Electronic Colloquium on Computational Complexity, Report No. 113 (2018)

PPSZ for $k \geq 5$: More Is Better

Dominik Scheder
Shanghai Jiaotong University

May 30, 2018

Abstract

We show that for $k \geq 5$, the PPSZ algorithm for k-SAT runs exponentially faster if there is an exponential number of satisfying assignments. More precisely, we show that for every $k \geq 5$, there is a strictly increasing function $f : [0,1] \to \mathbb{R}$ with $f(0) = 0$ that has the following property. If F is a k-CNF formula over n variables and $|\mathrm{sat}(F)| = 2^{\delta n}$ solutions, then PPSZ finds a satisfying assignment with probability at least $2^{-c_k n - o(n) + f(\delta)n}$. Here, $2^{-c_k n - o(n)}$ is the success probability proved by Paturi, Pudlák, Saks, and Zane [10] for k-CNF formulas with a unique satisfying assignment.

Our proof rests on a combinatorial lemma: given a set $S \subseteq \{0,1\}^n$, we can partition $\{0,1\}^n$ into subcubes such that each subcube B contains exactly one element of S. Such a partition $\mathcal{B}$ induces a distribution on itself, via $\Pr[B] = |B|/2^n$ for each $B \in \mathcal{B}$. We are interested in partitions that induce a distribution of high entropy. We show that, in a certain sense, the worst case ($\min_{S:|S|=s} \max_{\mathcal{B}} H(\mathcal{B})$) is achieved if S is a Hamming ball. This lemma implies that every set S of exponential size allows a partition of linear entropy. This in turn leads to an exponential improvement of the success probability of PPSZ.

1 Introduction

The problem of finding a satisfying assignment of a propositional formula in conjunctive normal form, short SAT, is one of the most important NP-complete problems. If every conjunct, or *clause*, contains at most k literals, we call the instance a k-CNF formula and the problem of finding a solution k-SAT. Many improved exponential algorithms for k-SAT have been designed over the years. The currently fastest algorithm PPSZ, named after its inventors Paturi, Pudlák, Saks, and Zane [10]. For $k = 3$ a tiny improvement is known [5, 14], but for $k \geq 4$ currently nothing beats PPSZ. The

ISSN 1433-8092

algorithm is beguilingly simple: iterate over the variables of F in random order; when considering variable x, fix it to 0 or 1 randomly, unless the correct value of x is *obvious*. To make this a formal algorithm, we have to specify what *obvious* means. Formally, suppose we are given a correct but incomplete heuristic P, which, when called on a formula F and variable x returns $P(F, x) \in \{0, 1, ?\}$. We call the heuristic P *correct* if it never makes a wrong assertion, that is, if $P(F, x) = b \in \{0, 1\}$ then $F \models (x = b)$, i.e., all satisfying assignments of F set x to b. We allow P to be *incomplete*, that is, it may answer $P(F, x) =?$ even if the correct value of x is already determined, i.e., even if $F \models (x = b)$ for some $b \in \{0, 1\}$. We state the PPSZ algorithm formally below in Algorithm 1.

2 The PPSZ Algorithm

Algorithm 1 PPSZ

```
1:  procedure PPSZ(β, π, F, P)
2:      α := the empty assignment on V
3:      for x ∈ V in the order of π do
4:          if P(F, x) ∈ {0, 1} then
5:              α(x) := P(F, x)
6:          else
7:              α(x) := β(x)
8:          end if
9:          F := F|_{x=α(x)}.
10:     end for
11:     if F has been satisfied then
12:         return α
13:     else
14:         return failure
15:     end if
16: end procedure
```

Note that PPSZ has two sources of randomness: π, the random order in which the variables are processed, and β, a random assignment in $\{0, 1\}^n$ from which it reads the bit $\beta(x)$ if the proof heuristic P fails to infer a value for x. It will be convenient to specify π and β as explicit input parameters to PPSZ; it is then the responsibility of the "user" to call PPSZ with random π and β.

The success probability of PPSZ depends crucially on the power of P. For example, if we chose P to be the "empty prover" that always answers "?", then PPSZ would amount to nothing more than pure guessing, and its success probability would obviously be $\frac{|\mathrm{sat}(F)|}{2^n}$. If we were to let P be the "complete prover" that outputs $P(F,x) = b$ whenever $F \models (x = b)$, its success probability would be 1. Of course, this is unrealistic: such a proof heuristic itself would be NP-hard. Consider the proof heuristic P_d, which checks whether $(x = b)$ can be derived from a set of up to d clauses in F. Clearly, P_d can be implemented in polynomial time as long as d is a constant; even for some slowly growing $d = d(n) \in \omega(1)$, we can implement it in subexponential time. Paturi, Pudlák, Saks, and Zane [10] prove the following bound on the success probability of PPSZ using the heuristic $P_{\omega(1)}$:

Theorem 1 (PPSZ on Unique k-SAT [10])**.** *Let F be a k-CNF formula with exactly one satisfying assignment α. Then*

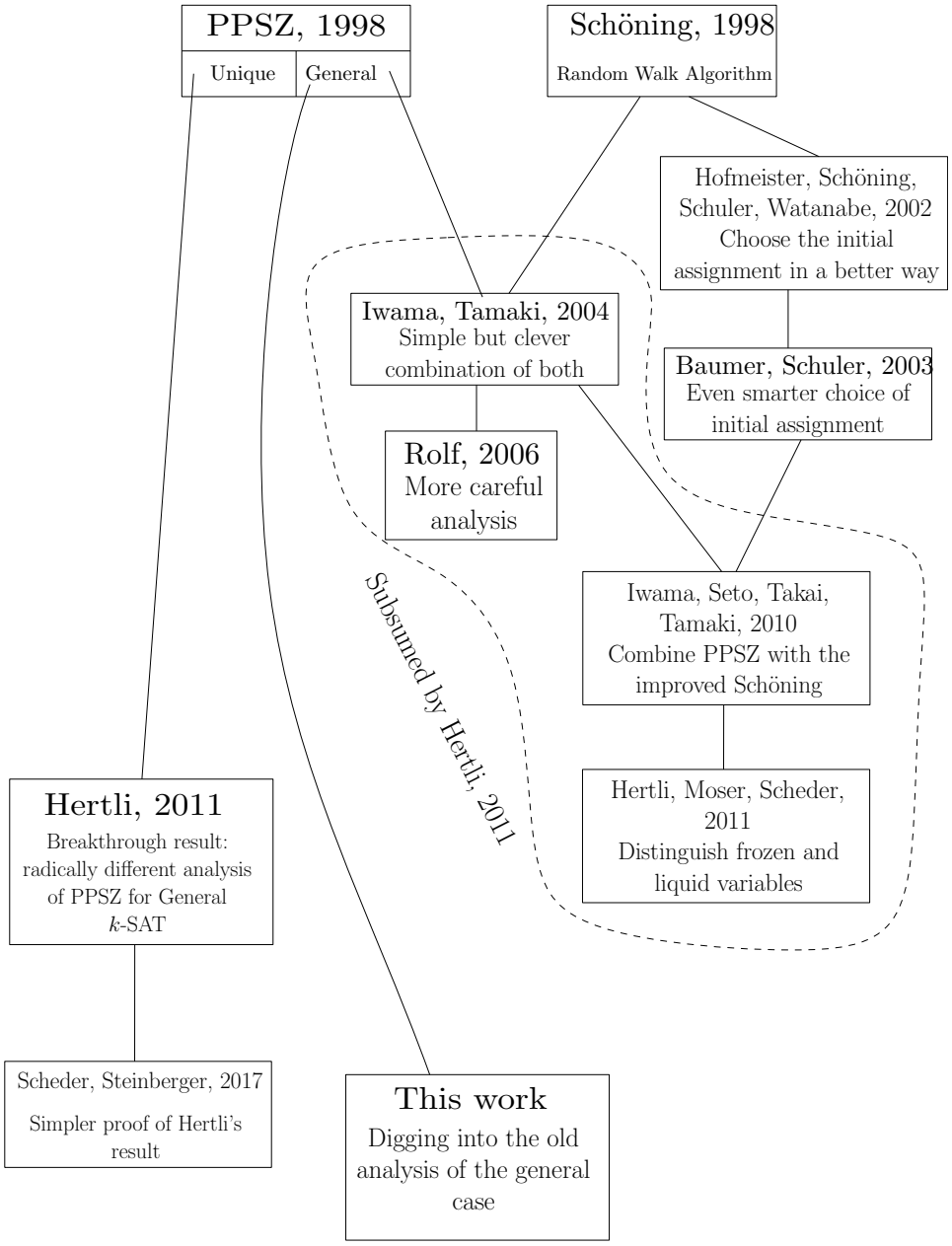
$$\Pr_{\pi,\beta}[\mathtt{ppsz}(\beta,\pi,F,P_{\omega(1)}) = \alpha] \geq 2^{-c_k n - o(n)} \ ,$$

where c_k is defined as follows: consider the infinite $(k-1)$-ary tree. Assign every node u a random value $\pi(u) \in [0,1]$ and delete all nodes u with $\pi(u) < \pi(\mathrm{root})$. Then c_k is the probability that the root is contained in an infinite connected component.

3 PPSZ for General k-SAT—Previous Work

Intuitively, having many solutions should make it easier for a randomized algorithm to find one. On second thought, though, having a unique solution might force F to have a special structure that can be exploited by an algorithm. This seems to be the case for PPSZ, where the "critical clause tree" is only guaranteed to exist if there is a unique solution. Still, with a more careful analysis, Paturi, Pudlák, Saks, and Zane [10] are to show that Theorem 1 holds also for General k-SAT, as long as $k \geq 5$.

Their key idea is to fix a partition $\mathcal{B}$ of the search space $\{0,1\}^n$ into subcubes, such that every subcube $B \in \mathcal{B}$ contains exactly one solution $\alpha \in \{0,1\}^n$. In a next step, they focus on analyzing $\Pr[\mathtt{ppsz}$ returns $\alpha \mid \beta \in B_\alpha]$, the probability that PPSZ returns α, conditioned on β being in the unique $\mathcal{B}$-subcube B_α that contains α. For $k \geq 5$, it turns out that a smaller B_α is always better, so in the worst case $B_\alpha = \{0,1\}^n$, meaning α is the only solution. For $k = 3, 4$ this is not the case, and some "medium size" B_α turns


PPSZ, 1998
Unique
General
Schöning, 1998
Random Walk Algorithm
Hofmeister, Schöning, Schuler, Watanabe, 2002
Choose the initial assignment in a better way
Iwama, Tamaki, 2004
Simple but clever combination of both
Baumer, Schuler, 2003
Even smarter choice of initial assignment
Rolf, 2006
More careful analysis
Subsumed by Hertli, 2011
Iwama, Seto, Takai, Tamaki, 2010
Combine PPSZ with the improved Schöning
Hertli, Moser, Scheder, 2011
Distinguish frozen and liquid variables
Hertli, 2011
Breakthrough result: radically different analysis of PPSZ for General k-SAT
Scheder, Steinberger, 2017
Simpler proof of Hertli's result
This work
Digging into the old analysis of the general case

out to be the worst case. Thus, their bounds for $k = 3, 4$, although better than anything known before, are exponentially worse than the unique case.

