpha$, the expected number of guessed variables is at most $(S_p + o(1))n$.

By Lemma 2.5, we have the following corollary:

Corollary 2.8. *Let F a (≤ 3)-CNF with unique satisfying assignment α. Then the probability that $\mathrm{PPSZ}(F)$ returns α is at least $2^{(-S-o(1))n}$.*

Furthermore, suppose we pick every variable of F with probability p, independently, and let V_p be the resulting set. Then the expected $\log$ of the probability (over the choice of V_p) that $\mathrm{PPSZ}(F^{[\alpha|_{V_p}]})$ returns $\alpha|_{V \setminus V_p}$ is at least $(-S_p - o(1))n$.

The first statement is actually what is shown in [7], and the second statement is a direct consequence. We need this later when we replace PPSZ by a different algorithm on variables with place at most p. It is easily seen that for a (≤ 3)-CNF F, $\mathrm{PPSZ}(F)$ runs in time $2^{o(n)}$. Hence by a standard repetition argument, PPSZ gives us an algorithm finding an assignment in time $2^{(S+o(1))n}$ and we (re-)proved Theorem 1.3.

3 Reducing to One Critical Clause per Variable

In this section we show that an exponential improvement for the case where every variable has exactly one critical clause gives an exponential improvement for Unique 3-SAT.

Definition 3.1 ([7])**.** *Let F be a CNF formula satisfied by α. We call a clause C* critical *for x (w.r.t. α) if α satisfies exactly one literal of C, and this literal is over x.*

Definition 3.2. *A 1C-Unique (≤ 3)-CNF is a uniquely satisfiable (≤ 3)-CNF where every variable has at most one critical clause. Call the corresponding promise problem 1C-Unique 3-SAT.*

4

All formulas we consider have a unique satisfying assignment; critical clauses will be always w.r.t. that. First we show that a variables with more than one critical clause are guessed less often; giving an exponential improvement for formulas with a linear number of such variables. A similar statement for shorter critical clauses is required in the next section.

Lemma 3.3. *Let F be a (≤ 3)-CNF uniquely satisfied by α. A variable x with at least two critical clauses (w.r.t. α) is guessed given $\beta = \alpha$ with probability at most $S - 0.0014 + o(1)$. Furthermore, a variable x with a critical (≤ 2)-clause is guessed with probability at most $S - 0.035 + o(1)$*

Proof. Suppose $\pi(x) = r$. Let C_1 and C_2 be two critical clauses of x. If C_1 and C_2 share no variable besides x, then the probability that x is forced is at least $2r^2 - r^4$ by the inclusion-exclusion principle. If C_1 and C_2 share one variable besides x, then the probability that x is forced is at least $2r^2 - r^3$ (which is smaller than $2r^2 - r^4$. C_1 and C_2 cannot share two varibles besides x: in that case $C_1 = C_2$, as being a critical clause for x w.r.t. α predetermines the polarity of the literals. Intutiviely, if r is small, then $2r^2 - r^3$ is almost twice as large as $\frac{r^2}{(1-r)^2}$; therefore in this area the additional clause helps us and the overall forcing probability increases. For a critical (≤ 2)-clause the argument is analogous. Here, the probability that x is forced given place r is at least r. The statement follows now by integration using the dominated convergence theorem, see appendix A.1. □

Corollary 3.4. *Let F be a (≤ 3)-CNF formula uniquely satisfied by α. If Δn variables of F have two critical clause, PPSZ finds α with probability at least $2^{-(S-0.0014\Delta+o(1))n}$.*

If Δn variables of F have a critical (≤ 2)-clause clause, PPSZ finds α with probability at least $2^{-(S-0.035\Delta+o(1))n}$.

If there are only few variables (less than $\Delta_1 n$) with one critical clause, we can find and guess them by brute-force. If we choose Δ_1 small enough, any exponential improvement for 1C-Unique 3-SAT gives a (diminished) exponential improvement to Unique 3-SAT. To bound the number of subsets of size $\Delta_1 n$, we define the binary entropy and use a well-known upper bound to the binomial coefficient.

