PPSZ for General k-SAT – Making Hertli's Analysis Simpler and 3-SAT Faster*

Dominik Scheder[1] and John P. Steinberger[2]

1 **Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai, China**
`dominik@cs.sjtu.edu.cn`

2 **Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China**
`jpsteinb@gmail.com`

Abstract

The currently fastest known algorithm for k-SAT is PPSZ named after its inventors Paturi, Pudlák, Saks, and Zane [7]. Analyzing its running time is much easier for input formulas with a unique satisfying assignment.

In this paper, we achieve three goals. First, we simplify Hertli's 2011 analysis [1] for input formulas with multiple satisfying assignments. Second, we show a "translation result": if you improve PPSZ for k-CNF formulas with a unique satisfying assignment, you will immediately get a (weaker) improvement for general k-CNF formulas.

Combining this with a result by Hertli from 2014 [2], in which he gives an algorithm for Unique-3-SAT slightly beating PPSZ, we obtain an algorithm beating PPSZ for general 3-SAT, thus obtaining the so far best known worst-case bounds for 3-SAT.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Boolean satisfiability, exponential algorithms, randomized algorithms

Digital Object Identifier 10.4230/LIPIcs.CCC.2017.9

1 Introduction

The problem of SAT, deciding whether a proposition formula conjunctive normal form has a satisfying assignment (or even constructing such a solution) enjoys a central position among NP-complete problems. The case of k-SAT, in which the input is restricted to k-CNF formulas, i.e., formulas of clause width bounded by k, has drawn special attention. An obvious brute-force algorithm solves SAT in time $O\left(2^n \text{poly}(n)\right)$, where n is the number of variables. For k-SAT, this running time has been improved quite a bit. Two approaches stand out: local search algorithms and encoding based algorithms. In 1999, Schöning [11] gave a simple local search algorithm for k-SAT. Paturi, Pudlák, and Zane [8] came up with an encoding-based algorithm, called PPZ in their honor. PPZ is not as good as Schöning, but has interesting applications in circuit complexity [8] and complexity of exponential algorithms [4].

Most importantly for this paper, there exists a "PPZ 2.0 version" called PPSZ (Paturi, Pudlák, Saks, and Zane [7]). This is the currently fastest randomized algorithm for k-SAT.

* Dominik Scheder gratefully acknowledges support by the National Natural Science Foundation of China under grant 61502300.

32nd Computational Complexity Conference (CCC 2017).
Editor: Ryan O'Donnell; Article No. 9; pp. 9:1–9:15
COMPUTATIONAL COMPLEXITY CONFERENCE
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

It is quite simple to state but challenging to analyze. We should state that its actual worst-case running time is not understood at all: Chen, Scheder, Talebanfard, Tang [10] construct exponentially hard instances, but their bounds are quite poor. Perhaps counterintuitively, the analysis in [7] incurs an exponential loss if the input formula has multiple solutions. Only in 2011, Timon Hertli [1] closed this gap in a breakthrough paper by a better (and simpler, yet still quite challenging) analysis. Still, PPSZ continued to be the best algorithm. A first crack in the wall appeared in 2014, when Hertli [2] combined PPSZ with several other algorithms, and showed that this improves the running time of Unique-3-SAT by a small but exponential amount. By *Unique-k-SAT* we mean k-SAT where the input formula F can have at most one satisfying assignment. If F may have multiple solutions, we write *general* k-SAT.

In this paper we first give a simpler analysis of Hertli's 2011 result [1]. This analysis also yields a translation result: if you improve PPSZ for Unique-k-SAT, you immediately get a (smaller) improvement for general k-SAT. Thus, researchers who want to "crack the PPSZ barrier" can focus on Unique-k-SAT for the time being. This, together with Hertli's 2014 improvement for Unique-3-SAT [2], gives the currently fastest known running time for general 3-SAT.

To give the reader an impression of which running time we are talking about, let us state some bounds for 3-SAT, ignoring subexponential factors. PPZ [8] runs in time $O\left(2^{2n/3}\right) \approx O\left(1.59^n\right)$, Schöning [11] in time $O\left(\left(\frac{4}{3}\right)^n\right) \approx O\left(1.334^n\right)$, and PPSZ [7] in time $O\left(2^{(2\ln(2)-1)n}\right) \approx O\left(1.308^n\right)$. The improvements by Hertli [2] and this paper are quite small (think of in the ballpark of tenth digit after the dot) and serve more as a demonstration that PPSZ *can* be improved, even if they do not improve it by much.

1.1 The PPSZ Algorithm

PPSZ is a probabilistic algorithm that tries to incrementally construct a satisfying assignment of F. The "generic PPSZ algorithm" is easy to state. Given a k-CNF formula F, choose a variable x therein uniformly at random; then choose a value $b \in \{0, 1\}$. Choose b uniformly at random, unless we can determine the "correct" truth value of x by some correct yet incomplete proof heuristic.

Let us state things more formally. A proof heuristic is a deterministic procedure P which on input F and x outputs a value $b \in \{0, 1, ?\}$. Correctness means that $P(F, x) = b \in \{0, 1\}$ means that $F \models (x = b)$, i.e., b is really the correct value of x; incompleteness means that we allow $P(F, x)$ to output "?", even if only one value $b \in \{0, 1\}$ for x is feasible. From now on, when we say *proof heuristic*, we always mean a correct but possibly incomplete heuristic.

Suppose now that $\alpha \in \mathrm{sat}(F)$, i.e., it is a satisfying assignment. Below we give procedure Encode that, given access to α, F, the heuristic P, and a permutation π of the variables of F, encodes α into a bit string c, hopefully using fewer than n bits. Intuitively, it iterates through the variables in the order given by π and outputs $\alpha(x)$ for every variable, unless this value is already implied by F and the bits output so far. This encoding is reversible: the procedure Decode can recover α when given access to F, P, π, and the encoding c. The generic algorithm RandomDecode then is simply to choose π and c randomly, start decoding and hoping for the best.