The work of Paturi, Pudlák, Saks, and Zane [10] has triggered a series of papers, all trying to narrow or close the gap. The impatient reader might have a look at Figure 3. Iwama and Tamaki [9] observed β landing in a small subcube immediately implies a better bound for Schöning's Random Walk algorithm, and achieved a substantial improvement. Rolf [12] made a small progress by analyzing $\Pr[\texttt{ppsz}$ returns $\alpha \mid \beta \in B_\alpha]$ more carefully. Iwama, Seto, Takai, and Tamaki [8] further improved things by combining PPSZ not with Schöning's algorithm, but with an improved version of Schöning's, due to Hofmeister, Schöning, Schuler, and Watanabe [7] and Baumer and Schuler [1]. Hertli, Moser, and Scheder [6] combined this with the idea of *liquid* versus *frozen* variables. A variable x is liquid if both $F|_{x=1}$ and $F|_{x=0}$ are satisfiable. As long as the fraction of liquid variables is high, every step by PPSZ (choose a variable at random and set it) has a high probability of keeping F satisfiable. Once this fraction falls below a certain threshold, they resort to the analysis of PPSZ. Combining this with [8], they obtain yet another small improvement.

All improvements mentioned above suffer from three weaknesses: first, they are very technical; second, the narrow the gap between general and unique 3-SAT, but do not close it; third, to understand them one needs to dig deep into the analysis of [10] and the workings of the proof heuristic $P_{\omega(1)}$. A breakthrough occurred in 2011, when Timon Hertli [4] published an analysis of PPSZ for the general case that elegantly bypasses any analysis of $P_{\omega(1)}$ and thus is much simpler than the one in [10], uses almost no information about the heuristic P, and therefore works for all k. Hertli's analysis is still quite technical and conveys little about "what is going on". Recently, John Steinberger and myself [14] gave an alternative proof of Hertli's result that is shorter and gives more intuition. Furthermore, it comes with a "lifting theorem" stating that improving PPSZ for Unique k-SAT immediately translates to a (smaller) improvement of PPSZ for general k-SAT. Together with an algorithm of Hertli that slightly beats PPSZ for Unique 3-SAT [5], this gives the currently best bounds for 3-SAT.

3.1 More Means Better?

Since Hertli [4] we know that the $2^{-c_k n - o(n)}$ bound holds for General k-SAT, as well. An obvious question is whether having many solutions actually *increases* the success probability of PPSZ. Before elaborating, let us think how much improvement we can expect in the best case. Suppose we have

an algorithm with success probability p^n. If F has $2^{\delta n}$ assignments, it could be the case that F ignores the first δn variables $x_1, \ldots, x_{\delta n}$. and is a worst-case instance on the remaining ones. So F is basically a worst-case instance over $(1-\delta)n$ variables and the best we can expect is a success probability of $p^{(1-\delta)n}$. If indeed our algorithm achieves this bound, i.e., has success probability at least $p^{(1-\delta)n}$ for every k-CNF formula F on n variables with $|\mathrm{sat}(F)| \geq 2^{\delta n}$, we say that F *scales optimally with the number of solutions*.

Suppose $g : [0,1] \to \mathbb{R}$ is some strictly increasing function with $g(0) = 0$ and thus $g(\delta) > 0$ for every $\delta > 0$. If our algorithm has success probability at least $p^{(1-g(\delta)n)}$ for every input k-CNF formula F with $|\mathrm{sat}(F)| \geq 2^{\delta n}$, we say that the algorithm *scales with the number of solutions*. Let us give some known examples. In most cases, proving that an algorithm scales involves a close look at is running time analysis plus some combinatorial statement about the Hamming cube.

Example: Trivial Guessing. The pure guessing algorithm, which simply guesses a solution and verifies it, has worst-case success probability at least 2^{-n}. If $|\mathrm{sat}(F)| = 2^{\epsilon n}$ then it succeeds with probability $2^{(1-\epsilon)n}$, so it scales optimally with the number of solutions.

Example: PPZ algorithm. The PPZ algorithm (Paturi, Pudlák, and Zane [11]) scales optimally with the number of solutions. This has been shown by Calabro, Impagliazzo, Kabanets, and Paturi [3]. The proof relies on the edge-isoperimetric inequality of the Hamming cube [2] and shows that the worst case is attained if $\mathrm{sat}(F)$ is a subcube. I include the proof in Appendix A.

Example: Schöning's algorithm. Schöning's random walk algorithm [15] scales with the number of solutions, but we do not know whether it scales optimally. This also seems to be an unpublished but well-known result. The proof uses the vertex-isoperimetric inequality of the Hamming cube [2] and shows that in the worst case, $\mathrm{sat}(F)$ is a Hamming ball (note that this cannot be the case if F is a k-CNF formula, so the analysis is most likely sub-optimal). I include a proof in Appendix B.

3.2 Our Contribution

Theorem 2. *The PPSZ algorithm on k-SAT scales with the number of satisfying assignments, as long as $k \geq 5$. More precisely, for every $k \geq 5$, there is a strictly increasing function $g_k(\delta)$ with $g_k(0) = 0$ such that PPSZ, called on a k-CNF formula F over n variables with at least $2^{\delta n}$ satisfying assignments, finds a solution with probability at least $2^{-c_k n + g_k(\delta)n - o(n)}$.*

We obtain Theorem 2 by looking again at the analysis of Paturi, Pudlák, Saks, and Zane [10] for the general case. This is surprising, since after Hertli's result [4], one would believe [10] and subsequent work had been subsumed. I tried to prove something like Theorem 2 using the analysis of Hertli [4] and of Steinberger and myself [14], bypassing the technicalities of the proof heuristic $P_{\omega(1)}$ and thus extending the theorem to $k = 3, 4$. However, this approach has not born fruit as of now.

4 Proof of Theorem 2

If F has multiple solutions, the analysis of the unique case breaks down. Indeed, any correct heuristic will fail to infer $x = b$ if both values 0 and 1 are feasible for x. Paturi, Pudlák, Saks, and Zane [10] deal with this in an ingenious way, which we now sketch.

Lemma 3 (Lemma 9 in [10])**.** *For every non-empty $S \subseteq \{0,1\}^n$ there is a partition $\mathcal{B} = \{B_\alpha\}_{\alpha \in S}$ of $\{0,1\}^n$ into subcubes B_α such that each B_α contains exactly one element from S, namely α. That is, $B_\alpha \cap S = \{\alpha\}$. We call $\mathcal{B}$ an S-partition of $\{0,1\}^n$.*

Proof. If $|S| = 1$ the lemma holds trivially by setting $\mathcal{B} = \{\{0,1\}^n\}$. Otherwise, there must be a coordinate i such that $S_b := \{x \in S \mid x_i = b\}$ is non-empty for both $b = 0$ and $b = 1$. Partition $\{0,1\}^n$ in two parts along x_i and recurse on S_0 and S_1. □

Note that this proof implicitly constructs a binary decision tree in which every leaf is labeled by a unique $\alpha \in S$. The assignments along the path from the root to the tree define the subcube B_α.

Definition 4. *An S-partition $\mathcal{B}$ defines a distribution on S (and thus on $\mathcal{B}$) in a natural way: $\Pr_{\mathcal{B}}[\alpha] := |B_\alpha|/2^n$. One can sample from this distribution by randomly descending from the root to a leaf in the aforementioned binary tree. Alternatively, one can sample from $\mathcal{B}$ by choosing $\beta \in \{0,1\}^n$ uniformly at random and outputting the unique $B \in \mathcal{B}$ containing β.*

The reader should take note that this partition only exists for the purpose of the analysis. We do not try to explicitly construct this partition or the corresponding tree; thus, when we say "sample from $\mathcal{B}$", we do not mean it in any algorithmic sense. From now on, let $S := \mathrm{sat}(F)$ and fix an S-partition $\mathcal{B}$ of $\{0,1\}^n$. We can look at $\Pr[\texttt{ppsz}(\beta, \pi, F, P)$ is successful] in a

more fine-grained way:

$$\begin{aligned}\Pr_{\beta,\pi}[\text{success}] &= \sum_{\alpha\in S} \Pr[\beta \in B_\alpha] \Pr[\text{success} \mid \beta \in B_\alpha] \\ &\geq \sum_{\alpha\in S} \Pr[\beta \in B_\alpha] \Pr[\texttt{ppsz}(\beta,\pi,F,P) = \alpha \mid \beta \in B_\alpha] \ . \qquad (1) \\ &\geq \min_{\alpha\in S} \Pr[\texttt{ppsz}(\beta,\pi,F,P) = \alpha \mid \beta \in B_\alpha] \ . \qquad (2)\end{aligned}$$

That is, we try to understand the probability that `ppsz` returns α, conditioned on the fact that the random string β is in B_α to begin with! Why does this make sense? Well, if B_α is small, then α and β already agree on a large number of variables, and thus many guesses of `ppsz` will be correct with probability 1. On the other hand, if B_α is large, then α is the only solution in the quite large region B_α. This makes it easier for the heuristic $P_{\omega(1)}$ to spot the correct choices for some variables x. To make the previous sentences precise, we would have to look in depth into the method of [10]. Instead we simply use the following corollary from [10]:

Lemma 5 (Corollary 14, page 360 in [10])**.** *Let $\alpha \in \mathrm{sat}(F)$ and let B_α its subcube in the $\mathrm{sat}(F)$-partition $\mathcal{B}$ and define $a_k := \frac{k-2}{k-1} \cdot 2^{c_k}$. Then*

$$\Pr[\texttt{ppsz}(\beta,\pi,F,P) = \alpha \mid \beta \in B_\alpha] \geq 2^{-c_k n - o(n)} \cdot a_k^{n - \log|B_\alpha|} \ , \qquad (3)$$

where c_k is defined in Theorem 1.

