Chapter 6*

The PPSZ Algorithm

In this chapter we present an improvement of the ppz algorithm. As ppz is called after its inventors, Paturi, Pudlák, and Zane [PPZ99], the improved version, ppsz, is called after Paturi, Pudlák, Saks, and Zane [PPSZ05]. The idea behind the improvement is very beautiful and simple; its analysis is challenging, but very elegant uses a variety of interesting concepts. In this chapter, we analyze the algorithm for the case that the input formula has a *unique* satisfying assignment. This analysis is from [PPSZ05]. If there are multiple satisfying assignments, the analysis surprisingly becomes much more difficult. In fact, it has been open for thirteen years whether one generalize the analysis for the unique case to the general case – without getting a worse running time. This was achieved in 2011 by Timon Hertli [Her11].

The basic idea of the ppz-algorithm is very simple: Given a CNF formula F, choose a variable $x \in \mathrm{vbl}(F)$ uniformly at random. If $\{x\} \in F$ or $\{\bar{x}\} \in F$, set x the obvious way. Otherwise, set x randomly. Then iterate. The analysis of this simple algorithm shows surprising connections to encodings of satisfying assignments and lower bounds on depth-3 boolean circuits. From an algorithmic point of view, here is a simple idea for improvement:

Instead of checking whether $\{x\} \in F$ or $\{\bar{x}\} \in F$, check whether x or $\bar{x}$ can be derived from F in some way. Of course, checking this is again NP-complete and no big help. However, we can check whether x or $\bar{x}$ follows from F in some simple, polynomial-time checkable way. Indeed, $\{x\}$ being a unit clause in F is probably the simplest way how x can follow from F.

Definition 6*.1 *Let* F *be a CNF formula, let* $D \in \mathbb{N}_0$*, and let* u *be a literal. We say that* F implies u *if any assignment* α *satisfying* F *also satisfies*

u. *We say* F D-implies u *if there is some* $G \subseteq F$ *with* $|G| \leq D$ *that implies* u.

In this terminology, in the ppz-algorithm we check whether x is 1-implied in F. The ppsz-algorithm applies an obvious improvement: Check whether x is D-implied for some (large) constant D. This can be done in polynomial time. Simple as this sounds, the analysis of this algorithm turns out to be quite sophisticated. To state the ppsz-algorithm formally, we think of it choosing a random permutation π of $\mathrm{vbl}(F)$ and an assignment $\beta \in \{0,1\}^n$ in advance and then processing the variables according to π and assigning them values according to β, unless they turn out to be D-implied.

F a 3-CNF formula, $\pi = (x_1, \ldots, x_n)$, a permutation on $\mathrm{vbl}(F)$, $\beta \in \{0,1\}^n$, $D \in \mathbb{N}_0$.

POSTCONDITION: returns a satisfying assignment or `failure`

function $\mathrm{ppsz}(F, \pi, \beta, D)$

$\alpha \leftarrow$ the empty assignment ;

$F_1 \leftarrow F$;

for $i \leftarrow 1$ **to** $\mathrm{vbl}(F)$ **do**

  if F_i D-implies x_i **then** $\alpha(x) \leftarrow 1$;

  elseif F_i D-implies $\bar{x}_i$ **then** $\alpha(x) \leftarrow 0$;

  else $\alpha(x) \leftarrow \beta(x)$;

  $F_{i+1} \leftarrow F_i^{[x \mapsto \alpha(x)]}$;

if α satisfies F **then return** α;

else return `failure`;

When does this algorithm succeed in finding a satisfying assignment? For this need the concept of *forced* and *guessed* variables.

Definition 6*.2 *Let* F *be a CNF formula over* n *variables,* $\alpha \in \{0,1\}^n$ *a satisfying assignment, and* $\pi = (x_1, \ldots, x_n)$ *a permutation of* $\mathrm{vbl}(F)$, *and* $D \geq 0$. *A variable* x_i *is called* forced *(with respect to* F, α, π, *and* D*) if* $F^{[x_1 \mapsto \alpha(x_1), \ldots, x_{i-1} \mapsto \alpha(x_{i-1})]}$ D*-implies* x_i *or* $\bar{x}_i$. *Otherwise the variable is called* guessed. *We denote the set of forced (guessed) variables by* $\mathrm{Forced}(F, \alpha, \pi, D)$ *(*$\mathrm{Guessed}(F, \alpha, \pi, D)$*).*

When F, α, π and D are understood from the context, we drop these parameters and just write Forced and Guessed.

Observation 6*.3 *Let* F *be a CNF formula and* α *be a satisfying assignment. Then* $\mathrm{ppsz}(F, \pi, \beta, D)$ *returns* α *if and only if* $\alpha(x) = \beta(x)$ *for all* $x \in \mathrm{Guessed}(F, \alpha, \pi, D)$.