Note that the running time of RandomDecode is dominated by the running time of P. Thus, as long as P runs in polynomial (subexponential) time, so does RandomDecode. Consequently, we measure the goodness of RandomDecode not in terms of running time, but in terms of *success probability*, which will usually be of the form 2^{-pn} for some constant p. To make RandomDecode into an algorithm, we still have to specify P. Here are some examples:

Algorithm 1 Generic Encoding Procedure

procedure ENCODE(α, π, F, P)
  $\beta :=$ the empty assignment on V
  for $x \in V$ in the order of π **do**
    if $P(F|_\beta, x) = ?$ **then**
      output $\alpha(x)$
    end if
    add $[x \mapsto \alpha(x)]$ to β
  end for
end procedure

Algorithm 2 Generic Decoding Procedure

procedure DECODE(c, π, F, P)
  $\beta :=$ the empty assignment on V
  for $x \in V$ in the order of π **do**
    if $P(F|_\beta, x) = b \in \{0, 1\}$ **then**
      $\beta(x) := b$
    else
      $\beta(x) :=$ the next bit of c
    end if
  end for
  return β
end procedure

Algorithm 3 Generic Random Decoding Procedure

procedure RANDOMDECODE(F, P)
  $\pi :=$ a random permutation on V
  $c :=$ a random string in $\{0, 1\}^n$
  $\beta :=$ DECODE(c, π, F, P)
  return β if it satisfies F, else `failure`
end procedure

Example: P_0**.** This heuristic always outputs "?". Obviously, RANDOMDECODE(F, P_0) is just random guessing, and each solution α appears with probability 2^{-n}. This is not a very good algorithm.

Example: P_1**.** This heuristic answers $P_1(F, x) = b \in \{0, 1\}$ if F is a CNF formula and F contains the unit clause $(x = b)$[1] RANDOMDECODE(F, P_1) is the algorithm PPZ, invented by Paturi, Pudlák, and Zane [7]. Its success probability on k-CNF formulas is $2^{-(1-1/k)n}$.

Example: P_d**.** This heuristic generalizes P_1. It answers $P_d(F, x) = b$ if F is a CNF formula and it contains a subset G of at most d clauses for which $G \models (x = b)$. With this heuristic, RANDOMDECODE(F, P_d) becomes PPSZ, although Paturi, Pudlák, Saks, and Zane[7] state

[1] If F contains both $(x = 0)$ and $(x = 1)$ then $P_1(F, x)$ can be either 0 or 1, but in this case F is unsatisfiable anyway.

it slightly differently. Its success probability is much higher than that of PPZ (we will give more details below) but it is still not completely understood.

Example: P_∞. This heuristic employs the whole power of propositional logic. It answers $P_\infty(F, x) = b \in \{0, 1\}$ if F implies $(x = b)$. Obviously, determining this is itself NP-hard, so this is not an efficient heuristic. Still, it will be important in this paper. Note that for satisfiable F, RandomDecode(F, P_∞) always outputs a solution. Thus, it defines a distribution Q on pairs (π, α), where π is the permutation is chooses and α the solution it outputs. The distribution Q will be very important in our proofs below.

1.2 Gauging the Strength of the Proof Heuristic P

Towards an analysis of its success probability time, let $C_x(\alpha, \pi)$ be the indicator variable which is 1 if Encode outputs a bit for x., i.e., if $P(F|_\beta, x) = ?$ in Line 4 of Encode. So $C(\pi, \alpha) := \sum_x C_x(\pi, \alpha)$ is the length of the encoding, i.e., $|c| = C(\pi, \alpha)$. Note that $C_x(\pi, \alpha)$ also depends on F and P. Since they are usually fixed throughout, we choose to drop them for the sake of readability.

▶ **Observation 1.** $\Pr[\text{RandomDecode}(F, P) \text{ returns } \alpha] = \mathbb{E}_\pi\left[2^{-C(\pi,\alpha)}\right]$.

Proof. Let $c^* := \text{Encode}(\alpha, \pi, F, P)$. RandomDecode returns α iff the first $C(\pi, \alpha)$ bits of its random string $c \in \{0, 1\}^n$ agree with c^*. ◀

We write $F \models T$ as a shorthand of "F implies T", i.e., every satisfying assignment of F satisfies T. If $F \models (x = 0)$ or $F \models (x = 1)$ we say that x is *frozen in* F. Equivalently, all satisfying assignments of F agree on x. Otherwise, we say that x is *liquid*. Note that $C_x(\pi, \alpha)$ can be 1 for two reasons. First, it could be that in Line 4 of Encode, x is liquid in $F|_\beta$ and thus every correct proof heuristic P must answer $P(F|_\beta, x) = ?$. In this case we set $I_x(\pi, \alpha) = 1$. Second, it could be that x is frozen in $F|_\beta$ and therefore $P(F|_\beta, x) = ?$ is due to the incompleteness of P. In this case we set $J_x(\pi, \alpha) = 1$. Thus, $C_x(\pi, \alpha) = I_x(\pi, \alpha) + J_x(\pi, \alpha)$. We also set $I(\pi, \alpha) = \sum_x I_x(\pi, \alpha)$ and $J(\pi, \alpha) = \sum_x J(\pi, \alpha)_x$. Note that $I(\pi, \alpha) = 0$ if F has a unique satisfying assignment, since all variables are frozen. Also, $J(\pi, \alpha) = 0$ for P_∞, since this heuristic never fails. Here is a plausible notion of strength for proof heuristics: if P is a strong proof heuristic, then $J_x(\pi, \alpha) = 1$ should not happen too often:

▶ **Definition 2** (Error of P)**.** Let $\mathcal{C}$ be a class of formulas and P be a proof heuristic. P has *error at most p against $\mathcal{C}$* if $\mathbb{E}_\pi[J_x(\pi, \alpha)] \le p$ for every $F \in \mathcal{C}$, solution α, and variable x in F.

▶ **Theorem 3** ([8])**.** *P_1 has error $1 - 1/k$ against k-CNF formulas.*

Paturi, Pudlák, Saks, and Zane[7] prove the following bound on the error of P_d (although they do not use this exact wording). Consider the infinite $(k-1)$-ary rooted tree. For each vertex v in this tree, choose $\pi_v \in [0, 1]$ uniformly at random. Delete each vertex v with $\pi_v < \pi_{\text{root}}$. Let s_k be probability that the root is contained in an infinite connected component. It is easy to see that $s_2 = 0$. A simple calculation shows that $s_3 = 2\ln(2) - 1$.

▶ **Theorem 4** ([7])**.** *P_d has error $s_k + \epsilon_{d,k}$ against k-CNF formulas, where $\epsilon_{d,k} \to 0$ as $d \to \infty$.*

▶ **Observation 5.** *Let P be a proof heuristic of error at most p against $\mathcal{C}$. If $F \in \mathcal{C}$ has a unique satisfying assignment α, then* RandomDecode$(F, P) = \alpha$ *with probability at least* 2^{-pn}.