It turns out that $a_k > 1$ for $k \geq 5$ and thus the right hand side of (3) is larger for smaller B_α. Paturi, Pudlák, Saks, and Zane used this inequality to conclude that for $k \geq 5$, the worst case happens if $|B_\alpha| = 2^n$, i.e., if there is only one satisfying assignment. Lemma 5 comes a bit out of the blue. For the curious reader, we sketch a proof in Appendix C. Going back to inequality (1), we obtain a finer estimate:

$$\begin{aligned}\Pr_{\beta,\pi}[\text{success}] &\geq \sum_{\alpha\in S} \Pr[\beta \in B_\alpha] \Pr[\texttt{ppsz}(\beta,\pi,F,P) = \alpha \mid \beta \in B_\alpha] \quad \text{(by (1))} \\ &\geq 2^{-c_k n - o(n)} \cdot \sum_{\alpha\in S} \frac{|B_\alpha|}{2^n} \cdot a_k^{n-\log|B_\alpha|} \quad \text{(by (3))} \\ &= 2^{-c_k n - o(n)} \cdot \mathop{\mathbb{E}}_{B\sim\mathcal{B}} \left[a_k^{-\log_2 \Pr_{\mathcal{B}}[B]} \right] \ ,\end{aligned}$$

where we remember that $\mathcal{B}$ defines a probability distribution via $\Pr[B] := \frac{|B|}{2^n}$. By Jensen's Inequality, the above is at least

$$\begin{aligned} &2^{-c_k n - o(n)} a_k^{\mathbb{E}_{B\sim\mathcal{B}}[-\log_2 \Pr_{\mathcal{B}}[B]]} \\ =&2^{-c_k n - o(n)} a_k^{H(\mathcal{B})} \ , \end{aligned} \tag{4}$$

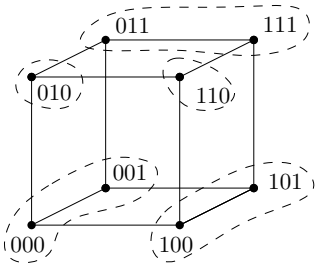
where $H(\mathcal{B})$ is the Shannon entropy of $\mathcal{B}$, viewed as a probability distribution over subcubes $B \in \mathcal{B}$. Since $a_k > 1$, we get an exponential "bonus" if $H(\mathcal{B})$ is linear.

Lemma 6. *Any set $S \subseteq \{0,1\}^n$ of size $2^{\Omega(n)}$ has an S-partition $\mathcal{B}$ for which $H(\mathcal{B}) \in \Omega(n)$. More formally, there is a strictly increasing function $g(\delta)$ with $g(0) = 0$ such that every $S \subseteq \{0,1\}^n$ of size $|S| = 2^{\delta n}$ has an S-partition of entropy at least $g(\delta) \cdot n$.*

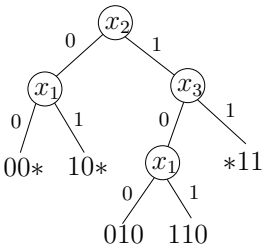
Together with (4), this proves Theorem 2. It remains to prove the lemma.

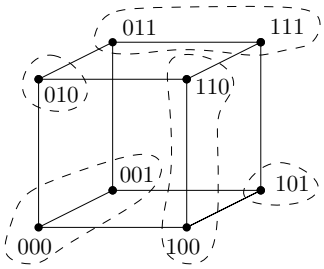
4.1 High-Entropy Tree-like Partitions—Proof of Lemma 6

Let T be a binary decision tree over n variables. Every path from the root to a node corresponds naturally to a partial assignment, and thus a subcube. We can imagine every node, in particular every leaf, being labeled with a subcube of $\{0,1\}^n$. In this way, a binary decision tree induces a partition of $\{0,1\}^n$ into subcubes. Note that the correspondence between decision trees and partitions is neither unique nor complete—a partition might be defined by different decision trees, and some partitions cannot be defined by a decision tree at all.



This is a treelike partition.
It can be described by a binary decision tree.
In this case, the tree happens to be unique.





This is not a treelike partition and cannot be described by a decision tree.

The problem is that even if S is exponential, there might be S-partitions $\mathcal{B}$ of *constant* entropy. For example, consider the set S consisting of all strings $x \in \{0,1\}^n$ in which a pair of consecutive 0's is followed by yet another 0 (and thus yet another, until the string is "full"). We now build a partition by always splitting on the first possible variable, in the order $x_1, x_2, \ldots, x_n$. If we take a random walk down from the root, we reach a leaf as soon as we go left twice in a row, since this means setting $x_i = x_{i+1} = 0$. If we imagine the tree to go on forever, it takes on expectation four steps until we reach a leaf. Since the tree is finite, the true expectation will be a bit smaller. In any case, $H(\mathcal{B}) \leq 4$ for the partition defined by this tree. A quick induction shows that $|S| = F_{n+3} - 1$, where F_n is the n^{th} Fibonacci number, so $|S|$ is clearly exponential in n.

So not every S-partition will have linear entropy. We will have to be a little bit careful how to construct the partition. We will give three different proofs.

First proof of Lemma 6. Let $B_n(d) := \{x \in \{0,1\}^n \mid |x|_1 \leq d\}$ be the Hamming ball of radius d around 0, and define $\Phi_n(d) := |B_n(d)| = \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{d}$. Let d be the largest integer such that $|S| \geq \Phi_n(d)$. The Sauer-Shelah Lemma [13, 16] states that there is a set $I \subseteq [n]$ of size d that is *shattered* by S, i.e., $S|_I = \{0,1\}^I$. Imagine a decision tree of depth d querying the variables of I, in any order. Each leaf of this tree is labeled with a subcube B of $\{0,1\}^n$, and $S \cap B \neq \emptyset$. Thus, we can extend this decision tree

at every leaf, refining the partition until each subcube contains exactly one element of S. We obtain a partition $\mathcal{B}$ of min-entropy at least d, and thus $H(\mathcal{B}) \geq H_{\min}(\mathcal{B}) \geq d$. Note that if $|S| \geq 2^{\delta n}$ then $d \geq (H^{-1}(\delta) - o(1)) \cdot n$, where H is the binary entropy function, so $d \in \Omega(n)$. □

If $S = B_n(d)$, then any treelike S-partition has a leaf at depth d, so $H_{\min}(\mathcal{B}) = d$, and in this sense the above proof is optimal. However, one checks that $H(\mathcal{B}) \approx 2d$ (we will give an exact formula below), so the above proof is suboptimal concerning H.

Second proof of Lemma 6. Sampling $x \in S$ uniformly at random gives us a random variable of entropy $\Omega(n)$. By sub-additivity of entropy, there is a coordinate $i \in [n]$ such that x_i has entropy $\Omega(1)$. If we split S along coordinate i, both sets S_0, S_1 have size at least $\Omega(|S|)$. That is, we lose a constant factor along every edge, and thus every leaf of the tree has linear depth. We obtain an S-partition with $H_{\min}(\mathcal{B}) \in \Omega(n)$. □

Although not needed to obtain our main result, it is natural to ask which set S behaves worst in this context. The next lemma, proved in the appendix, shows that the worst case is achieved by Hamming balls. This is the third proof of Lemma 6.

Lemma 7. *Let $B_n(d) := \{x \in \{0,1\}^n \mid |x|_1 \leq d\}$ be the Hamming ball of radius d centered at 0. Note that by symmetry, all treelike $B_n(d)$-partitions have the same entropy $g_n(d)$. Let $S \subseteq \{0,1\}^n$ be a set of size $|S| = |B_n(d)| = \Phi_n(d)$. Then there is a treelike S-partition $\mathcal{B}$ of entropy $g_n(d)$.*

A rough estimate gives us that $g_n(d) \approx 2d$ if $d/n < 1/2$ and n is large ($2d$ is the expected position of the d^{th} 1 in an infinite random bit string). We also have a precise formula:

$$g_n(d) = 2d + \frac{n \cdot \Phi_n(d-1) - d \cdot \Phi_{n+1}(d)}{2^n} \ .$$

We prove this formula in Lemma 14 below.

5 Proof of Lemma 7—Hamming Balls are Worst

For a non-empty set $S \subseteq \{0,1\}^n$, let $h(S) := \max H(\mathcal{B})$, where the maximum is taken over all S-partitions of $\{0,1\}^n$. Let $h_t(S)$ be the corresponding expression, where we maximize only over *treelike* partitions. Clearly, $h(S) \geq h_t(S)$. In this section we prove Lemma 7 by showing that Hamming balls

sets minimize $h_t(S)$. To make this precise, let $B_n(d) := \{x \in \{0,1\}^n \mid |x|_1 \leq d\}$ and $\Phi_n(d) = |B_n(d)| = \binom{n}{\leq d} = \sum_{i=0}^{d} \binom{n}{i}$. The following theorem is equivalent to Lemma 7.

Theorem 8. *Of all sets S of size $\Phi_n(d)$, the one minimizing h_t is the Hamming ball. That is, $h_t(S) \geq h_t(B_n(d))$ if $|S| = \Phi_n(d)$.*

We will prove this theorem by constructing a treelike S-partition. For this, we choose a coordinate i that partitions S into two parts S_0, S_1 in the most equitable way. Then we apply induction to S_0 and S_1. The problem with this approach is that even if S has the size of a Hamming ball, S_0 and S_1 need not. We therefore need to generalize the statement of the theorem to cover sets S of all sizes. First, define $g_n(d) := h_t(B_n(d))$. It is easy to see that $g_n(d) \approx 2d$ for large values of n: in an infinite random string, where does the d^{th} 1 appear? At position $2d$ on expectation. In Lemma 14 below, we will give an explicit formula for $g_n(d)$. For now, the following recurrence will be enough.

Lemma 9. *The function $g_n(d)$ satisfies the following recurrence:*

$$g_n(d) = \begin{cases} 0 & \text{if } d = 0\ , \\ n & \text{if } d = n\ , \\ 1 + \frac{g_{n-1}(d-1)+g_{n-1}(d)}{2} & \text{otherwise.} \end{cases}$$

Proof. If $d = 0$ then the partition contains only one part. If $d = n$ then the Hamming ball is the whole cube, and each $x \in \{0,1\}^n$ is in its own part, giving entropy n. Otherwise, assume without loss of generality that the treelike partition splits along some x_n and then recurses on the $(n-1)$-dimensional Hamming balls of radius d (left child, for $x_n = 0$) and $d-1$ (right child, for $x_n = 1$). Thus, the total expected depth is 1 plus the average of the expected depths of the two subtrees. □

Define a function f_n by setting $f_n(s) = g_n(d)$ if $s = \Phi_n(d)$; for all other values of $s \in [0, 2^n]$ we interpolate linearly between these points. Thus, $f_n(s)$ is a piecewise linear function.

Lemma 10. *The function f_n is concave on $[0, 2^n]$.*

The proof of this lemma is a tedious computation. We defer it to the end of this section. From now on, we aim to show that $h_t(S) \geq f_n(|S|)$.