Definition 3.5. *For $p \in [0,1]$, $H(p) := -p\log p - (1-p)\log(1-p)$ $(0\log 0 := 0)$.*

Lemma 3.6 (Chapter 10, Corollary 9 of [6]). *If pn is an integer, then $\binom{n}{pn} \leq 2^{H(p)n}$.*

We will manily prove that we have *some* exponential improvement. The claimed numbers are straightforward to check by inserting the values from the following table.

name	value	description
ε_1	10^{-19}	improvement in 1C-Unique 3-SAT
ε_2	10^{-24}	improvement in Unique 3-SAT
Δ_1	10^{-21}	threshold fraction of vars. with more than 1 crit. clause
Δ_2	$6 \cdot 10^{-5}$	$\Delta_2 n$ is the amount of variables for Δ_2-sparse and Δ_2-dense
ε_3	10^{-3}	exponential savings on repetitions if F is Δ_2-sparse
p^*	$8 \cdot 10^{-7}$	prob. that a var. is assigned using indep. 2-clauses instead of PPSZ

Lemma 3.7. *If there is a randomized algorithm $\text{ONECC}(F)$ solving 1C-Unique 3-SAT in time $2^{(S-\varepsilon_1+o(1))n}$ for $\varepsilon_1 > 0$, then there is a randomized algorithm (Algorithm 2) solving Unique 3-SAT in time $2^{(S-\varepsilon_2+o(1))n}$ for some $\varepsilon_2 > 0$.*

Proof. Let F be a (≤ 3)-CNF uniquely satisfied by α. Let $c(F)$ be the number of variables of F with more than one critical clause. If $c(F) \geq \Delta_1 n$, PPSZ is faster by Corollary 3.4. If $c(F) = 0$, we can use $\text{ONECC}(F)$.

However, what if $0 < c(F) < \Delta_1 n$? In that case, we get rid of these variables by brute-force: For all $\lfloor \Delta_1 n \rfloor$-subsets W of variables and for all $2^{\lfloor \Delta_1 n \rfloor}$ possible assignments α' on W, we try $\text{ONECC}(F^{[\alpha']})$. For

---
Algorithm 2 PPSZIMPROVED(CNF formula F)
---
repeat PPSZ(F) $2^{(S-\varepsilon_2)n}$ times, return if a satisfying assignment has been found
for all subsets W of size $\lfloor \Delta_1 n \rfloor$ and all assignments α' on W, try ONECC($F^{[\alpha']}$)

Algorithm 3 ONECC((≤ 3)-CNF F)
---
try DENSE(F)
try SPARSE(F)

---

one such α', we have $F^{[\alpha']}$ satisfiable and $c(F) = 0$; namely if W includes all variables with multiple critical clauses and α' is compatible with α. This is because fixing variables according to α does not produce new critical clauses w.r.t. α.

There are $\binom{n}{\lfloor \Delta_1 n \rfloor}$ subsets of size $\lfloor \Delta_1 n \rfloor$ of the variables of F, each with $2^{\lfloor \Delta_1 n \rfloor}$ possible assignments. As $\binom{n}{\lfloor \Delta_1 n \rfloor} \leq 2^{H(\Delta_1)n}$ (Lemma 3.6), we invoke ONECC($F^{[\alpha']}$) at most $2^{(\Delta_1 + H(\Delta_1))n}$ times. Setting Δ_1 small enough such that $\Delta_1 + H(\Delta_1) < \varepsilon_1$ retains an improvement for Unique 3-SAT. $\qquad\square$

4 Using One Critical Clause per Variable

In this section we give an exponential improvement for 1C-Unique 3-SAT.

Theorem 4.1. *Given a 1C-Unique (≤ 3)-CNF on n variables,* ONECC*(F) runs in expected time $2^{(S-\varepsilon_1+o(1))n}$ and finds the satisfying assignment with probability $2^{-o(n)}$.*

Obtaining a randomized algorithm using $2^{o(n)}$ independent repetitions and Markov's inequality is straightforward.