Proof. We use Observation 1 and Jensen's Inequality:

$$\begin{aligned}
\Pr[\text{ PPSZ succeeds}] &= \mathop{\mathbb{E}}_{\pi}\left[2^{-C(\pi,\alpha)}\right] \geq 2^{-\mathbb{E}_\pi[C(\pi,\alpha)]} && \text{(Jensen's Inequality)}\\
&= 2^{-\mathbb{E}_\pi[J(\pi,\alpha)]} && (I = 0 \text{ since only one assignment})\\
&\geq 2^{-pn} && (P \text{ has error at most } p)
\end{aligned}$$

◀

1.3 Previous Work

In case F has multiple satisfying assignments, the proof of Observation 5 breaks down, and it is not clear why a proof heuristic of error at most p should give an algorithm of success probability 2^{-pn}. A series of authors have improved PPSZ for the general case of multiple satisfying assignments. Paturi, Pudlák, Saks, and Zane [7] already gave an analysis, which has an exponential loss for $k = 3, 4$. Iwama and Tamaki [6] combine PPSZ for Schöning's random walk algorithm [11] to obtain a better algorithm. This combination was then further explored by Rolf [9], Iwama, Seto, Takai, and Tamaki [5], and Hertli, Moser, and Scheder [3]. All these improvements have serious drawbacks: they still have an exponential loss compared to the Unique-k-SAT bound for $k = 3, 4$; they are extremely technical; they use detailed knowledge of the proof heuristic P; finally, the latter four have to combine PPSZ with a second algorithm (Schöning's random walk algorithm [11]) to achieve their improvement. In 2011, Timon Hertli achieved a breakthrough by proving the following theorem:

▶ **Theorem 6** (Hertli [1]). *Suppose P has error at most p against $\mathcal{C}$, and $p \geq p^* := \frac{2-\log(e)}{2} \approx 0.279$. For every satisfiable $F \in \mathcal{C}$,* RandomDecode(F, P) *returns a satisfying assignment with probability at least 2^{-pn}.*

Note the mysterious p^* in the theorem. We suspect that it is an artefact of the proof and make the following conjecture:

▶ **Conjecture 7.** *Theorem 6 holds for all $p \geq 0$.*

Currently, the only supporting evidence for the conjecture is (1) our failure to construct a counterexample, despite some trying, and (2) that it would simply be very weird if it were false. Anyway, since $1 - s_k \geq p^*$ for all $k \geq 3$, Hertli's theorem works for the current version of PPSZ, for all $k \geq 3$. It might be, however, that future research brings about proof heuristics of error probability less than p^*, in which case the above theorem would again incur an exponential loss. Ingenious as it is, Hertli's proof is quite long and tedious.

1.4 Our Contribution

The first contribution of this paper is to give a much simpler proof of Theorem 6. Our proof in fact highlights why certain previous attempts fail, demonstrates more clearly "what is going on", and also points towards further improvements.

As a second contribution, we show that any improvement of PPSZ for Unique-k-SAT translates into a (weaker) improvement for General k-SAT. In particular, we will prove a stronger version of Theorem 6, which we now explain.

▶ **Definition 8.** A class $\mathcal{C}$ of formulas or circuits is *closed under restrictions* if $F \in \mathcal{C}$ implies that $F|_{x=b} \in \mathcal{C}$, for every variable x and value $b \in \{0, 1\}$.

Note that this applies to most "reasonable" circuit classes, in particular to k-CNF formulas.

▶ **Definition 9.** A proof heuristic P is called *monotone* if $P(F, x) \in \{0, 1\}$ implies that $P(F|_{y=b}, x) \in \{0, 1\}$, for every F, $y \neq x$, and $b \in \{0, 1\}$.

In other words, if P can deduce the value of x, then it can also do so after we add the additional information that $y = b$. Note that P_0, P_1, P_d, P_∞ define above are all monotone. Recall that RandomDecode(F, P_∞) chooses a uniformly random permutation $\pi \in \mathrm{Sym}(V)$ and always outputs a satisfying assignment. Thus, it defines a distribution Q on $\mathrm{Sym}(V) \times \mathrm{sat}(F)$ with $Q(\pi, \alpha) = \frac{1}{n!} \cdot 2^{-I(\pi,\alpha)}$.

▶ **Theorem 10.** *Suppose P has error at most p against $\mathcal{C}$, and set $q := p - p^*$ for $p^* := \frac{2-\log(e)}{2} \approx 0.279$. Let $F \in \mathcal{C}$ be satisfiable. Then* RandomDecode *returns a satisfying assignment with probability at least $2^{-pn+q\,\mathbb{E}_{(\pi,\alpha)\sim Q}[I(\pi,\alpha)]}$, where $q := p - p^*$.*

Since $s_k > p^*$ for all $k \geq 3$, the value q above is positive, which immediately reproves Hertli's Theorem (Theorem 6). As pointed out by one of the referees, the "bonus term" $\mathbb{E}_{(\pi,\alpha)\sim Q}[I(\pi, \alpha)]$ has an information-theoretic interpretation: it is the conditional entropy $H(\alpha|\pi)$. Our theorem has a nice by-product, a "translation result" from Unique-k-SAT to General k-SAT: suppose you have an algorithm A which is exponentially better than PPSZ for Unique-k-SAT. Given an input k-CNF formula F, there are two cases: first, it could be that $\mathbb{E}_Q[I]$ is "large" for this F, in which case Theorem 10 already gives an exponential bonus; or it is "small", in which case there is a small restriction ρ such that $F|_\rho$ has a unique satisfying assignment. We can now guess ρ and apply A to $F|_\rho$. Formally, we obtain the following theorem:

▶ **Theorem 11.** *Suppose P is a monotone proof heuristic with error probability at most p against class $\mathcal{C}$. We assume that $\mathcal{C}$ is closed under restrictions.*

1. *If* RandomDecode$(P, \cdot)$ *solves* Unique-$\mathcal{C}$-SAT *with probability at least $2^{(-p+\epsilon)n}$, then it solves* $\mathcal{C}$-SAT *with probability at least $2^{(-p+\epsilon')n}$.*
2. *If there is an algorithm A for* Unique-$\mathcal{C}$-SAT *with success probability $2^{(-p+\epsilon)n}$, then there is an algorithm A' for* $\mathcal{C}$-SAT *with success probability at least $2^{(-p+\epsilon')n}$ and running time n times that of A.*
3. *If there is Monte Carlo algorithm B solving* Unique-$\mathcal{C}$-SAT *running in time $2^{(p-\epsilon)n}$, then there exists a Monte Carlo algorithm B' solving* $\mathcal{C}$-SAT *in time $2^{(p-\epsilon')n}$.*

Here, $\epsilon' > 0$ if $\epsilon > 0$.