To build our treelike partition, we split S along the "most equitable coordinate". More formally, for a set $S \subseteq \{0,1\}^n$, an index $1 \leq i \leq n$, and a value $b \in \{0,1\}$, define $S_b^i := \{x \in S \mid x_i = b\}$. Clearly, $S = S_0^i \cup S_1^i$ is a bipartition of S, for any i. We are interested in a most equitable partition, that is, an index i that maximizes $\min\{|S_0^i|, |S_1^i|\}$. Using information theoretic tools, it is easy to prove the following lemma:

Lemma 11. *Let $p \in [0, 1/2]$ be the unique number such that $|S| = 2^{H(p)\cdot n}$. Then there exists an index i such that $|S_0^i|$ and $|S_1^i|$ are both at least $p \cdot |S|$.*

Also note that this lemma is approximately tight if S is a Hamming ball. However, we want to that the Hamming ball is *exactly* optimal, thus Lemma 11 is not good enough. Before proving an optimal lemma, let us consider how $B := B_n(d)$ splits along a coordinate. If we fix x_i to be 0, we are left with $n-1$ coordinates, up to d of which can be 1. That is, $|B_0^i| = \Phi_{n-1}(d)$. If we fix x_i to be 1, we are left with $n-1$ coordinates, up to $d-1$ of which can be 1. Thus, $|B_1^i| = \Phi_{n-1}(d-1)$. Clearly $|B_1^i| \leq |B_0^i|$. The next lemma claims that every set S of size $\Phi_n(s)$ can be split at least as equitably as the Hamming ball $B_n(d)$.

Lemma 12. *If $S \subseteq \{0,1\}^n$ has size $\Phi_n(d)$, then there exists an index i such that $|S_0^i|$ and $|S_1^i|$ are both at least $\Phi_{n-1}(d-1)$.*

Proof. We can assume without loss of generality that $|S_0^i| \geq |S_1^i|$ for all i, that is, every coordinate i is biased towards 0. To see this, note that we can always reflect everything along a coordinate i without changing any combinatorial structure. Note that also $|B_0^i| \geq |B_1^i|$ holds for every i. Thus, it suffices to show that $|S_1^i| \geq |B_1^i|$ for some index i.

For a set $S \subseteq \{0,1\}^n$, define the *weight* of S to be $w(S) := \sum_{x \in S} |x|_1$, i.e., the total Hamming weight of all its elements. Which set of size $\Phi_n(d)$ minimizes $w(S)$? Obviously the one whose elements are as close to 0 as possible: the Hamming ball. In other words, $w(S) \geq w(B)$.

Note that $w(S) = \sum_{i=1}^n |S_1^i|$ for every set $S \subseteq \{0,1\}^n$. First, this implies that $w(B) = n\Phi_{n-1}(d-1)$. Second, by an averaging argument, it implies that there exists an index i for which $|S_1^i| \geq \frac{w(S)}{n} \geq \frac{w(B)}{n} = \Phi_{n-1}(d-1)$. □

We need a generalization of Lemma 12 that covers the case that S does not have the size of a Hamming ball.

Lemma 13. *Suppose $S \subseteq \{0,1\}^n$ has size $(1-\alpha)\Phi_n(d) + \alpha\Phi_n(d+1)$ for some $\alpha \in [0,1]$. Then there exists an index i such that $|S_0^i|$ and $|S_1^i|$ are both at least $(1-\alpha)\Phi_{n-1}(d-1) + \alpha\Phi_{n-1}(d)$.*

That is, the lower bound from Lemma 12 interpolates linearly if S does not have the size of a Hamming ball.

Proof. The size of S is $|S| = (1-\alpha)\Phi_n(d) + \alpha\Phi_n(d+1) = \Phi_n(d) + \alpha(\Phi_n(d+1) - \Phi_n(d)) = \Phi_n(d) + \alpha\binom{n}{d+1}$. Write $m = \alpha\binom{n}{d+1}$. As above, we assume that any coordinate i is biased towards 0, that is, $|S_0^i| \geq |S_1^i|$. Which set of size $|S|$ minimizes $w(S)$? By the same thought as in the previous proof, it is a set that contains $B_n(d)$ plus m additional elements of Hamming weight $d+1$. Thus, we get $w(S) \geq w(B_n(d)) + m(d+1) = n\Phi_{n-1}(d-1) + \alpha\binom{n}{d+1} \cdot (d+1)$. Since $w(S) = \sum_{i=1}^n |S_1^i|$, an averaging argument shows that there is some index i such that

$$\begin{aligned} |S_1^i| \geq& \Phi_{n-1}(d-1) + \alpha \cdot \binom{n}{d+1} \cdot \frac{d+1}{n} \\ =& \Phi_{n-1}(d-1) + \alpha \cdot \binom{n-1}{d} = (1-\alpha)\Phi_{n-1}(d-1) + \alpha\Phi_{n-1}(d) \ . \end{aligned}$$

This concludes the proof. □

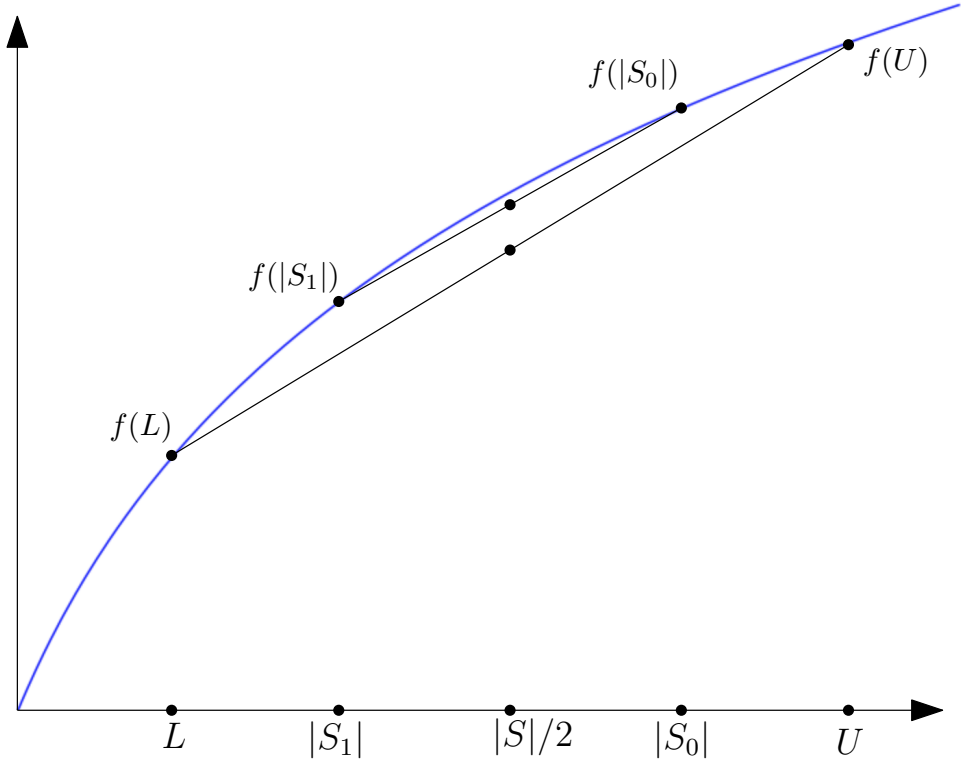
In words, if S has the size of a Hamming ball $B_n(d)$, then there is a coordinate that partitions S at least as equitably as $B_n(d)$; if the size of S is between that of two Hamming balls of radii d and $d+1$, then the "equitableness" of the best bipartition interpolates linearly between that of those two balls.

We are now ready to prove $h_t(S) \geq f_n(|S|)$ by induction. Let d and $\alpha \in [0,1]$ be such that $|S| = (1-\alpha)\Phi_n(d) + \alpha\Phi_n(d+1)$. Find the most equitable coordinate i, that is, the one maximizing $\min(|S_0^i|, |S_1^i|)$. By removing the i^{th} coordinate, the sets S_0^i, S_1^i correspond to sets $S_0, S_1 \in \{0,1\}^{n-1}$. By induction, S_0 and S_1 have a treelike partition of entropy at least $f_{n-1}(|S_0|)$ and $f_{n-1}(|S_1|)$, respectively. Thus, we conclude that

$$h_t(S) \geq 1 + \frac{f_{n-1}(|S_0|) + f_{n-1}(|S_1|)}{2}$$

Assuming without loss of generality that $|S_0| \geq |S_1|$, we obtain by Lemma 13 that $|S_1| \geq L := (1-\alpha)\Phi_{n-1}(d-1) + \alpha\Phi_{n-1}(d)$ and $|S_0| \leq U := |S| - L = (1-\alpha)\Phi_{n-1}(d) + \alpha\Phi_{n-1}(d+1)$.[1] Recall that f_n is a concave function, as stated in Lemma 10. Thus, the whole picture looks like this:

[1] To see the last inequality, note that $\Phi_n(d) = \Phi_{n-1}(d-1) + \Phi_{n-1}(d)$.



We abbreviate f_{n-1} as f in the above picture and the computation below. We see that concavity of f_{n-1} implies that

$$\begin{aligned} h_t(S) \geq & 1 + \frac{f(|S_0|) + f(|S_1|)}{2} \\ \geq & 1 + \frac{f(L) + f(U)}{2} \\ = & 1 + \frac{f((1-\alpha)\Phi_{n-1}(d-1) + \alpha\Phi_{n-1}(d)) + f((1-\alpha)\Phi_{n-1}(d) + \alpha\Phi_{n-1}(d+1))}{2} \end{aligned} .$$

Since f is linear on the interval $[\Phi_{n-1}(d-1), \Phi_{n-1}(d)]$ and on $[\Phi_{n-1}(d), \Phi_{n-1}(d+1)]$, the above expression equals

$$\begin{aligned} & 1 + \frac{(1-\alpha)(f(\Phi_{n-1}(d-1)) + f(\Phi_{n-1}(d))) + \alpha(f(\Phi_{n-1}(d)) + f(\Phi_{n-1}(d+1)))}{2} \\ = & (1-\alpha) \cdot \frac{1 + f(\Phi_{n-1}(d-1)) + f(\Phi_{n-1}(d))}{2} + \alpha \cdot \frac{1 + f(\Phi_{n-1}(d)) + f(\Phi_{n-1}(d+1))}{2} \\ = & (1-\alpha) \cdot \frac{1 + g_{n-1}(d-1) + g_{n-1}(d)}{2} + \alpha \cdot \frac{1 + g_{n-1}(d) + g_{n-1}(d+1)}{2} \\ = & (1-\alpha) g_n(d) + \alpha g_n(d+1) = f(|S|) \ , \end{aligned}$$

where the penultimate equality follows from Lemma 9 and the last equality from the definition of f_n and $|S| = (1-\alpha)\Phi_n(d) + \alpha\Phi_n(d+1)$. This finishes the proof.

5.1 Proof of Lemma 10

Recall that we defined $g_n(d)$ to be $h_t(B_n(d))$. In Lemma 9 we stated a recurrence for g_n: $g_n(d) = 1 + \frac{g_{n-1}(d)+g_{n-1}(d+1)}{2}$. We will now give an explicit formula for $g_n(d)$:

Lemma 14. $g_n(d) = 2d + \frac{n\cdot\binom{n}{\leq d-1} - d\cdot\binom{n+1}{\leq d}}{2^n}$.