Corollary 4.2. *There exists a randomized algorithm for 1C-Unique 3-SAT running in time $2^{(S-\varepsilon_1+o(1))n}$.*

Together with Lemma 3.7 this immediately implies Theorem 1.4. We obtain the improvement by doing a case distinction into sparse and dense formulas, as defined now:

Definition 4.3. *For a CNF formula F and a variable x, the* degree *of x in F, $\deg(F,x)$ is defined to be the number of clauses in F that contain the variable x. The* 3-clause degree *of x in F, $\deg_3(F,x)$ is defined to be the number of 3-clauses in F that contain the variable x.*

For a set of variables W, denote by $F \setminus W$ the part of F independent *of W that consists of the clauses of F that do not contain variables of W.*

We say that F is Δ-sparse if there exists a set W of at most Δn variables such that $F \setminus W$ has maximum 3-clause degree 4. We say that F is Δ-dense otherwise.

We will show that for Δ_2 small enough, we get an improvement for Δ_2-sparse 1C-Unique (≤ 3)-CNF formulas. On the other hand, for any Δ_2 we will get an improvement for Δ_2-dense 1C-Unique (≤ 3)-CNF formulas. In the sparse case we can fix by brute force a small set of variables to obtain a formula with few 3-clauses. We need to deal with the (≤ 2)-clauses and then use an algorithm from Wahlström for CNF formulas with few clauses.

Algorithm 4 GETIND2CLAUSES$((\leq 3)$-CNF$F)$

{for the analysis, F is considered to be Δ_2-dense; the procedure might fail otherwise}
$F_3 \leftarrow \{C \in F \mid |C| = 3\}, F_2 \leftarrow \{\}$
for $\lceil \Delta_2 n \rceil$ times **do**
    let x be a variable with $\deg_3(F,x) \geq 5$ (return failure if no such variable exists)
    Choose C u.a.r. from all of F with $x \in \mathrm{vbl}(C)$.
    $l \leftarrow$ literal of C over x; $C_2 \leftarrow C \setminus l$
    $F_2 \leftarrow F_2 \cup C_2$
    {remove all clauses of F_3 sharing variables with C_2}
    $F_3 \leftarrow \{C_3 \in F_3 \mid \mathrm{vbl}(C') \cap \mathrm{vbl}(C_2) = \emptyset\}$
end for
return F_2

Algorithm 5 DENSE$((\leq 3)$-CNF $F)$

$F_2 \leftarrow$ GETIND2CLAUSES(F)
for $2^{(S-\varepsilon_1)n}$ times **do**
    $V_{p^*} \leftarrow$ pick each $x \in \mathrm{vbl}(F)$ with probability p^*
    $\alpha' \leftarrow \{\}$
    for $C_2 \in F_2$ **do**
        if $\mathrm{vbl}(C_2) \subseteq V_p$ **then**
            Let $\{u,v\} = C_2$
$$(\alpha'(u), \alpha'(v)) \leftarrow \begin{cases} (0,0), & \text{with probability } \frac{3}{15} \\ (0,1),(1,0),(1,1), & \text{with probability } \frac{4}{15} \text{ each} \end{cases}$$
        end if
    end for
    for all $x \in V_p$, if $\alpha'(x)$ is not defined yet let $\alpha'(x) \leftarrow_{\text{u.a.r.}} \{0,1\}$
    PPSZ$(F^{[\alpha']})$; if a satisfying assignment α has been found, return $\alpha \cup \alpha'$
end for

4.1 Dense Case

First we show the improvement for any Δ_2-dense 1C-Unique (≤ 3)-CNF. Δ_2-density means that even after ignoring all clauses over any $\Delta_2 n$ variables, a variable with 3-clause degree of at least 5 remains. The crucial idea is that for a variable x with 3-clause degree of at least 5, picking one occurence of x u.a.r. and removing it gives a 2-clause satisfied (by the unique satisfying assignment) with probability at least $\frac{4}{5}$. The only way a non-satisfied 2-clause can arise is if the 3-clause x was deleted from was critical for x. However we assumed that there is at most one critical clause for x.