6*.1 Having a Unique Satisfying Assignment

In this section we analyze the special case that F has a *unique* satisfying assignment α. This case is easier to analyze and at the same time we can introduce all the concepts we will need for the general case. Without loss of generality, assume $\alpha = (1, \ldots, 1)$ sets all variables to 1. By the above observation, ppsz returns α if and only if β agrees with α on all variables from $\mathrm{Guessed}(F, \alpha, \pi, D)$. This set is a random object, since it depends on π, which is a random permutation.

$$\Pr_{\beta,\pi}[\text{ppsz returns } \alpha] \quad = \quad \mathrm{E}_{\pi}\left[2^{-|\mathrm{Guessed}(F,\alpha,\pi,D)|}\right]$$

This expression is rather difficult to work with. It would be easier to bound $\mathrm{E}\left[|\mathrm{Guessed}(F, \alpha, \pi, D)|\right]$, for example. Luckily, we have Jensen's inequality, which in this case states that for a random variable X, it holds that

$$\mathrm{E}\left[2^{X}\right] \geq 2^{\mathrm{E}[X]}\,, \tag{6*.1}$$

since the function $x \mapsto 2^x$ is a convex function. With this, we obtain

$$\Pr_{\beta,\pi}[\text{ppsz returns } \alpha] \quad \geq \quad 2^{-\mathrm{E}_{\pi}[|\mathrm{Guessed}(F,\alpha,\pi,D)|]} \tag{6*.2}$$

The right-hand side here is a much nicer expression, since by linearity of expectation it equals the sum of the probabilities

$$\Pr[x \in \mathrm{Guessed}(F, \alpha, \pi, D)]\,. \tag{6*.3}$$

6*.1.1 Building a Tree

We will now try to bound $\Pr[x \in \mathrm{Guessed}(F, \alpha, \pi, D)]$ from above. In this section, we confine ourselves to k-CNF formulas F having a unique satisfying assignment, which we assume to be $\alpha = (1, \ldots, 1)$. Fix some $x \in \mathrm{vbl}(F)$. We construct a binary tree T_x, called the *critical clause tree of* x that in some way represents the multiple ways x can become D-implied in ppsz.

Definition 6*.4 *Let* F *be a* $(\leq k)$*-CNF formula. A* clause tree T *is a tree with the following properties:*

- *Every node has at most* $k-1$ *children.*
- *Every node* $u \in V(T)$ *is labeled with a variable from* $\mathrm{vbl}(()F)$*, which we denote by* $\text{var-label}(u)$*, and additionally with a clause* $C \in F$*, denoted by* $\text{clause-label}(u)$*.*
- *If* u *is a proper ancestor of* v*, then* $\text{var-label}(u) \neq \text{var-label}(v)$*.*

We will define a clause tree T_x for each $x \in \mathrm{vbl}(F)$. See Figures 6*.1– 6*.5 for an illustration.

1. Start with T_x consisting of a single root. This root has variable label x, and does not have a clause label yet.
2. As long as there is a leaf $u \in V(T)$ that does not yet have a clause label, do the following:
   (a) Define $U := \{\text{var-label}(v) \mid v \in V(T) \text{ is an ancestor of } u \text{ in } T\}$, where *ancestor* includes u itself and the root.
   (b) Define the assignment β as
   $$\beta : \mathrm{vbl}(F) \to \{0,1\}, \quad z \mapsto \begin{cases} 0 & \text{if } z \in U, \\ 1 & \text{otherwise.} \end{cases}$$
   (c) Let $C \in F$ be a clause not satisfied by β. Since $\beta \neq \alpha$ and α is the unique satisfying assignment, such a clause exists. Set $\text{clause-label}(u) := C$.
   (d) For each negative literal $\bar{y} \in C$, create a new leaf, label it with y, and attach it to u as a child. The new leaf does not yet have a clause label.

Note that every clause in F has at most $k-1$ negative literals, and thus in step (d) we append at most $k-1$ children. Suppose v is an ancestor of u and $\text{var-label}(v) = y$. Since $\beta(y) = 0$, we conclude that $\bar{y} \notin C$. Therefore, no node has the same label as one of its ancestors. This also implies that the height of the tree cannot exceed n, thus the process terminates. We denote the resulting clause tree by T_x.

Figure 6*.1: Consider the formula $F = \{\{x, \bar{y}, \bar{z}\}, \{x, \bar{v}, \bar{w}\}, \{x, y, v\}, \{x, z, \bar{a}\}, \dots\}$. We begin the construction of T_x with a tree consisting of a root. It is labeled with x, but does not yet have a clause label. We consider the assignment $\alpha[x \mapsto 0]$ and observe that this leaves $\{x, \bar{y}, \bar{z}\}$ unsatisfied. We attach two new leaves to the root, labeled with y and z, respectively.