Of course one could prove this lemma by induction, showing that the expression satisfies the recurrence of Lemma 9. This is a bit unsatisfactory, since it does not explain how one could possibly find the above explicit formula. We give a more "combinatorial" proof.

Proof. Imagine an infinite string $x = x_1x_2\ldots$. The probability that the d^{th} 1 appears at position i is $\frac{\binom{i-1}{d-1}}{2^i}$. Indeed, there are 2^i ways to choose $x_1, ..., x_i$, and for exactly $\binom{i-1}{d-1}$ of them do we have a 1 at position i and $d-1$ many 1's in the positions $1, \ldots, i-1$. Also, the probability that the $x_1, \ldots, x_n$ contain fewer than n 1's is $\frac{\binom{n}{\leq d-1}}{2^n}$. We can thus write down an exact formula for $g_n(d)$:

$$g_n(d) = \sum_{i=0}^{n} i \cdot \frac{\binom{i-1}{d-1}}{2^i} + n \cdot \frac{\binom{n}{\leq d-1}}{2^n} \ .$$

We finish the proof by showing that the first sum equals $2d - \frac{d\cdot\binom{n+1}{\leq d}}{2^n}$. □

Proposition 15. *The following identity holds:*

$$\sum_{i=0}^{n} i \cdot \frac{\binom{i-1}{d-1}}{2^i} = 2d - \frac{d \cdot \binom{n+1}{\leq d}}{2^n} \ .$$

Proof. First, observe that

$$\sum_{i=0}^{n} i \cdot \binom{i-1}{d-1} \cdot 2^{-i} = \sum_{i=0}^{\infty} i \cdot \binom{i-1}{d-1} \cdot 2^{-i} - \sum_{i=n+1}^{\infty} i \cdot \binom{i-1}{d-1} \cdot 2^{-i} \ . \quad (5)$$

The first sum evaluates to $2d$. To see this, note that the first sum is the expectation of the index at which the d^{th} 1 appears in the infinite string $x_1x_2x_3\ldots$. This is $2d$. Thus, let us work on the second sum:

$$\begin{aligned}
&\sum_{i=n+1}^{\infty} i \cdot \binom{i-1}{d-1} \cdot 2^{-i} = \sum_{i=n+1}^{\infty} d \cdot \binom{i}{d} \cdot 2^{-i} \\
=&d \cdot \sum_{i=n+1}^{\infty} \Pr[x_1 + \cdots + x_i = d] \\
=&d \cdot \mathbb{E}\left[|\{i \geq n+1 \mid x_1 + \cdots + x_i = d\}|\right]
\end{aligned}$$

Focus on the set $\{i \geq n+1 \mid x_1 + \cdots + x_i = d\}$. If the $(d+1)^{\text{st}}$ 1 appears within $x_1, \ldots, x_{n+1}$, then this set is empty. Otherwise, it includes all indices from (including) the d^{th} 1 (or from $n+1$, whichever comes later) to (not including) the $(d+1)^{\text{st}}$ 1. Now if the d^{th} 1 comes after $n+1$, then the expected size of this set is 2—it takes on average two tosses until we see the next 1, after the d^{th}. If the d^{th} is among the positions $1, \ldots, n+1$, then, conditioned on the next 1 coming after $n+1$, it is again 2. The probability that the $(d+1)^{\text{st}}$ 1 comes strictly after $n+1$ is $\binom{n+1}{\leq d} \cdot 2^{-(n+1)}$. Thus,

$$\begin{aligned}
&\sum_{i=n+1}^{\infty} i \cdot \binom{i-1}{d-1} \cdot 2^{-i} = d \cdot \mathbb{E}\left[|\{i \geq n+1 \mid x_1 + \cdots + x_i = d\}|\right] \\
=&2 \cdot d \cdot \binom{n+1}{\leq d} \cdot 2^{-(n+1)} = d \cdot \binom{n+1}{\leq d} \cdot 2^{-n}
\end{aligned}$$

This concludes the proof. □

In the rest of this section we prove that our piecewise linear function $f_n : [0, 2^n] \to \mathbb{R}$ is concave. Since f is piecewise linear, it suffices to show that f is concave at values $s = \Phi_n(d)$ for $1 \leq d \leq n-1$. For this, we have to show that the point $(\Phi_n(d), g_n(d))$ lies above the line ℓ from $(\Phi_n(d-1), g_n(d-1))$ to $(\Phi_n(d+1), g_n(d+1))$. Note that we can write $\Phi_n(d) = a\Phi_n(d+1) + (1-a)\Phi_n(d-1)$ for a unique $a \in [0,1]$. The y-coordinate of the line ℓ at position $x = \Phi_n(d)$ is $ag_n(d+1) + (1-a)g_n(d-1)$. Thus, we have to show that $ag_n(d+1) + (1-a)g_n(d-1) \leq g_n(d)$. It is easy to see that the value of a is

$$a = \frac{\Phi_n(d) - \Phi_n(d-1)}{\Phi_n(d+1) - \Phi(d-1)} = \frac{\binom{n}{d}}{\binom{n}{d+1} + \binom{n}{d}} = \frac{\binom{n}{d}}{\binom{n+1}{d+1}} = \frac{d+1}{n+1} \ .$$

Now let us show the above claimed inequality:

$$\begin{aligned} a g_n(d+1) + (1-a) g_n(d-1) \leq \quad & g_n(d) \quad \Longleftrightarrow \\ (d+1) 2^n g_n(d+1) + (n-d) 2^n g_n(d-1) \leq \quad & (n+1) 2^n g_n(d) \ . \end{aligned}$$

The left-hand side of this claimed inequality equals

$$\begin{aligned} & (d+1) \left(2^{n+1}(d+1) + n \Phi_n(d) - (d+1) \Phi_{n+1}(d+1) \right) \\ + & (n-d) \left(2^{n+1}(d-1) + n \Phi_n(d-2) - (d-1) \Phi_{n+1}(d-1) \right) \end{aligned}$$

and the right-hand side equals

$$(n+1) d 2^{n+1} + n(n+1) \Phi_n(d-1) - d(n+1) \Phi_{n+1}(d) \ .$$

Expanding and comparing these expressions becomes longish, so let us break us down both sides and their difference separately for the coefficients of n^2, d^2, nd, n, d, and 1, respectively.

	left-hand side	right-hand side
n^2	$\Phi_n(d-2)$	$\Phi_n(d-1)$
d^2	$-\Phi_{n+1}(d+1) + \Phi_{n+1}(d-1) = -\binom{n+1}{d+1} - \binom{n+1}{d} = -\binom{n+2}{d+1}$	0
nd	$\Phi_n(d) + 2^{n+1} - \Phi_n(d-2) - \Phi_{n+1}(d-1)$	$2^{n+1} - \Phi_{n+1}(d)$
n	$\Phi_n(d) - 2^{n+1} + \Phi_{n+1}(d-1)$	$\Phi_n(d-1)$
d	$3 \times 2^{n+1} - 2 \cdot \Phi_{n+1}(d+1) - \Phi_{n+1}(d-1)$	$2^{n+1} - \Phi_{n+1}(d)$
1	$2^{n+1} - \Phi_{n+1}(d+1)$	0

Let us now compute the right-hand side minus the left-hand side. Using the recurrence $\Phi_{n+1}(d) = \Phi_n(d) + \Phi_n(d-1)$ and $\Phi_n(d) = \Phi_n(d-1) + \binom{n}{d}$, we can further simplify the expressions:

	right-hand side minus left-hand side	simplified
n^2	$\binom{n}{d-1}$	$\binom{n}{d-1}$
d^2	$\binom{n+2}{d+1}$	$\binom{n+2}{d+1}$
nd	$-2\binom{n+1}{d}$	$-2\binom{n+1}{d}$
n	$2^{n+1} - \Phi_{n+1}(d-1) - \binom{n}{d}$	$2^{n+1} - 2\Phi_n(d) + \binom{n+1}{d}$
d	$-2 \times 2^{n+1} + 2\Phi_{n+1}(d) + 2\binom{n+1}{d+1} - \binom{n+1}{d}$	$-2 \times 2^{n+1} + 4\Phi_n(d) + 2\binom{n}{d+1} - \binom{n+1}{d}$
1	$\Phi_{n+1}(d+1) - 2^{n+1}$	$-2^{n+1} + 2\Phi_n(d) + \binom{n}{d+1}$

Let us now collect things again, separating terms containing a binomial coefficients from those containing 2^{n+1} or $\Phi_n(d)$. The difference between

the right-hand and the left-hand side equals

$$2 \cdot (2^n - \Phi_n(d)) \cdot (n - 2d - 1) +$$

$$n^2 \binom{n}{d-1} + d^2 \binom{n+2}{d+1} - 2nd \binom{n+1}{d} + n \binom{n+1}{d} + 2d \binom{n}{d+1} - d \binom{n+1}{d} + \binom{n}{d+1}$$

The second line looks complicated. However, by multiplying the second line by $\frac{(d+1)!(n-d+1)!}{n!}$ and expanding each binomial coefficient as $\binom{n}{d} = \frac{n!}{d!(n-d)!}$, one obtains that the second line equals $2(n-d)\binom{n}{d}$. Thus, the difference is

$$2 \cdot (2^n - \Phi_n(d)) \cdot (n - 2d - 1) + 2(n-d) \binom{n}{d} .$$

We can further simplify this expression by defining $r := n - d$ and replacing d by $n - r$. Using the fact that $\binom{n}{d} = \binom{n}{r}$ and $2^n - \Phi_n(d) = \Phi_n(r-1)$ and dividing everything by 2, we conclude that the above expression is twice the one below:

$$\Delta_n(r) := \Phi_n(r-1)(2r - n - 1) + r \binom{n}{r} .$$

With the next proposition, we show that $\Delta_n(r) \geq 0$ for all $1 \leq r \leq n-1$.