Repeating such deletions and ignoring all 3-clauses sharing variables with the produced 2-clauses, as in listed in GETIND2CLAUSES(F), gives us the following:

Observation 4.4. *For a Δ_2-dense 1C-Unique (≤ 3)-CNF F, GETIND2CLAUSES(F) returns a set of $\lceil \frac{1}{2}\Delta_2 n \rceil$ independent 2-clauses, each satisfied (by the unique satisfying assignment of F) independently with probability $\frac{4}{5}$.*

As a random 2-clause is satisfied with probability $\frac{3}{4}$ by a specific assignment, this set of 2-clauses gives us nontrivial information about the unique satisfying asignment. Now we show how to use these 2-clauses to improve PPSZ:

Lemma 4.5. *Let F be a Δ_2-dense 1C-Unique (≤ 3)-CNF for some $\Delta_2 > 0$. Then there exists an algorithm ($\mathrm{DENSE}(F)$) runing in time $2^{(S - \varepsilon_1 + o(1))n}$ for $\varepsilon_1 > 0$ and returning the satisfying assignment α of F with probability $2^{-o(n)}$.*

Proof. First we give some intuition. For variables that occur late in PPSZ, the probability of being forced is large (being almost 1 in the second half). However for variables that come at the beginning, the probability is very small; a variable x at place p is forced (in the worst case) with probability $\Theta(p^2)$ for $p \to 0$, hence we expect $\Theta(p^3 n)$ forced variables among the first pn variables in total.

However, a 2-clause that is satisfied by α with probability $\frac{4}{5}$ can be used to guess both variables in a better way than uniform, giving constant savings in random bits required. For $\Theta(n)$ such 2-clauses, we expect $\Theta(p^2 n)$ of them to have both variables among the first pn variables. For each 2-clause we have some nontrivial information; intuitively we save around 0.01 bits. In total we save $\Theta(p^2 n)$ bits among the first pn variables, which is better than PPSZ for small enough p.

Formally, let V_{p^*} be a random set of variables, where each variable of V is added to V_{p^*} with probability p^*. On V_{p^*}, we replace PPSZ by our improved guessing; on the remaining variables $V \setminus V_{p^*}$ we run PPSZ as usual. Let $\mathrm{E_{guess}}$ be the event that the guessing on V_{p^*} (to be defined later) finds $\alpha|_{V_{p^*}}$. Let $\mathrm{E_{PPSZ}}$ be the event that $\mathrm{PPSZ}(F^{[\alpha|_{V_{p^*}}]})$ finds $\alpha|_{V \setminus V_{p^*}}$. Observe that for a fixed V_{p^*}, $\mathrm{E_{guess}}$ and $\mathrm{E_{PPSZ}}$ are independent. Hence we can write the overall probability to find α (call it p_s) as an expectation over V_{p^*}:

$$
\begin{aligned}
p_s &= E_{V_{p^*}}[\Pr(\mathrm{E_{guess}} \cap \mathrm{E_{PPSZ}}|V_{p^*})] \\
&= E_{V_{p^*}}[\Pr(\mathrm{E_{guess}}|V_{p^*})\Pr(\mathrm{E_{PPSZ}}|V_{p^*})] \\
&= E_{V_{p^*}}[2^{\log \Pr(\mathrm{E_{guess}}|V_{p^*}) + \log \Pr(\mathrm{E_{PPSZ}}|V_{p^*})}] \\
&\geq 2^{E_{V_{p^*}}[\log \Pr(\mathrm{E_{guess}}|V_{p^*}) + \log \Pr(\mathrm{E_{PPSZ}}|V_{p^*})]} \\
&= 2^{E_{V_{p^*}}[\log \Pr(\mathrm{E_{guess}}|V_{p^*})] + E_{V_{p^*}}[\log \Pr(\mathrm{E_{PPSZ}}|V_{p^*})]},
\end{aligned}
$$

where in the last two steps we used Jensen's inequality and linearity of expectation.