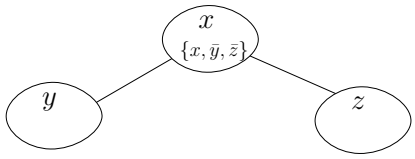

Figure 6*.2: We take the leaf labeled with z. The assignment $\alpha[x, z \mapsto 0]$ leaves the clause $\{x, z, \bar{a}\}$ unsatisfied. Thus, we attach one new child to z and label it with a.

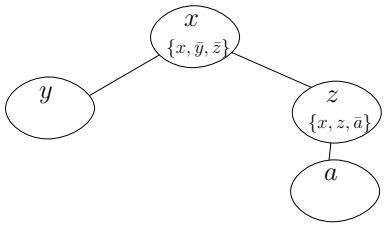

Figure 6*.3: We take the leaf labeled with y. The assignment $\alpha[x, y \mapsto 0]$ leaves the clause $\{x, \bar{v}, \bar{w}\}$ unsatisfied. Thus, we attach two new children to y.

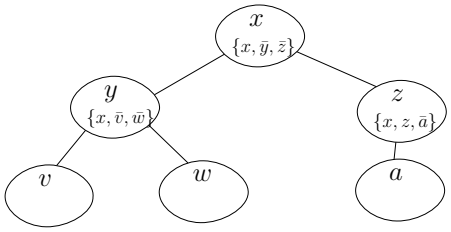

Figure 6*.4: We pick the leftmost leaf. The assignment $\alpha[x, y, v \mapsto 0]$ leaves the clause $\{x, y, v\}$ unsatisfied. We do not attach further leaves here.

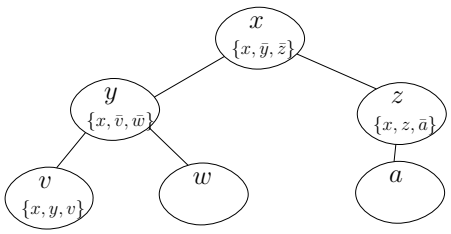

Figure 6*.5: The tree now has 6 nodes. Two leaves are still do not have a clause label, thus the process is not finished yet.

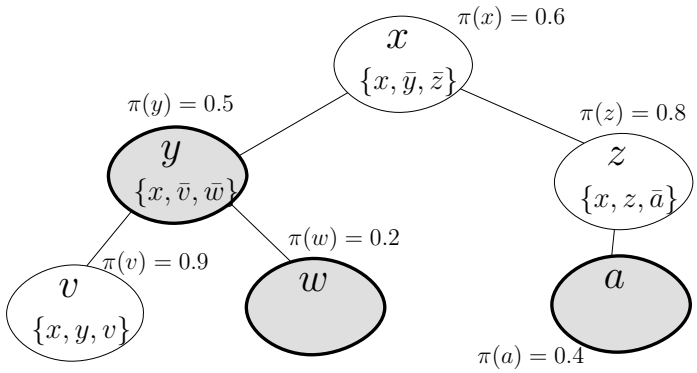

Figure 6*.6: Illustration of Lemma 6*.5. Nodes u with $\pi(u) < \pi(x)$ are shown gray. Two nodes are reachable, those labeled with x and z. Their clause labels yield the subformula $\{\{x, \bar{y}, \bar{z}\}, \{x, z, \bar{a}\}\}$. This formula implies the clause $\{x, \bar{y}, \bar{a}\}$ (which can be seen by resolution, for example). Therefore, every assignment satisfying this formula and setting all Gray variables to 1 must set x to 1.

From now on, we do not think about π as a permutation of $\mathrm{vbl}(F)$, but rather as a function $\mathrm{vbl}(F) \to [0,1]$. If the values $\pi(x)$ are chosen independently and uniformly at random from $[0,1]$ for each $x \in \mathrm{vbl}(F)$, then with probability 1, π is injective, and by sorting the values $\pi(x)$ we obtain a uniformly distributed permutation of $\mathrm{vbl}(F)$. Note that π also defines a function $V(T) \to [0,1]$ via $\pi(u) := \pi(\text{var-label}(u))$.

Let $\gamma \in [0,1]$ and T be a clause tree. We call a node *reachable* if there exists a path $\text{root} = v_0, v_1, \ldots, v_m = u$ in T such that $\pi(v_i) \geq \gamma$ for all $1 \leq i \leq m$. We let $\mathrm{Reachable}(T, \gamma, \pi)$ be the set of all reachable nodes.