▶ **Theorem 12** (Hertli [2])**.** *There exists a Monte-Carlo algorithm solving Unique-3-SAT in time $O\left(2^{(s_3-\epsilon)n}\right)$ for some $\epsilon > 0$.*

Together with Theorem 11 we immediately obtain improvement for general 3-SAT and achieve the currently best running time.

▶ **Theorem 13.** *There is a Monte-Carlo algorithm solving 3-SAT in time $O\left(2^{(s_3-\epsilon')n}\right)$ for some $\epsilon' > 0$.*

2 Proof of Theorem 10

In addition to $Q(\pi,\alpha) = \frac{1}{n!} \cdot 2^{-I(\pi,\alpha)}$, we consider another distribution R on $\mathrm{Sym}(V) \times \mathrm{sat}(F)$. We estimate the success probability of RANDOMDECODE:

$$\begin{aligned}
\Pr[\text{success}] &= \sum_{\alpha \in \mathrm{sat}(F)} \mathop{\mathbb{E}}_{\pi}\left[2^{-C(\pi,\alpha)}\right] = \sum_{\alpha \in \mathrm{sat}(F)} \frac{1}{n!} \sum_{\pi} 2^{-I(\pi,\alpha) - J(\pi,\alpha)} \\
&= \sum_{\alpha \in \mathrm{sat}(F)} \sum_{\pi} R(\pi,\alpha) \frac{2^{-I(\pi,\alpha)-J(\pi,\alpha)}}{n! R(\pi,\alpha)} \\
&= \mathop{\mathbb{E}}_{(\pi,\alpha)\sim R}\left[\frac{Q(\pi,\alpha)}{R(\pi,\alpha)} \cdot 2^{-J(\pi,\alpha)}\right] \\
&\geq 2^{\mathbb{E}_R\left[-\log_2\left(\frac{R(\pi,\alpha)}{Q(\pi,\alpha)}\right) - J(\pi,\alpha)\right]} \qquad \text{(by Jensen's inequality)} \\
&= 2^{-D(R||Q) - \mathbb{E}_R[J(\pi,\alpha)]} ,
\end{aligned}$$

where $D(R||Q)$ is called the *Kullback-Leibler divergence from Q to R*. We can now plug in any distribution R and aim to minimize the expression

$$D(R||Q) + \mathop{\mathbb{E}}_{(\pi,\alpha)\sim R}[J(\pi,\alpha)] . \tag{1}$$

Here we face a tradeoff. If we choose R to be uniform over $\mathrm{Sym}(V) \times \mathrm{sat}(F)$, we get $\mathbb{E}_R[J(\pi,\alpha)] = \mathbb{E}_\alpha\left[\sum_x \mathbb{E}_\pi[J_x(\pi,\alpha)]\right] \leq pn$, since P has error at most p; however, $D(R||Q)$ might be too large. Choosing $R = Q$ makes $D(R||Q) = 0$, but the second term can become larger than pn. Informally speaking, the problem is that for certain F, P, and α, if we sample π from the conditional distribution $Q|\alpha$, frozen variables x tend to come earlier (compared to a uniformly sampled π). Thus, when we call $P(F|_\beta, x)$, we have less information (β tends to be a shorter partial assignment), and J_x is more likely to be 1. In Section B we provide examples where these phenomena actually happen.

The process SAMPLE-R below defines a distribution R on pairs (π,α) that resembles Q (and thus keeps the divergence $D(R||Q)$ small) while showing a moderate preference for moving frozen variables to the back of π (keeping $\mathbb{E}_R[J(\pi,\alpha)]$ small). Note that unlike under Q, the marginal distribution on permutations induced by R is not necessarily uniform. Indeed, if we call SAMPLE-R(F,V) for $F = x$ and $V = \{x,y\}$ then π is (y,x) with probability $2/3$ and (x,y) with probability $1/3$. On the other hand, R and Q induce the same marginal distribution on satisfying assignments. The reader is encouraged to verify this, but this property is not required for the proof. We call the resulting distribution R_F to highlight its dependency on F. If F is understood from the context, we simply write R.

▶ **Lemma 14.** *$D(R||Q) \leq p^* \, \mathbb{E}_R[I]$ for every F.*

This is where the mysterious $p^* = \left(\frac{2-\log(e)}{2}\right)$ comes from. The proof of Lemma 14 is a little bit technical but rather straightforward for somebody familiar with information theory, and can be found in the appendix.

▶ **Lemma 15.** *Let $\mathcal{C}$ be a formula class closed under restrictions, P a monotone proof heuristic with error at most p against $\mathcal{C}$. Then for every $F \in \mathcal{C}$ and every frozen variable x of F it holds that $\mathbb{E}_R[J_x] \leq p$.*

This lemma is in some way the heart of our proof. Its proof studies how the conditional distribution $R(\pi|\alpha)$ differs from the uniform distribution over π and applies two careful

Algorithm 4 Sampling from the distribution R

```
1: procedure SAMPLE-R(F, V)
2:     if V = ∅ then
3:         return (∅, ∅)
4:     end if
5:     S(F) := {(x, b) ∈ V × {0, 1} | F|_{x=b} is satisfiable }
6:     (x, b) := a random element from S
7:     (π, α) := SAMPLE-R(F|_{x=b}, V \ {x})
8:     return (xπ, α ∪ [x = b])
9: end procedure
```

coupling arguments. It is also the place where we use that P is monotone. Lemma 15 has the following consequence:

▶ **Lemma 16.** *$\mathbb{E}_R[pI_x + J_x] \leq p$ for every $x \in V$, and $\mathbb{E}_R[pI + J] \leq pn$.*

Proof. Imagine we run the process SAMPLE-R but pause when (1) x becomes frozen or (2) x, as a non-frozen variable, is chosen in line 6. Everything what happens before the pause is called *the past*. If (2) happens, then $I_x = 1, J_x = 0$ and thus $\mathbb{E}_R[pI_x + J_x|\text{the past}] = \mathbb{E}_R[p \cdot 1 + 0] = p$. Otherwise, if (1) happens, then $I = 0$ since x becomes frozen, and $\mathbb{E}_R[pI_x + J_x|\text{the past}] = \mathbb{E}_R[J_x|\text{the past}]$. After *the past* has happened, the sampling process has arrived at a new formula $F' \in \mathcal{C}$, and x is frozen in F'. Since $\mathcal{C}$ is closed under restrictions, $F' \in \mathcal{C}$, too, and we can apply Lemma 15 to conclude that $\mathbb{E}_{R_F}[J_x|\text{the past}] = \mathbb{E}_{R_{F'}}[J_x] \leq p$. Thus, $\mathbb{E}_R[pI_x + J_x] \leq p$. ◀

▶ **Lemma 17.** $\mathbb{E}_R[I] = \mathbb{E}_Q[I]$.