By Corollary 2.8, $E_{V_{p^*}}[\log \Pr(\mathrm{E_{PPSZ}})] = (-S_p + o(1))n$. We now define the guessing and analyze $E_{V_{p^*}}[\log \Pr(\mathrm{E_{guess}})]$ (see Algorithm 5 as a reference): By Observation 4.4 we obtain a set of $\lceil \frac{1}{2}\Delta_2 n \rceil$ independent 2-clauses F_2, each satisfied (by α) independently with probability $\frac{4}{5}$. In the following we assume that F_2 has at least a $\frac{4}{5}$-fraction of satisfied 2-clauses as this happens with constant probability (for a proof, see e.g. [2]) and we only need to show subexponential success probability.

Using the set of 2-clauses F_2, we choose an assignment α' on V_{p^*} as follows: For every clause C_2 in F_2 completely over V_{p^*} choose an assignment on both of its variables: with probability $\frac{1}{5}$ such that C_2 is violated, and with probability $\frac{4}{15}$ each on of the three assignments that satisfy C_2. Afterwards, guess any remaining variable of V_{p^*} u.a.r. from $\{0, 1\}$.

Given V_{p^*}, let m_0 be the number of clauses of F_2 completely over V_{p^*} *not satisfied* by α. Let m_1 be the number of clauses of F_2 completely over V_{p^*} *satisfied* by α. Then

$$
\Pr(\mathrm{E_{guess}}|V_{p^*}) = \left(\frac{1}{2}\right)^{|V_{p^*}| - 2m_0 - 2m_1} \left(\frac{1}{5}\right)^{m_0} \left(\frac{4}{15}\right)^{m_1}.
$$

This is seen as follows: For any variable for which no clause in C_2 is completely over V_{p^*}, we guess uniformly at random and so correctly with probability $\frac{1}{2}$. For any clause C_2 which is completely over V_{p^*}, we violate the clause with probability $\frac{1}{5}$, and choose a non-violating assignment with probability $\frac{4}{5}$. For any clause not satisfied by α, we hence set both variables according to α with probability $\frac{1}{5}$. For any clause satisfied by α, we set both variables according to α with probability $\frac{4}{15}$, as we have to pick the right one of the

8

Algorithm 6 SPARSE$((\leq 3)$-CNF F)

---

repeat the following $2^{(S-\varepsilon_3)n}$ times:
  for all subsets W of size $\lfloor \Delta_2 n \rfloor$ and all assignments α' on W **do**
    $F' \leftarrow F^{[\alpha']}$
    while no satisfying assignment found **do**
      try PPSZ$(F^{[\alpha']})$
      $F_2' \leftarrow \{C \in F' \mid |C| \leq 2\}$
      if $|F_2'| \leq \frac{1}{10}|\mathrm{vbl}(F')|$ **then**
        with probability $2^{-(S-0.015)|\mathrm{vbl}(F')|}$, run WAHLSTROEM$(F')$
      end if
      {set all literals in a uniform (≤ 2)-clause to 1}
      $C' \leftarrow_{\text{u.a.r.}} F_2'$, if $F_2' = \{\}$ return failure
      for $l \in C'$ **do**
        $F' \leftarrow F'^{[l \mapsto 1]}$
      end for
    end while
  end for

---

three assignments that satisfy C_2. As $E[V_{p^*}] = p^* n$, $E[m_0] \leq \frac{1}{5}p^{*2}\lceil \Delta_2 n \rceil$, $E[m_1] \geq \frac{4}{5}p^{*2}\lceil \Delta_2 n \rceil$, $E[m_0 + m_1] = p^{*2}\lceil \Delta_2 n \rceil$, we have

$$E[\log \Pr(\mathrm{E}_{\text{guess}}|V_{p^*})] = -E[V_{p^*} - 2m_0 - 2m_1] + \log\left(\frac{1}{5}\right)E[m_0] + \log\left(\frac{4}{15}\right)E[m_1]$$

$$\geq -p^* n + p^{*2}\lceil \Delta_2 n \rceil \left(2 + \log\left(\frac{1}{5}\right)\frac{1}{5} + \log\left(\frac{4}{15}\right)\frac{4}{5}\right).$$

The inequality follows from the observations and $\log\left(\frac{4}{15}\right) \geq \log\left(\frac{1}{5}\right)$. One can calculate $2 + \log\left(\frac{1}{5}\right)\frac{1}{5} + \log\left(\frac{4}{15}\right)\frac{4}{5} \geq 0.01$, corresponding to the fact that a four-valued random variable where one value occurs with probability at most $\frac{1}{5}$ has entropy at most 1.99.