Proposition 16. *$\Delta_n(r) \geq 0$ is non-negative for all $1 \leq r \leq n-1$.*

Proof. We proceed by induction. For $r = 1$, we obtain $\Delta_n(1) = \Phi_n(0)(2 - n - 1) + \binom{n}{1} = 1$. This also covers the case $n = 2$ since for $n = 2$ the only possible value for r is 1 (recall that $1 \leq r \leq n-1$). So we can now assume that $n \geq 3$. What about $r = n - 1$? In this case, $2r - n - 1 = n - 3 \geq 0$, so both terms above are non-negative. For $n \geq 3$ and $2 \leq r \leq n-2$ we know by induction that

$$\Delta_{n-1}(r) = \Phi_{n-1}(r-1)(2r - n) + r \binom{n-1}{r} \geq 0 \quad \text{and}$$

$$\Delta_{n-1}(r-1) = \Phi_{n-1}(r-2)(2r - 2 - n) + (r-1) \binom{n-1}{r-1} \geq 0 .$$

Also, applying the recursive formula $\Phi_n(r) = \Phi_{n-1}(r) + \Phi_{n-1}(r-1)$ and $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ we see that

$$\Delta_n(r) = (\Phi_{n-1}(r-1) + \Phi_{n-1}(r-2))(2r - n - 1) + r \left(\binom{n-1}{r} + \binom{n-1}{r-1} \right) .$$

Finally, we compute $\Delta_n(r) - \Delta_{n-1}(r) - \Delta_{n-1}(r-2)$:

$$\Delta_n(r) - \Delta_{n-1}(r) - \Delta_{n-1}(r-2) = -\Phi_{n-1}(r-1) + \Phi_{n-1}(r-2) + \binom{n-1}{r-1} = 0 .$$

Thus, $\Delta_n(r)) = \Delta_{n-1}(r) + \Delta_{n-1}(r-2) \geq 0$. □

6 Conclusion and Open Questions

The biggest drawback of our result is that is does not apply for $k = 3, 4$. Indeed, the appeal of Timon Hertli's work [4] is that it generalizes the analysis of PPSZ to all k and that it treats the workings of the proof heuristic as a black box. Our proof above does not—it relies on Lemma 9 in [10], which in turn looks deeply into the details of the proof heuristic $P_{\omega(1)}$.

6.1 More General Setting?

One hope is to refine the analysis of Steinberger and myself [14]. Treelike partitions also appear there (albeit implicitly). The trouble is that in [14], the decision tree queries the variables in a random order, and thus we do not have the luxury of choosing an adequate or even optimal S-partition. Indeed, it is easy to construct sets S of exponential size such that $\mathbb{E}_\pi[H(\mathcal{B}_\pi)]$ is sublinear, where $\mathcal{B}_\pi$ is the treelike S-partition defined by querying the variables in the order of π. One such example would be $S := B_d(n) \cup \{x \in \{0,1\}^n \mid x_1 = \cdots = x_{n/2} = 1\}$ for d being some large constant. Querying the variables in random order, it will only take a constant number of steps until a "wrong" decision $x_i = 0$ for $i \leq n/2$ is made, from which on we are left with at most $|B_d(n)| \leq n^d$ elements. Indeed, the expected entropy will be logarithmic.

6.2 Easier Proof that Hamming Balls are Worst

Is there an easier proof of Theorem ? Our proof is a tedious induction with lots of calculations. It would be nice to have a more "combinatorial" one, maybe along the lines of the proof of the vertex isoperimetric inequality of the Hamming cube.

6.3 General S-Partitions

We proved that every set S of size $|S| = B_n(d)$ has as treelike S-partition whose entropy is at least that of a treelike partition of $B_n(d)$. We also gave an explicit formula for the latter, which is roughly equal to $2d$ when $d/n < 1/2$. However, as illustrated above, not all S-partitions are treelike partitions. This raises the question whether $B_n(d)$ allows partitions of higher entropy than treelike ones. I think not, but I do not have a proof.

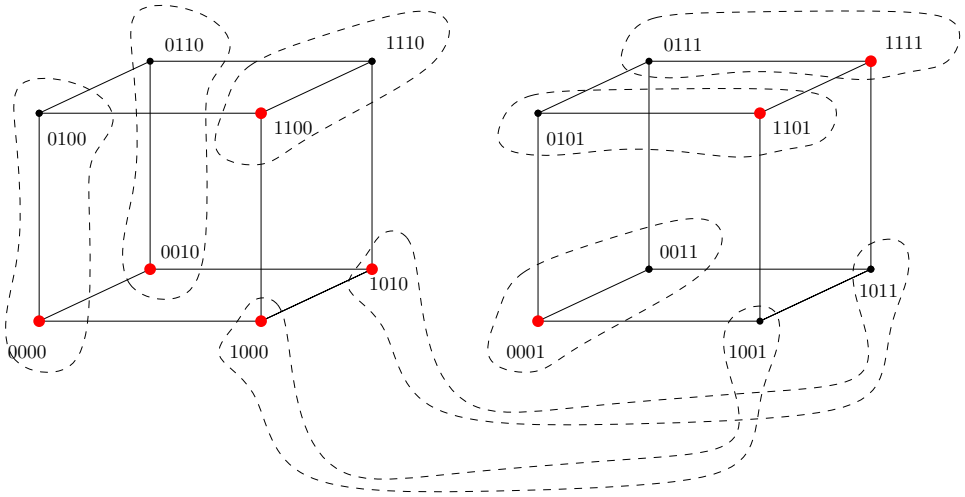
More formally, remember that we defined $h(S) := \max H(\mathcal{B})$, where the maximum is taken over all S-partitions. Taking the maximum over all treelike partitions, we obtain $h_t(S)$.

Conjecture 17. *Treelike partitions are optimal for the Hamming ball:* $h(B_n(d)) = h_t(B_n(d))$.

Finally let us define non-adaptive treelike partitions, which query the variables in a fixed ordering π, but are allowed to skip a variable x_i if its value in the remaining set is already fixed. Let $h_{\mathrm{na}}(S)$ denote the maximum entropy of a non-adaptive treelike partition. Clearly $h_{\mathrm{na}}(S) \leq h_t(S) \leq h(S)$. Also, our first proof of Lemma 6 shows that $h_{\mathrm{na}}(S) \in \Omega(n)$ if S has exponential size.

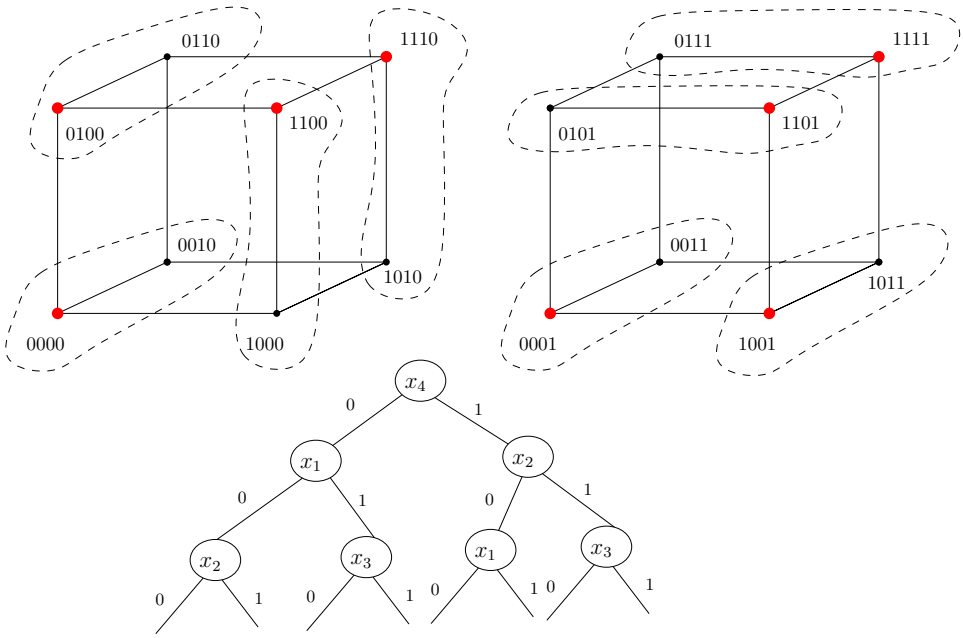
Question 18. *What is the relation between* $h_{\mathrm{na}}(S)$, $h_t(S)$, *and* $h(S)$*? How far can they be apart?*

Here is an example of a set $S \subseteq \{0,1\}^4$ for which $h_t(S) < h(S) = 3$:



The above partition induces the uniform distribution over S and thus $h(S) = 3$. It is easy to check that no treelike partition can achieve a uniform distribution.

Next, we will give a set $S \subseteq \{0,1\}^4$ that has a treelike partition achieving the uniform distribution, giving entropy 3.



Again, it is easy to check that the only way to achieve the uniform distribution is to (1) first query x_4; (2) querying x_1 or x_2, depending on whether $x_4 = 0$ or $x_4 = 1$. Thus, the optimal tree must be adaptive, and thus $h_{\mathrm{na}}(S) < h_t(S) = 3$.

References

[1] S. Baumer and R. Schuler. Improving a probabilistic 3-sat algorithm by dynamic search and independent clause pairs. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2003.

[2] B. Bollobás. *Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability*. Cambridge University Press, New York, NY, USA, 1986.

[3] C. Calabro, R. Impagliazzo, V. Kabanets, and R. Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. *J. Comput. Syst. Sci.*, 74(3):386–393, 2008.

[4] 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.

[5] T. Hertli. Breaking the PPSZ Barrier for Unique 3-SAT. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. 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.

[6] T. Hertli, R. A. Moser, and D. Scheder. Improving PPSZ for 3-SAT using critical variables. In *Proceedings of STACS*, pages 237–248, 2011.

[7] T. Hofmeister, U. Schöning, R. Schuler, and O. Watanabe. A probabilistic 3-sat algorithm further improved. In H. Alt and A. Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 192–202. Springer, 2002.

[8] K. Iwama, K. Seto, T. Takai, and S. Tamaki. Improved randomized algorithms for 3-sat. In O. Cheong, K. Chwa, and K. Park, editors, *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, volume 6506 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2010.

[9] 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.

[10] 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.

[11] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. *Chicago J. Theoret. Comput. Sci.*, pages Article 11, 19 pp. (electronic), 1999.

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

[13] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145 – 147, 1972.

[14] D. Scheder and J. P. Steinberger. PPSZ for General k-SAT - making Hertli's analysis simpler and 3-SAT faster. In R. O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 9:1–9:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[15] 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.

[16] S. Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972.