Let us put everything together. $D(R||Q) + \mathbb{E}_R[J] \leq p^* \mathbb{E}_R[I] + \mathbb{E}_R[J] = \mathbb{E}_R[pI + J] - (p - p^*)\,\mathbb{E}_R[I] \leq pn - q\,\mathbb{E}_Q[I]$. Thus, RANDOMDECODE succeeds with probability at least $2^{-pn+q\,\mathbb{E}_Q[I]}$. This proves Theorem 10.

3 Unique to General

We are now ready to prove Theorem 11, which claims that if you can beat PPSZ for UNIQUE-$\mathcal{C}$-SAT, then you can beat it for $\mathcal{C}$-SAT.

Proof of Theorem 11. Let $\delta > 0$ be a fixed number, to be determined later. If $\mathbb{E}_Q[I] \geq \delta \cdot n$, then

$$\Pr[\text{RANDOMDECODE}(F, P) \text{ successful}] \geq 2^{-pn+\delta cn} , \tag{2}$$

which is exponentially larger than 2^{-pn}.

Otherwise, assume that $\mathbb{E}_Q[I] \leq \delta n$. In particular, $I(\pi, \alpha) \leq \delta n$ for *some* permutation π and assignment α. This means that there is a partial assignment ρ fixing δn variables such that $F|\rho$ has a unique satisfying assignment. We prove Point 1 of the theorem. When running RANDOMDECODE on F, with probability $\binom{n}{\leq \delta n}^{-1} \cdot 2^{\delta n}$ the first δn steps produce exactly ρ, and the remaining $(1-\delta)n$ steps are like running RANDOMDECODE$(F|_\rho, P)$. $F|_\rho$, has the unique solution α, and thus RANDOMDECODE$(F|_\rho, P)$ finds α with probability at least $2^{(-p+\epsilon)(n-\delta)}$. Altogether,

$$\Pr[\text{RANDOMDECODE}(F, P) = \alpha] \geq \binom{n}{\delta n}^{-1} \cdot 2^{-\delta n} \cdot 2^{(-p+\epsilon)(n-\delta)} . \tag{3}$$

By choosing $\delta > 0$ optimally, we can make sure that both (2) and (3) are at least $2^{(-p+\epsilon')n}$, for some $\epsilon' > 0$. This proves Point 1 of the theorem. The proofs of the other two points are similar. ◀

4 Open Questions

Can we show that formulas with a unique solution are the worst case for RandomDecode under every "reasonable" heuristic P?

Can we show that the success probability of RandomDecode is exponentially larger than 2^{-pn} if F has an exponential number of solutions? Unfortunately, the current "bonus term" $\mathbb{E}_Q[I]$" can be *constant* for some formulas with a large number of solutions, for example for $F = (x_1 \wedge \cdots \wedge x_{n/2}) \vee (|\mathbf{x}| \leq 100)$ (note that $\mathbb{E}_Q[I]$ only depends on the underlying boolean function, not on its representation as a CNF formula).

Acknowledgements. Dominik Scheder want to thank Navid Talebanfard for inspiring discussions.

References

1. Timon 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. `doi:10.1109/FOCS.2011.22`.
2. Timon Hertli. Breaking the PPSZ Barrier for Unique 3-SAT. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 600–611. Springer, 2014. `doi:10.1007/978-3-662-43948-7_50`.
3. Timon Hertli, Robin A. Moser, and Dominik Scheder. Improving PPSZ for 3-SAT using critical variables. In *Proceedings of STACS*, pages 237–248, 2011. URL: `http://arxiv.org/abs/1009.4830`.
4. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. `doi:10.1006/jcss.2000.1727`.
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 Comput. Sci.*, 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. *J. ACM*, 52(3):337–364 (electronic), 2005. `doi:10.1145/1066100.1066101`.
8. Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theoret. Comput. Sci.*, pages Article 11, 19 pp. (electronic), 1999.
9. Daniel Rolf. Improved Bound for the PPSZ/Schöning-Algorithm for 3-SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:111–122, 2006.
10. Dominik Scheder, Bangsheng Tang, Shiteng Chen, and Navid Talebanfard. Exponential lower bounds for the PPSZ k-sat algorithm. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013,*

New Orleans, Louisiana, USA, January 6-8, 2013, pages 1253–1263. SIAM, 2013. doi:10.1137/1.9781611973105.91.

11 Uwe 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. doi:10.1109/SFFCS.1999.814612.

A Proof of the lemmas

For a formula F over variable set V, recall that $S(F,V)$ is the set of all pairs $(x,b) \in V \times \{0,1\}$ for which $F|_{x=b}$ is satisfiable. Note that if F is satisfiable then $|S(F,V)|$ is n plus the number of liquid variables.

▶ **Lemma 14** (restated). *$D(R||Q) \le \left(\frac{2-\log(e)}{2}\right) \mathbb{E}_R[I]$, for every formula F.*

Proof. Let us spell out a pair (π,α) as $(x_1 \dots x_n, b_1 \dots b_n)$, where x_i is the i^{th} variable under π and $b_i = \alpha(x_i)$. Let $\tau_i := (x_1 \dots x_i, b_1 \dots b_i)$ be a "prefix" of (π,α). Define R_{τ_i} be the distribution of (b_{i+1}, x_{i+1}) under R conditioned on τ_i. Similarly define Q_{τ_i}. By the chain rule for the divergence we get

$$D(R||Q) = \sum_{i=0}^{n-1} \mathop{\mathbb{E}}_{\tau_i \sim R} [D(R_{\tau_i}||Q_{\tau_i}) \ .$$