Hence by our calculations and Observation 2.7 (to evaluate S_p), we have

$$\frac{1}{n}\log p_s \geq -S + p^* - \int_0^{p^*} \frac{r^2}{(1-r)^2}dr + o(1) - p^* + \Delta_2 p^{*2} \cdot 0.01$$

$$= -S - \int_0^{p^*} \frac{r^2}{(1-r)^2}dr + \Delta_2 p^{*2} \cdot 0.01 + o(1).$$

This gives an improvement over PPSZ of $-\int_0^{p^*} \frac{r^2}{(1-r)^2}dr + \Delta_2 p^{*2} \cdot 0.01 + o(1)$. The first term corresponds to the savings PPSZ would have, the second term corresponds to the savings we have in our modified guessing. Observe that for small p^*, the integral evaluates to $\Theta(p^{*3})$, but the second term is $\Theta(p^{*2})$. Hence choosing p^* small enough gives an improvement. $\qquad\square$

4.2 Sparse Case

Now we show that if $\Delta_2 > 0$ is small enough we get an improvement for a Δ_2-sparse 1C-Unique (≤ 3)-CNF. For this, we need the following theorem by Wahlström:

Theorem 4.6 ([9]). *Let F be a CNF formula with average degree at most* 4.2 *where we count degree* 1 *as* 2 *instead. Then satisfiability of F can be decided in time $O(2^{0.371n}) \leq 2^{(S-0.015+o(1))n}$. Denote this algorithm by* WAHLSTROEM(F).

Lemma 4.7. *Let F be a Δ_2-sparse 1C-Unique (≤ 3)-CNF. For Δ_2 small enough, there exists an algorithm (SPARSE(F)) running in expected time $2^{(S-\varepsilon_3+o(1))n}$ for $\varepsilon_3 > 0$ and finding the satisfying assignment α of F with probability $2^{-o(n)}$.*

Proof. Similar to Section 3, we first check by brute-force all subsets W of $\lfloor \Delta_2 n \rfloor$ variables and all possible assignments α' of W; by definition of Δ_2-sparse for some W, the part of F independent of W (i.e. $F \setminus W$) has maximum 3-clause degree 4. If furthermore α' is compatible with α, $F' := F^{[\alpha']}$ is a 1C-Unique (≤ 3)-CNF with maximum 3-clause degree 4: We observed that critical clauses cannot appear in the process of assigning variables according to α; furthermore any clause of F not independent of W must either disappear in F' or become a (≤ 2)-clause. As earlier, there are at most $2^{(\Delta_2+H(\Delta_2))n}$ cases of choosing W and α'. We now analyze what happens for the correct choice of F':

We would like to use WAHLSTROEM on F'; however F' might contain an arbitrary amount of (≤ 2)-clauses. The plan is to use the fact that either there are many critical (≤ 2)-clauses, in which case PPSZ is better, or few critical (≤ 2)-clauses, in which case all other (≤ 2)-clauses are non-critical and have only satisfied literals.

The algorithm works as follows: We have a 1C-Unique (≤ 3)-CNF on F' on $n' := |\text{vbl}(F')|$ variables; the maximum degree in the 3-clauses is at most 4. First we try PPSZ: if there are $\frac{1}{30}n'$ critical (≤ 2)-clauses, this gives a satisfying assignment with probability $2^{(S-0.035\frac{1}{30})n'}$. Otherwise, if there are less than $\frac{1}{10}n'$ (≤ 2)-clauses, the criterion of Theorem 4.6 applies: We invoke WAHLSTROEM(F') with probability $2^{-(S-0.015)n'}$; this runs in expected time $2^{-o(n)}$ and finds a satisfying assignment with probability $2^{-(S-0.015)n'}$.