A PPZ Scales Optimally

In our terminology, the PPZ algorithm is the special case of PPSZ using P_1 as a proof heuristic. That is, in every step is checks whether the variable to be set is already contained in a unit clause. If so, it sets it accordingly, otherwise it guesses its truth value. For a satisfying assignment α let $I(\alpha)$ denote the number of variables x such that $\alpha \oplus x \notin \mathrm{sat}(F)$. Here, $\alpha \oplus x$ is the assignment obtained from α by flipping its value on x. Paturi, Pudlák, and Zane [11] prove that

$$\Pr[\text{PPZ returns } \alpha] \geq 2^{-n+I(\alpha)/k} .$$

This is not stated as a separate lemma in [11], but it is well-known and follows from Lemma 2 in [11] (the Satisfiability Coding Lemma) by Jensen's Inequality. Now let $S := \mathrm{sat}(F)$ be the set of satisfying assignments and suppose $|S| = 2^{\delta n}$. By the above inequality, we obtain

$$\Pr[\text{PPZ succeeds}] \geq \sum_{\alpha \in S} 2^{-n+I(\alpha)/k} = |S| \mathop{\mathbb{E}}_{\alpha \in S}\left[2^{-n+I(\alpha)/k}\right] \geq |S| 2^{-n+\mathbb{E}_{\alpha \in S}[I(\alpha)]/k} . \tag{6}$$

Note that $\mathbb{E}_\alpha[I(\alpha)] = \frac{1}{|S|}\sum_\alpha I(\alpha) = \frac{1}{|S|} e(S, \bar{S})$, where $e(S, \bar{S})$ is the number of edges from S to its complement, if we view $\{0,1\}^n$ as a graph (the

Hamming cube). The edge isoperimetric inequality of the Hamming cube [2] states that $e(S, \bar{S}) \geq |S|(n - \log|S|) = 2^{\delta n}(n - \delta n)$, which is tight if S is a subcube of dimension δn. Thus, we see that (6) is at least

$$|S|2^{-n+(n-\delta n)/k} = 2^{\delta n - n + (n-\delta)n/k} = 2^{-n(1-\delta)(1-1/k)} ,$$

and thus PPZ scales optimally with the number of satisfying assignments.

B Schöning Scales

Schöning's Random Walk Algorithm samples a uniformly random assignment $\beta \in \{0,1\}^n$. It then applies $3n$ local correction steps: as long as β violates some clause $\{u_1, \ldots, u_k\}$, it picks some u_i at random from this clause and flips its value under β, thus satisfying this clause (and possibly violating some others). Schöning showed that if the input formula is a k-CNF formula and α is some satisfying assignment, then the local correction phase finds some satisfying assignment (possibly different from α) with probability at least

$$\frac{1}{\mathrm{poly}(n)}(k-1)^{-d(\alpha,\beta)},$$

where d is the Hamming distance on $\{0,1\}^n$, i.e., the number of variables on which α and β disagree. Note that β is random, so $d(\alpha, \beta) = X_1 + \cdots + X_n$, where X_i is 1 if $\alpha(x_i) \neq \beta(x_i)$ and 0 otherwise. Those variables are all independent and therefore

$$\mathop{\mathbb{E}}_{\beta}\left[(k-1)^{-d(\alpha,\beta)}\right] = \prod_{i=1}^{n} \mathbb{E}\left[(k-1)^{-X_i}\right] = \left(\frac{k}{2(k-1)}\right)^n .$$

For example, this means that Schöning's algorithm has a success probability of $(3/4)^n/\mathrm{poly}(n)$ on satisfiable 3-CNF formulas.

Towards analyzing Schöning's Algorithm for formulas with many satisfying assignments, let $S := \mathrm{sat}(F)$. Since the choice of $\alpha \in S$ is arbitrary, we observe that the above probability is actually at least

$$\frac{1}{\mathrm{poly}(n)}(k-1)^{-d(\beta,S)},$$

where $d(\beta, S)$ is the Hamming distance from β to the closest $\alpha \in S$. For $i \in \mathbb{N}$ and a set $A \subseteq \{0,1\}^n$ let $\Gamma_i(A) := \{y \in \{0,1\}^n \mid d(y, A) \leq i\}$. Let $B(n, r) := \{x \in \{0,1\}^n \mid d(x, 0) \leq r\}$ be the Hamming ball of radius r centered at the

all-0-vector. The vertex isoperimetric inequality of the Hamming cube [2] states that if $|A| \geq |B(n,r)|$ then $|\Gamma_i(A)| \geq |\Gamma_i(B(n,r))| = |B(n,r+i)|$. Let r be the largest integer such that $|S| \geq |B(n,r)|$ and let $S^* := B(n,r)$. Let $Y \in \{0,1\}^n$ be a random point in the Hamming cube and let $D := d(Y,S)$ and $D^* := d(Y,S^*)$. D and D^* are random variables, and the above vertex isoperimetric inequality states that $\Pr[D \leq i] \geq \Pr[D^* \leq i]$. In other words, D^* statistically dominates D, and therefore

$$\begin{aligned}\Pr[\text{Schöning succeeds}] &\geq \frac{1}{\text{poly}} \mathop{\mathbb{E}}_{\beta}\left[(k-1)^{-d(\beta,S)}\right] \\ &= \frac{1}{\text{poly}} \mathbb{E}\left[(k-1)^{-D}\right] \\ &\geq \frac{1}{\text{poly}} \mathbb{E}\left[(k-1)^{-D^*}\right] \ . \end{aligned} \tag{7}$$

This means that under the above analysis, Schöning's algorithm performs worst if $\text{sat}(F)$ forms a Hamming ball. Now if $|S| \geq 2^{\delta n}$ then $r \geq \rho n - o(n)$, where ρ is such that $H(\rho) = \delta$, H being the binary entropy function. From this it follows easily that (7) is exponentially larger than $\left(\frac{k}{2(k-1)}\right)^n$ if $\delta > 0$.

We spare the reader the details of calculating exact bounds, since the above analysis is most likely suboptimal, anyway: the set $\text{sat}(F)$ cannot possibly be a Hamming ball of radius $\Omega(n)$ if F is a k-CNF. Thus, an analysis that takes into account the fact that F is a k-CNF might reveal a much better scaling behavior. However, no such analysis is known to me at this point in time.

C Analysis of PPSZ for General k-SAT—Proof of Lemma 5

In this section we sketch the proof of Lemma 5. This is basically a summary of Section 4 in [10], including all main arguments but skipping over some details and calculations. Our aim is to be sufficiently detailed so that all gaps could be filled by the reader given a moderate amount of time and sufficient motivation. For full detail, we refer the reader to [10].

Lemma 19 (Lemma 5, restated)**.** *Let $\alpha \in \text{sat}(F)$ and let B_α its subcube in the $\text{sat}(F)$-partition $\mathcal{B}$ and define $a_k := \frac{k-2}{k-1} \cdot 2^{c_k}$. Then*

$$\Pr[\texttt{ppsz}(\beta,\pi,F,P) = \alpha \mid \beta \in B_\alpha] \geq 2^{-c_k n} \cdot a_k^{n - \log|B_\alpha|} \cdot 2^{-o(n)} \ , \tag{8}$$

where c_k is defined in Theorem 1.

Let F be a CNF formula and $\alpha \in \mathrm{sat}(F)$ a solution. In a thought experiment, we call PPSZ (refer to the pseudocode in Algorithm 1) with $\texttt{ppsz}(\alpha, \pi, F, P)$. Yes, we actually feed the solution α into it. Observe that this always returns α, no matter what π or P is. We define some bookkeeping variables. For $x \in V(F)$, let $C_x(\pi, \alpha)$ be 1 if PPSZ was not able to infer $\alpha(x)$ using the heuristic P—that is, if PPSZ sets x in the **else**-clause in Line 7. Otherwise, if PPSZ sets x in the **then**-clause in Line 5 we set $C_x(\pi, \alpha) = 0$. Finally, $C(\pi, \alpha) = \sum_{x \in V(F)} C_x(\pi, \alpha)$. Note that $C(\pi, \alpha)$ is the number of bits PPSZ looks up in β. Note that $\texttt{ppsz}(\beta, \pi, F, P)$ returns α if and only if α and β agree on all variables where PPSZ probes β. Thus,

$$\Pr_{\beta}[\texttt{ppsz}(\beta, \pi, F, P) = \alpha] = 2^{-C(\pi,\alpha)}, \tag{9}$$

for every fixed π, and

$$\Pr_{\beta,\pi}[\texttt{ppsz}(\beta, \pi, F, P) = \alpha] = \mathop{\mathbb{E}}_{\pi}\left[2^{-C(\pi,\alpha)}\right] \geq 2^{-\mathbb{E}_{\pi}[C(\pi,\alpha)]} ,$$

which follows from Jensen's Inequality. By linearity of expectation we get that $\mathbb{E}_{\pi}[C(\pi, \alpha)] = \sum_{x \in V(F)} \mathbb{E}_{\pi}[C_x(\pi, \alpha)]$. If α is the unique satisfying assignment, we could now focus on analyzing $\mathbb{E}[C_x(\pi, \alpha)]$. In the case of multiple solutions, though, we have to be a bit more careful.

Let $S = \mathrm{sat}(F)$ and fix an S-partition $\mathcal{B}$. From now on, we fix a solution $\alpha \in \mathrm{sat}(F)$ and let $B = B_\alpha$ be the unique subcube $B \in \mathcal{B}$ containing α, and α is the unique satisfying assignment in B. We assume without loss of generality that $\alpha = (1, \ldots, 1)$. Thus, B is of the form $B = \{\beta \in \{0,1\}^n \mid \beta(y) = 1 \ \forall y \in V_D\}$ for some $V_D \subseteq V$. Let $V_N := V \setminus V_D$. Paturi, Pudlák, Saks, and Zane call V_D the *defining variables* and V_N the *non-defining variables* of B. Note that $|B| = 2^{|V_N|}$. The trick is now to condition on $\beta \in B$. Instead of (9) we obtain

$$\Pr_{\beta \in B}[\texttt{ppsz}(\beta, \pi, F, P) = \alpha] = 2^{-\sum_{x \in V_N} C_x(\pi,\alpha)} ,$$

since under the condition $\beta \in B$, β and α already agree on every $y \in V_D$ with probability one. We could now directly apply Jensen's inequality, but this will not be enough. The reason is that Jensen's inequality can be very loose $\sum_{x \in V_N} C_x(\pi, \alpha)$ varies much with π. So let us think for a minute which permutations π are "good" for us. We want the sum to be small. Some thought shows that moving x to the back of π can only decrease $C_x(\pi, \alpha)$. Of course, not all x can be at the end of π; however, we only sum over $x \in V_N$. Thus, it

would be beneficial for us if all $y \in V_D$ tended to occur towards the beginning of π. Note that we choose π uniformly from all permutations. From now on, it will make more sense to think of π as a function $\pi : V \to [0,1]$, where each $\pi(x)$ is chosen independently and uniformly at random from $[0,1]$. Let Γ be the event $\Gamma := \left\{\pi \mid \pi(y) \leq \frac{k-1}{k-2}\ \forall y \in V_D\right\}$. This event happens with probability $\left(\frac{k-1}{k-2}\right)^{|V_D|}$; conditioned on Γ, non-defining variables $y \in V_N$ tend to come earlier in π than defining variables $x \in V_D$. Formally, we get

$$\begin{aligned}\Pr_{\beta\in B,\pi}[\texttt{ppsz}(\beta,\pi,F,P)=\alpha] &= \mathop{\mathbb{E}}_{\pi}\left[2^{-\sum_{x\in V_N} C_x(\pi,\alpha)}\right] \\ &\geq \Pr[\Gamma]\cdot \mathop{\mathbb{E}}_{\pi\in\Gamma}\left[2^{-\sum_{x\in V_N} C_x(\pi,\alpha)}\right] \\ &\geq \Pr[\Gamma]\cdot 2^{-\sum_{x\in V_N}\mathbb{E}_{\pi\in\Gamma}[C_x(\pi,\alpha)]}\ . \end{aligned} \tag{10}$$

From now on, we will analyze $\mathbb{E}_{\pi\in\Gamma}[C_x(\pi,\alpha)]$ for a fixed non-defining variable $x \in V_N$. Note that F and α are fixed, too, and we assume $\alpha = (1,\ldots,1)$ for notational convenience.