So let us fix a "past" τ_i and bound $D(R_{\tau_i}||Q_{\tau_i})$. Let $F_i := F|_{x_1 \mapsto b_1 \dots x_i \mapsto b_i}$ and $V_i := \{x_{i+1}, \dots, x_n\}$. So F_i is a CNF formula over V_i, and it is exactly the formula for which Sample-R is called in its i^{th} call. Let $n_i = |V_i|$, $s_i := |S(F_i, V_i)|$, f_i the number of frozen variables in V_i and l_i the number of liquid variables. Thus $f_i + l_i = n_i$ and $f_i + 2l_i = s_i$. Note that R_i is uniform over $S(F_i, V_i)$. Q_{τ_i} picks x_{i+1} uniformly at random from V_i and assigns it a random value from the (one or two) allowed values. Thus, $Q_{\tau_i}(x,b)$ is 0 if $(x,b) \notin S(F_i, V_i)$; otherwise, it is $1/n_i$ if x is frozen and $1/2n_i$ if x is liquid.

$$\begin{aligned}
D(R_{\tau_i}||Q_{\tau_i}) &= \sum_{(x,b)\in S(F_i,V_i)} R_{\tau_i}(x,b) \log\left(\frac{R_{\tau_i}(x,b)}{Q_{\tau_i}(x,b}\right) \\
&= \sum_{(x,b)\in S(F_i,V_i)} \frac{1}{s_i} \log\left(\frac{1/s_i}{[1/n_i \text{ if } x \text{ frozen, } 1/2n_i \text{ if } x \text{ liquid }]}\right) \\
&= \frac{2l_i}{s_i} \log\left(\frac{1/s_i}{1/2n_i}\right) + \frac{f_i}{s_i} \log\left(\frac{1/s_i}{1/n_i}\right) = \frac{2l_i}{s} \log\left(\frac{2n_i}{s_i}\right) + \frac{f_i}{s_i} \log\left(\frac{n_i}{s_i}\right) \\
&= \frac{2l_i}{s} + \log\left(\frac{n_i}{s_i}\right) = \frac{2l_i}{s} + \log\left(1 - \frac{l_i}{s_i}\right) \\
&\le \frac{2l_i}{s} - \log(e)\frac{l_i}{s_i} = \frac{l_i}{s_i}(2 - \log(e)) \ .
\end{aligned}$$

Let $\tilde{I}_i(\pi,\alpha) := I_{x_i}(\pi,\alpha)$, i.e., an indicator variable which is 1 if the i^{th} variable under π is liquid in F_{i-1}. We observe that $\mathbb{E}_{R_{\tau_i}}[\tilde{I}_{i+1}] = \frac{2l_i}{s_i}$, since there are exactly $2l_i$ pairs $(x,b) \in S(F_i, V_i)$ for which the variable x is liquid. Putting everything together, we get

$$D(R||Q) \le \sum_{i=0}^{n-1} \mathop{\mathbb{E}}_{\tau_i \sim R}\left[\frac{l_i}{s_i}(2 - \log(e))\right] = \frac{2-\log(e)}{2} \sum_{i=0}^{n-1} \mathop{\mathbb{E}}_{\tau_i \sim R}\left[\frac{2l_i}{s_i}\right] \ .$$

As we have just seen, the latter sum equals $\mathbb{E}_R\left[\sum_{i=1}^n \tilde{I}_i\right]$, which again equals $\mathbb{E}_R[I]$, since $\tilde{I}_i$, $i = 1, \dots, n$ simply counts I_x, $x \in V$ in a different order. ◀

A.1 Permutations that delay x – Proof of Lemma 15

Before we prove Lemma 15, we have to introduce some notation. We call a function $g : 2^V \to \mathbf{R}$ *monotone* if $g(A) \leq g(B)$ for any $A \subseteq B \subseteq V$. Let $x \in V$ be a fixed variable, $\pi \in \mathrm{Sym}(V)$ a permutation. We denote by $W(\pi)$ the set of variables appearing after x in π. Observe that $J_x(\pi, \alpha)$ only depends on $W(\pi)$, not on the particular order of the variables coming before x and of those coming after x.

▶ **Observation 18.** *J_x is a monotone function in W, since P is a monotone heuristic.*

For two strings σ, π, we write $\sigma \preceq \pi$ if σ is a prefix of π. A permutation π on set V of size n can be viewed as a string in V^n without repeated letters. A string $\sigma \in V^*$ without repeated letters is called a *partial permutation*. If D is a distribution over permutations on V and σ is a partial permutation, we write $D(\sigma) := \Pr_{\pi \sim D}(\sigma \preceq \pi)$.

▶ **Definition 19.** Let D be a distribution over permutations on V, and let $x \in V$. We say D *delays* x if for all $y \in V$ and all partial permutations σ not containing x or y, it holds that $D(\sigma x) \leq D(\sigma y)$.

Informally, at every stage, x is among the least likely elements to come next. For example, the uniform distribution delays x; so does the distribution that samples a permutation of $V \setminus \{x\}$ and places x at the end. Lemma 15 will follow from the next two lemmas:

▶ **Lemma 20.** *The distribution $(R|\alpha)$ delays x, for every frozen variable x.*

Here, $(R|\alpha)$ is the distribution on permutations conditioned on this fixed satisfying assignment α, i.e., $(R|\alpha)(\pi) = R(\pi, \alpha|\alpha)$.

▶ **Lemma 21.** *Let V be a finite set, $x \in V$, D a distribution over permutations of V that delays x, and $f : V \to \mathbb{R}$ a monotone function. Denote by $W = W(\pi)$ the set of elements coming after x in π. Then*

$$\mathop{\mathbb{E}}_{\pi \sim D}[f(W)] \leq \mathop{\mathbb{E}}_{\pi \sim \mathcal{U}}[f(W)] ,$$

where $\mathcal{U}$ is the uniform distribution over permutations.