If both approaches fail, we know that F' has at less than $\frac{1}{30}n'$ critical (≤ 2)-clauses clauses, but also more than $\frac{1}{10}n'$ (≤ 2)-clauses overall. Hence at most one third of the (≤ 2)-clauses is critical. However a non-critical (≤ 2)-clause must be a 2-clause with both literals satisfied. Hence choosing a (≤ 2)-clause of F' uniformly at random and setting all its literals to 1 sets two variables correctly with probability at least $\frac{2}{3} > 2^{-0.371 \cdot 2} > 2^{-(S-0.015) \cdot 2}$. That is we reduce the number of variables by 2 with a better probability than PPSZ overall; and we can repeat the process with the reduced formula. This shows that for the correct F', we have expected running time $2^{o(n)}$ and success probability $2^{(-S+\varepsilon_3-o(1))n}$ for $\varepsilon_3 > 0$. It is important to see that ε_3 does *not* depend on Δ_2. Repeating this process $2^{(-S+\varepsilon_3-o(1))n}$ times gives success probability $2^{o(n)}$.

Together with the brute-froce choice of W and α', we have expected running time of $2^{(S-\varepsilon_3+\Delta_2+H(\Delta_2)+o(1))n}$. By choosing Δ_2 small enough we are better than PPSZ. $\square$

5 Open Problems

Can we also obtain an improvement for general 3-SAT? In general 3-SAT, there might be even fewer critical clauses and critical clauses for some assignments are not always critical for others. We need to fit our improvement into the framework of [3]. As there is some leeway for multiple assignments, this seems possible, but nontrivial and likley to become very complex.

Another question is whether we can improve (Unique) k-SAT. PPSZ becomes slower as k increases, which makes an improvement easier. However the guessing in SPARSE relied on the fact that non-critical (≤ 2)-clauses have all literals satisfied, which is not true for larger clauses.

Suppose Wahlström's algorithm is improved so that it runs in time $O(c^n)$ on 3-CNF formulas with average degree D. The sparsification lemma [4] shows that for $c \to 1$ and $D \to \infty$, we obtain an algorithm for 3-SAT running in time $O(b^n)$ for $b \to 1$. Can our approach be extended to a similar sparsification result?

References

[1] R. G. Bartle. *The elements of integration and Lebesgue measure*, volume 92. Wiley-Interscience, 2011.

[2] K. Hamza. The smallest uniform upper bound on the distance between the mean and the median of the binomial and poisson distributions. *Statistics & Probability Letters*, 23(1):21 – 25, 1995.

[3] T. Hertli. 3-SAT faster and simpler—unique-SAT bounds for PPSZ hold in general. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011*, pages 277–284. IEEE Computer Soc., Los Alamitos, CA, 2011.

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

[5] K. Iwama and S. 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.

[6] F. F. J. MacWilliams and N. N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

[7] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364 (electronic), 2005.

[8] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 410–414. IEEE Computer Society, Los Alamitos, CA, 1999.

[9] M. Wahlström. An algorithm for the SAT problem for formulae of linear length. In *Algorithms—ESA 2005*, volume 3669 of *Lecture Notes in Comput. Sci.*, pages 107–118. Springer, Berlin, 2005.

[10] M. Wahlström. *Algorithms, measures and upper bounds for satisfiability and related problems*. PhD thesis, Department of Computer and Information Science, Linköpings Universitet Sweden, 2007.

[11] E. Welzl. Boolean Satisfiability – Combinatorics and Algorithms (Lecture Notes), 2005. `www.inf.ethz.ch/~emo/SmallPieces/SAT.ps`.