C.1 Critical Clause Trees

The central combinatorial object in analyzing $\mathbb{E}_{\pi\in\Gamma}[C_x(\pi,\alpha)]$ is the *critical clause tree*. In this tree T, every node u will have a variable label $\mathrm{var}(u)$, and some might have a clause label $\mathrm{clause}(u)$. Additionally, every node u has an assignment label α_u obtained as follows: we start with the assignment α; then we walk from the root of T to u, switching $\alpha(y)$ for every variable label encountered on that path, including that of the root and that of u. Every node in T labeled with a defining variable will be a leaf.

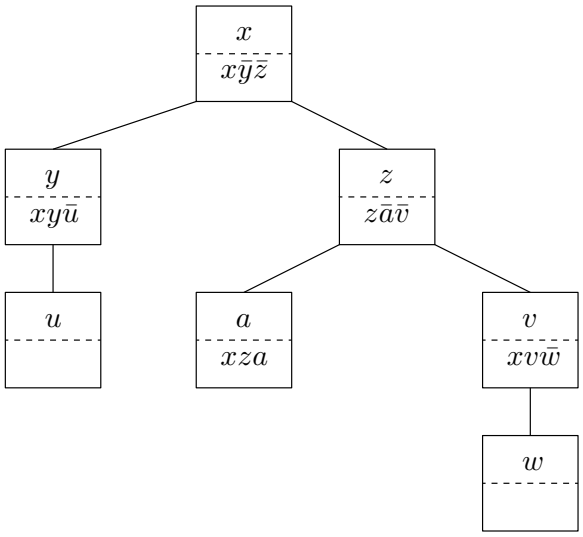
1. Start with T being a single node u and set its variable label $\mathrm{var}(u)$ to be x. Note that α_u is the assignment that sets x to 1 and all other variables to 0. We do not assign it a clause label yet.
2. While there exists a leaf u with $\mathrm{var}(u) \in V_N$:
   (a) Note that α_u differs from α only on the variables appearing on the path from the root to u. All those variables are non-defining, so $\alpha_u \in B$. Since $\alpha_u \neq \alpha$, it is not a satisfying assignment, and therefore some clause $C \in F$ violated by α_u. Set $\mathrm{clause}(u) := C$.
   (b) For every negative literal $\bar{v} \in C$, create a child for u; make v its variable label.

3. A leaf u of T with a defining clause label is called a *safe leaf*.

Suppose node u has clause label $C = y_1 \vee \ldots y_r \vee \bar{z}_1 \vee \bar{z}_s$. From the way T is constructed, it is clear that u has s children with variable labels $z_1, \ldots, z_s$. Since α satisfies C, it holds that $r \geq 1$. Thus, if F is a k-CNF formula then $s \leq k-1$. In other words, every node in T has at most $k-1$ children. Also, $\alpha_u(y_i) = 0$ and thus y_i must be the variable label of an ancestor of u (possibly u itself). Furthermore, $\alpha_u(z_j) = 1$, thus z_j does not occur as the variable label of any ancestor of u. Thus, $\text{var}(u) \neq \text{var}(v)$ whenever u is an ancestor of v. This in turn implies that the construction of T terminates, as T has depth at most n. The following figure gives an example of a critical clause tree for

$$F = (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee \bar{u}) \wedge (z \vee \bar{a} \vee \bar{v}) \wedge (x \vee z \vee a) \wedge (x \vee v \vee \bar{w})$$

and $V_D = \{u, w\}$.



Note that the left-most leaf and the right-most leaf are labeled with a defining variable and thus are safe leaves. The middle leaf, with clause label $x \vee z \vee u$, is different: its assignment $\alpha \oplus x \oplus z \oplus a$ violates some clause, namely $x \vee z \vee a$. However, there are no negative literals in this clause and thus this node has no children.

For a set $W \subseteq V(F) \setminus \{x\}$, let T_W denote the tree obtained from the critical clause tree by removing all nodes u with $\text{var}(u) \in W$ and all its descendants.

Lemma 20. *If T_W contains no safe leaf, then* $\text{clause}(T_W)|_{W=1}$ *implies* $x = 1$.

Here $\text{clause}(T_W)$ is the formula formed by collecting all clause labels in T_W, and $W = 1$ is the restriction obtained by setting to 1 all variables $y \in W$. For example, suppose $W = \{u, v\}$. Then T_W contains the root and its two children, and the left child of its right child; $\text{clause}(T_W) = (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee \bar{u}) \wedge (z \vee \bar{a} \vee \bar{v}) \wedge (x \vee z \vee a)$. Thus, $\text{clause}(T_W)|_{W=1} = (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y) \wedge (z \vee \bar{a}) \wedge (x \vee z \vee a)$, which clearly implies $x = 1$. The proof of the lemma is not difficult and we omit it.

Suppose PPSZ uses the proof heuristic P_D for some $D \in \mathbb{N}$. For $\pi : V(F) \to [0, 1]$ let $W := \{y \in V(F) \mid \pi(y) < \pi(x)\}$. Let E_D be the event (viewed as a set of π's) that T_W contains no safe leaf and $|\text{clause}(T_W)| \leq D$.

Corollary 21. *If E_D happens then $C_x(\pi, \alpha) = 0$.*

Let $\tilde{E}_d$ be the event that T_W contains no safe leaf and no node of depth greater than d. Obviously, if $\tilde{E}_d$ happens then $E_{2^{d+1}}$ happens. Thus, if PPSZ uses the proof heuristic P_D with $D = 2^{d+1}$, then $\mathbb{E}_{\pi \in \Gamma}[C_x(\pi, \alpha)] \leq 1 - \Pr_{\pi \in \Gamma}[\tilde{E}_d]$. Paturi, Pudlák, Saks, and Zane [10] show that $\tilde{E}_d$ is least likely to happen if all variable labels of T are distinct. Since we want to lower bound $\Pr[\tilde{E}_d]$, we will from now on assume that all variable labels are distinct. Since F is a k-CNF formula, every node in T has at most $k - 1$ children. We now extend T to $\tilde{T}$ in the following way: if a node u is not a safe leaf, we create new children until u has exactly $k-1$ children; we repeat the process at the children of u, potentially creating an infinite tree. Thus, in $\tilde{T}$ every node either (1) is a safe leaf, (2) has $k - 1$ children; furthermore, all variable labels in $\tilde{T}$ are distinct. For this, we assume an infinite supply of variable labels. Let $V(\tilde{T})$ be the (possibly infinite) set of all nodes of $\tilde{T}$. We sample $\pi(y) \in \left[0, \frac{k-1}{k-2}\right]$ if y is a safe leaf and $\pi(z) \in [0, 1]$ otherwise. Let W be the set of all nodes $u \in V(\tilde{T})$ for which $\pi(u) < \pi(\text{root})$. As before $\tilde{T}_W$ is the tree obtained from $\tilde{T}$ by deleting all nodes in W and its descendants. Let E_∞ be the event that $\tilde{T}_W$ is finite and has no safe leaf. The following lemma is not difficult to prove.

Lemma 22. $\Pr_{\pi \in \Gamma}[E_D] \geq \Pr[E_\infty] - \epsilon$ *for some ϵ that converges to 0 as D grows.*

Note that $\Pr[E_\infty]$ depends on the structure of $\tilde{T}$. We start by analyzing T^∞, the infinite $(k-1)$-ary tree, which has no safe leaves. Let E'_∞ be the event that T^∞_W is finite. Let us define the conditional probability

$$Q(r) := \Pr[E'_\infty \mid \pi(x) = r] \ .$$

Under the condition $\pi(x) = r$, every node is in W with probability r, and all events $[y \in W]$ are independent. For T_W^∞ to be finite, the following has to happen for every child y of the root: (1) y itself is deleted, which happens with probability r; or (2) y is not deleted, but the subtree rooted at y is itself finite after deletion. The second case happens with probability $(1-r)Q(r)$. Thus, we obtain the equation

$$Q(r) = (r + (1-r)Q(r))^{k-1} \ . \tag{11}$$

Clearly, 1 is a solution of this equation. It follows from the theory of Galton-Watson branching processes that $Q(r)$ is the smallest value in $[0,1]$ satisfying (11). For $r \geq \frac{k-2}{k-1}$ this is 1, for $r < \frac{k-2}{k-1}$ we get $Q(r) < 1$. From here on, it is not difficult to show that $\Pr[E_\infty]$ is minimized if $\tilde{T}$ has no safe leaves. In fact, the $\frac{k-1}{k-2}$ showing up in the definition of Γ is chosen exactly to make this proof go through. Removing the conditioning on $\pi(x) = r$ one obtains

$$\Pr[E_D] \geq \int_0^1 Q(r)dr - \epsilon.$$

Note that $c_k = 1 - \int_0^1 Q(r)dr$ is the probability that the root of the infinite $(k-1)$-ary tree is contained in an infinite component after the random deletion step. Putting everything together, one obtains

Lemma 23. *Let $x \in V_N$. Then $\mathbb{E}_{\pi\in\Gamma}[C_x(\pi,\alpha)] \leq c_k - \epsilon$.*

Recall that ϵ converges to 0 as D grows. Thus, if we use $P_{\omega(1)}$, i.e., P_D for $D = D(n)$ a growing function, ϵ becomes $o(1)$. Plugging this lemma into (10, we obtain

$$\begin{aligned}
\Pr_{\beta\in B,\pi}[\texttt{ppsz}(\beta,\pi,F,P_) = \alpha] &\geq \Pr[\Gamma] \cdot 2^{-\sum_{x\in V_N} \mathbb{E}_{\pi\in\Gamma}[C_x(\pi,\alpha)]} \\
&\geq \left(\frac{k-2}{k-1}\right)^{|V_D|} \cdot 2^{-c_k|V_N|-o(n)} \\
&= 2^{-c_k n - o(n)} \left(\frac{k-2}{k-1} \cdot 2^{c_k}\right)^{|V_D|} \\
&= 2^{-c_k n - o(n)} \left(\frac{k-2}{k-1} \cdot 2^{c_k}\right)^{n - \log_2|B|} \ .
\end{aligned}$$

This is exactly what the lemma states.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il