Proof Idea. Since D delays x, the set W tends to be smaller under D than under $\mathcal{U}$. Since f is monotone this means the expectation $f(W)$ is smaller, too. This is the intuition. The formal proof uses a coupling argument. ◀

▶ **Lemma 15** (restated)**.** *Let $\mathcal{C}$ be a formula class closed under restrictions, P a monotone proof heuristic with error at most p against $\mathcal{C}$. Then for every $F \in \mathcal{C}$ and every frozen variable x of F it holds that $\mathbb{E}_R[J_x] \leq p$.*

Proof. By assumption on P we have $\mathbb{E}_\pi[J_x(\pi, \alpha)] \leq p$ when π is uniform. Thus, we have to compare how the uniform distribution and $(R|\alpha)$ differ in their treatment of x, and how $J_x(\pi, \alpha)$ reacts to these differences. By Lemma 20, $(R|\alpha)$ delays x. By Observation 18, J is a monotone function in W, where $W = W(\pi)$ is the set of elements coming after x in π. Thus, by Lemma 21 we obtain that $\mathbb{E}_{\pi \sim R}[J_x(\pi, \alpha)] \leq \mathbb{E}_{\pi \sim \mathcal{U}}[J_x(\pi, \alpha)] \leq p$. ◀

A.2 Remaining proofs – Lemma 20 and Lemma 21

Proof of Lemma 20. By assumption, x is frozen and σ is a partial permutation not containing x nor y. Assume first that σ is empty. We have to show that $R(x|\alpha) \leq R(y|\alpha)$ or, equivalently, $R(x, \alpha) \leq R(y, \alpha)$.[2]

Consider the following alternative but equivalent way to sample R: order the s elements of $S(F, V)$ randomly into a sequence $\tau = (x_1, b_1), \ldots, (x_s, b_s)$ and then add the unit clauses $(x_i = b_i)$ to F, in this order, skipping a unit clause if adding it would make F unsatisfiable. This adds n unit clauses in some order $(x_{i_1} = b_{i_1}), \ldots, (x_{i_n} = b_{i_n})$ and thus defines a permutation π of V and an assignment α. The pair (π, α) has distribution R.

Let $T_{z,\alpha}$ denote the set of all such sequences τ that (1) result in α and (2) place z at the beginning of π. So $R(z, \alpha) = \frac{|T_{z,\alpha}|}{|S(F,V)|!}$. Since the first unit clause $(x_1 = b_1)$ in a sequence is always consistent with F, every sequence in $T_{z,\alpha}$ starts with $(z = \alpha(z))$. For a sequence $\tau \in T_{x,\alpha}$ define $f(\tau)$ to be the sequence τ' where we switch the positions of $(x = \alpha(x))$ and $(y = \alpha(y))$ (note that both must appear in τ, and $(x = \alpha(x))$ appears at the beginning). A minute of thought shows that the sequence $f(\tau)$ leads to α as well (the key observation is that x is frozen, so logically $(x = \alpha(x))$ is already present in F, whether it occurs at the beginning of τ or not). Thus $f(\tau) \in T_{y,\alpha}$ and we have just defined an injection from $T_{x,\alpha}$ into $T_{y,\alpha}$. This shows that $|T_{x,\alpha}| \leq |T_{y,\alpha}|$ and thus $R(x, \alpha) \leq R(y, \alpha)$.

If σ is not empty we write $\alpha = \alpha_\sigma \alpha_{\bar{\sigma}}$, where α_σ is the α restricted to the variables appearing in σ, and $\alpha_{\bar{\sigma}}$ is the rest. Write $F' := F|_{\alpha_\sigma}$ Now $R(\sigma z, \alpha)$ is the probability that Sample-R follows σ and α in its first $|\sigma|$ steps, times $R_{F'}(z, \alpha_{\bar{\sigma}})$. Thus, we have reduced non-empty σ case to the empty σ case. ◀

▶ **Lemma 21** (restated). *Let V be a finite set, $x \in V$, D be a distribution over permutations of V that delays x, and $f : V \to \mathbb{R}$ be a monotone function. Denote by $W = W(\pi)$ the set of elements coming after x in π. Then*

$$\mathop{\mathbb{E}}_{\pi \sim D}[f(W)] \leq \mathop{\mathbb{E}}_{\pi \sim \mathcal{U}}[f(W)] \ ,$$

where $\mathcal{U}$ is the uniform distribution over permutations.

Proof. Let W_D denote a random variable distributed like $W(\pi)$ with $\pi \sim D$, and similarly $W_{\mathcal{U}} = W(\pi)$ where π is uniform. Below, we define a process Sample-W which simultaneously samples W_D and $W_{\mathcal{U}}$ and guarantees $W_D \subseteq W_{\mathcal{U}}$. In other words, Sample-W defines a coupling under which $W_D \subseteq W_{\mathcal{U}}$ We write $D(z|\sigma) := D(\sigma z|\sigma) = \frac{D(\sigma z)}{D(\sigma)}$. This is the probability that z is chosen next, conditioned on σ having been sampled so far.

The process Sample-W clearly samples W_D from the correct distribution. Note that an element z gets removed from $W_{\mathcal{U}}$ whenever $t < D(x|\sigma)$, and then a uniformly random element is removed. Also, the process terminates when x has been removed from W_D. Obviously, it will be removed from $W_{\mathcal{U}}$ in the same iteration. So W_D and $W_{\mathcal{U}}$ have the correct distribution. Lastly, since $D(x|\sigma) \leq D(z|\sigma)$, when the element z is removed from $W_{\mathcal{U}}$, it has already been removed from W_D. Thus, $W_D \subseteq W_{\mathcal{U}}$ holds in every step. Thus, $f(W_D) \leq f(W_{\mathcal{U}})$ with probability 1 and therefore $\mathbb{E}_{\pi \sim D}[f(W)] \leq \mathbb{E}_{\pi \sim \mathcal{U}}[f(W)]$. ◀

[2] We have not formally introduced this notation. It is the probability that Sample-R outputs α and a permutation π starting with x (respectively, y)

Algorithm 5 Sampling W_D and $W_{\mathcal{U}}$

procedure SAMPLE-W(V, x)
  $\sigma :=$ the empty string
  $W_D = W_{\mathcal{U}} = V$
  while $x \in W_D$ **do**
    $(z, t) \in V \times [0, 1]$, uniformly at random
    if $t < D(z|\sigma)$ and $z \in W_D$ **then**
      remove z from W_D
      append z to σ
    end if
    if $t < D(x|\sigma)$ **then**
      remove z from $W_{\mathcal{U}}$
    end if
  end while
  return $W_D, W_{\mathcal{U}}$
end procedure

B Bad Examples

B.1 Why s Direct Application of Jensen's Does Not Work

We will demonstrate why proving Theorem 6 requires nontrivial effort. Let us proceed as in the proof of Observation 5. Let sat(F) be the set of all satisfying assignments of F. The success probability of Decode is

$$\begin{aligned}\Pr_{c,\pi}[\text{success}] &= \sum_{\alpha\in\text{sat}(F)} \Pr_{c,\pi}[\textsc{Decode}(c,\pi,F,P)=\alpha] \\ &= \sum_{\alpha\in\text{sat}(F)} \mathop{\mathbb{E}}_{\pi}\left[2^{-C(\pi,\alpha)}\right] \qquad (4) \\ &\geq \sum_{\alpha\in\text{sat}(F)} 2^{-\mathbb{E}_\pi[C(\pi,\alpha)]}\ , \qquad (5)\end{aligned}$$

where last line follows from Jensen's inequality.