A Omitted Proofs

Theorem A.1. *Let F be a (≤ 3)-CNF on n variables with unique satisfying assignment α^*. Let $x \in \mathrm{vbl}(F)$ and $r \in [0,1]$. In the PPSZ algorithm, conditioned on $\beta = \alpha^*$ and $\pi(x) = r$, x is* forced *with probability at least $b(r)$, where $b(r) = b_n(r) = \min\left\{1, \frac{r^2}{(1-r)^2}\right\} - o(1)$ is a monotone increasing function ($o(1)$ stands for a term that goes to 0 when n goes to ∞).*

Proof of Theorem A.1. This can be derived from [7]. There are two differences: The first is that we define PPSZ with s-implication instead of bounded resolution. It is easily seen that the critical clause tree construction of [7] also works with s-implication. We use s-implication because we think it makes the algorithm easier to understand.

The second difference it that we give a bound for a fixed $\pi(x)$. We need this to be able to modify PPSZ and analyze it in special situations. However, we can derive this result from [7]:

Let $f(r) := \min\left\{1, \frac{r^2}{(1-r)^2}\right\}$, the "ideal" lower bound of PPSZ that a variable is forced. Remember that $\int_0^1 f(r) = 1 - S$. In [7] tehy give a lower bound $0 \leq b'(r) \leq 1$ on the probability that a variable is forced given $\pi(x)$ with $b'(r) \leq f(r)$. This bound is shown to integrate to $1 - S - o(1)$. As the probability that a variable is forced does not decrease if it comes later in PPSZ, the bound can easily been made monotone (if it is not already) by setting $b(r) := \max_{\rho \leq r} b'(r)$,

For $r = 0$ the statement is trivial. Now suppose for some $r \in (0,1]$, $b(r) < f(r) - \varepsilon$ for some $\varepsilon > 0$ and all n. By the above observation, $f(r) - b(r) \geq 0$, and $\int_0^1 (f(r) - b(r))dr = o(1)$. By continuity of $f(r)$ and monotonicity of $b(r)$, we find $r' < r$ such that $b(r) < f(r') - \varepsilon$ for all n. But then by monotonicity of $f(r)$ and $b(r)$, $\int_0^1 (f(t) - b(t))dt \geq \int_{r'}^r (f(t) - b(t))dr \geq \int_{r'}^r (f(r') - b(r))dr > (r' - r)\varepsilon$, a contradiction. $\square$

To go from Theorem A.1, where the place of a variable is fixed to the expectation, we need to integrate (this complicated approach gives us some flexibility later). We need the following special case of the well-known dominated convergence theorem (e.g. see [1]). It essentially states that the $o(1)$ integrates to an $o(1)$ in our case.

Theorem A.2 (Dominated Convergence Theorem). *Let $f : [a,b] \to \mathbb{R}$ be a continuous function with $\int_a^b f(x)dx = t$. Let $f_n(x) = f(x) - o(1)$ be integrable with $|f_n(x)| \leq 1$. Then $\int_a^b f_n(x)dx = t - o(1)$.*

Combining Theorem A.1 with Lemma 2.5 and the dominated convergence theorem A.2 gives us the following corollary. Integrability of f_n follows from monotonicity.

Corollary A.3. *Let F a (≤ 3)-CNF with unique satisfying assignment α. Then in PPSZ(F) conditioned on $\beta = \alpha$, the expected number of guessed variables is at most $(S + o(1))n$.*

Furthermore, suppose we pick every variable of F with probability p, independently, and let V_p be the resulting set. Then in PPSZ(F) conditioned on $\beta = \alpha$, the expected number of guessed variables is at most $(S_p + o(1))n$.

A.1 Integrals of Lemma 3.3

From Theorem A.1, we know that the probability that x is forced is at least $\min\left\{\frac{r^2}{(1-r)^2}, 1\right\} - o(1)$. Hence the overall probability that x is forced is at least

$$\int_0^1 \max\left\{2r^2 - r^3, \min\left\{\frac{r^2}{(1-r)^2}, 1\right\} - o(1)\right\} dr \geq 0.6152 - o(1).$$

The $o(1)$ integrates to $o(1)$ due to the dominated convergence theorem (Theorem A.2). If we have a critical (≤ 2)-clause, the overall probability that x is forced is at least

$$\int_0^1 \max\left\{r, \min\left\{\frac{r^2}{(1-r)^2}, 1\right\} - o(1)\right\} dr \geq 0.649 - o(1).$$