We will construct an example in which Pr[success] $= 1$ but (5) is exponentially small. Consider $P = P_\infty$, the complete proof heuristic, which has error 0 against, well, every circuit class. Also note that (4) is 1, as Decode always returns a satisfying assignment if given access to P_∞. Let F be the Boolean function defined by $F(x) = 1$ if $|x| = 1$, i.e., exactly one of the n positions of x is 1. So sat$(F) = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$. Note that since P_∞ is the complete prover, it does not really matter in which way we represent F.

By symmetry, $\Pr[\textsc{Decode}(c, \pi, F) = \mathbf{e}_i] = 1/n$ for every i. What is $C(\mathbf{e}_i, \pi)$? Let j be the position of x_i in π. A minute of thought shows that $C(\mathbf{e}_i, \pi) = \min(j, n-1)$. Therefore

$$\mathop{\mathbb{E}}_{\pi}[C(\mathbf{e}_i,\pi)] = \frac{1}{n}\cdot\sum_{j=1}^{n-1} j + \frac{1}{n}(n-1) \geq \frac{\binom{n}{2}}{n} = \frac{n-1}{2}\ .$$

Summing up over all sat(F) we see that

$$(5) = \sum_{\alpha\in\text{sat}(F)} 2^{-\mathbb{E}_\pi[C(\pi,\alpha)]} \leq n\cdot 2^{-\frac{n-1}{2}}\ .$$

Thus, there is an exponential gap between (5) and $2^{-pn} = 2^{-0 \cdot n} = 1$, the bound in the conjecture. We conclude that this "naive" application of Jensen's inequality will not work.

B.2 A Smarter Application of Jensen's Inequality

Suppose we run $\textsc{Decode}(c, \pi, F)$ with random c and π and the complete prover P_∞. It will always return a satisfying assignment, and thus defines a probability distribution Q over $\mathrm{Sym}(V) \times \mathrm{sat}(F)$. It is easy to see that

$$Q(\pi, \alpha) = Q(\pi) \cdot Q(\alpha|\pi) = \frac{1}{n!} \cdot 2^{-I(\pi,\alpha)} \ .$$

We can now rewrite the success probability of Decode (using some incomplete proof heuristic P) as

$$\begin{aligned} \Pr[\text{success}] &= \sum_{\alpha \in \mathrm{sat}(F)} \mathop{\mathbb{E}}_{\pi} \left[2^{-C(\pi,\alpha)} \right] = \sum_{\pi,\alpha} \frac{1}{n!} 2^{-I(\pi,\alpha) - J(\pi,\alpha)} \\ &= \mathop{\mathbb{E}}_{(\pi,\alpha)\sim Q} \left[2^{-J} \right] & (6) \\ &\geq 2^{-\mathbb{E}_Q[J]} \ . & (7) \end{aligned}$$

Sadly, (7) can be exponentially smaller than 2^{-pn}, as we will show now.

B.3 Another Bad Example

Consider the following function:

$$\textsc{Exactly-Two}(x, y, z) \wedge \bigwedge_{i=1}^{n} (\textsc{At-Least-Two}(x, y, z) \rightarrow a_i) \ .$$

We can express this as a 3-CNF formula:

$$\begin{aligned} &(x \vee y) \wedge (x \vee z) \wedge (y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}) \wedge \\ \bigwedge_{i=1}^{n} &((\bar{x} \vee \bar{y} \vee a_i) \wedge (\bar{x} \vee \bar{z} \vee a_i) \wedge (\bar{y} \vee \bar{z} \vee a_i)) \ . \end{aligned}$$

Enumerating our variables as $x, y, z, a_1, \ldots, a_n$, the satisfying assignments are $\alpha_1 = (0111^n)$, $\alpha_2 = (1011^n)$, and $\alpha_3 = (1101^n)$. Consider the prover $P = P_1$, i.e., it checks whether the variable in question is contained in a unit clause. Since this is a 3-CNF, the error probability of P is at most $2/3$. What is $\mathbb{E}_Q[J]$?

$$\begin{aligned} \mathop{\mathbb{E}}_{Q}[J] &= \mathop{\mathbb{E}}_{\alpha\sim Q}[\mathop{\mathbb{E}}_{\pi\sim Q|\alpha}[J]] = \mathop{\mathbb{E}}_{\pi\sim Q|\alpha_1}[J(\alpha_1, \pi)] & \text{(by symmetry between the } \alpha_i) \\ &\geq n \mathop{\mathbb{E}}_{\pi\sim Q|\alpha_1}[J_{a_1}(\alpha_1, \pi)] \ . & \text{(by symmetry between the } a_i) \end{aligned}$$

One can now show by a straightforward calculation that $\mathbb{E}_{\pi\sim Q|\alpha_1}[J_{a_1}(\alpha_1, \pi)] = \frac{11}{16} > 2/3$. Thus, the expression $2^{-E_Q[J]}$ can be exponentially smaller than $2^{-\frac{2}{3}\cdot n}$, which is the true worst-case success probability of PPZ (i.e., PPSZ with proof heuristic P_1) on 3-CNF formulas. We strongly encourage the reader to compute $\mathbb{E}_{\pi\sim Q|\alpha_1}[J_{a_1}(\alpha_1, \pi)]$ for the above example.

Problem Assessment

Since π is uniform under Q, it holds that $Q(\pi|\alpha)$ is proportional to $Q(\alpha|\pi) = 2^{-I(\pi,\alpha)}$. For $\alpha_1 = (0111^n)$, the latter term is largest when x comes first (as setting x to 0 implies the values of both y and z). Informally speaking, y and z tend to come later among x, y, z. When can P_1 tell the value of a_1? The clause $(\bar{y} \vee \bar{z} \vee a_1)$ reduces to the unit clause (a_1) if y, z come before a_1. Normally, this happens with probability 1/3. Under $Q|\alpha_1$, however, y and z tend to come later, and the probability decreases to 5/16, and thus $\mathbb{E}_{\pi \sim Q|\alpha_1}[J_{a_1}(\alpha_1, \pi)] = 11/16$.