Lemma 6*.5 (Clause Tree Lemma) *If* $|\mathrm{Reachable}(T_x, \pi(x), \pi)| \leq D$, *then* $x \in \mathrm{Forced}(F, \alpha, \pi, D)$.

Proof . Let α' be the restriction of α to the variables y with $\pi(y) < \pi(x)$. By definition, x is forced if there is a formula $F' \subseteq F^{[\alpha']}$ with $|F'| \leq D$ that implies x. Let $G := \text{clause-label}(\mathrm{Reachable}(T_x, \pi(x), \pi))$, i.e., the subformula of F consisting of all clause labels of reachable nodes in T_x. Clearly $|G| \leq D$. We claim that $G^{[\alpha']}$ implies x.

Suppose, for the sake of contradiction, that $G^{[\alpha']}$ does not imply x. Then there is some assignment $\beta : \mathrm{vbl}(G) \to \{0,1\}$ such that (i) β satisfies G, (ii)

$\beta(y) = 1$ for all $y \in \text{vbl}(G)$ for which $\pi(y) < \pi(x)$, (iii) $\beta(x) = 0$.

Choose a maximal path in T_x starting at the root and containing only nodes v such that $\beta(\text{var-label}v) = 0$. Since $\beta(x) = 0$, this path is non-empty. Let u be its endpoint. The node u is reachable. All children of u are labeled with some variable z for which $\beta(z) = 1$, and all ancestors with some variable y for which $\beta(y) = 0$. One checks that clause-label(u) is unsatisfied by β, a contradiction. □

6*.1.2 How to Bound $\Pr[x \in \text{Forced}(F, \alpha, \pi, D)]$

For this section, fix a variable x By Lemma 6*.5 we know that

$$\Pr[x \in \text{Forced}(F, \alpha, \pi, D)] \geq \Pr[|\text{Reachable}(T_x, \pi(x), \pi)| \leq D] \ .$$

From now on we can forget the clause labels in our tree and focus only on its structure and variable labels. For a tree T and $\gamma \in [0, 1]$ let $h(T, \gamma)$ be the number of nodes in a longest path in T starting at the node and containing only reachable nodes. Hence $h(T, \gamma) = 0$ if $\pi(\text{root}(T)) < \gamma$.

Observation 6*.6 $|\text{Reachable}(T, \gamma, \pi)| \leq (k-1)^{h(T,\gamma)}$, *and therefore, for* $d := \log_{k-1} D$, $\Pr[|\text{Reachable}(T, \gamma, \pi)| \leq D] \geq \Pr[h(T, \gamma) \leq d]$.

Definition 6*.7 *For a clause tree* T, $\gamma \in [0, 1]$ *and* $d \geq 2$ *we define*

$$P_\gamma(T, d) := \Pr[h(T, \gamma) \leq d \mid \pi(\text{root}(T)) \geq \gamma] \ .$$

For a tree T, let T' be the tree we obtain by replacing each variable label in T by a new distinct variable.

Lemma 6*.8 (Distinct Label Lemma) *Let* T *be a clause tree,* $\gamma \in [0, 1]$, *and* $d \in \mathbf{N}_0$, *and* $\gamma \in [0, 1]$. *Then* $P_\gamma(T, d) \geq P_\gamma(T', d)$.

Proof . The proof consists of two crucial facts, plus a straightforward induction. The one fact is that if u is a proper ancestor in T of v, then var-label(u) $\neq$ var-label(v), and thus the event $[\pi(u) < \gamma]$ is independent of everything happening in u's subtree. The second fact is that two monotone boolean functions are positively correlated:

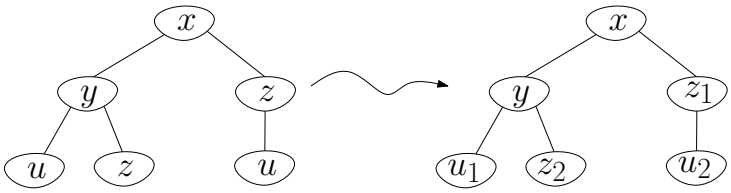

Figure 6*.7: Illustration of Lemma 6*.8: The left tree T, with non-unique labels, is at least as good as the right tree T' with distinct labels.

Theorem 6*.9 (FKG Inequality [FKG71]) *Let* $f, g : \{0,1\}^n \to \mathbb{R}$ *be two monotonically increasing boolean functions. Sample* $x \in \{0,1\}^n$ *by setting each* x_i *to* 1 *with some probability* $p_i \in [0,1]$*, independently. Then*

$$\Pr[f(x) = 1 \wedge g(x) = 1] \geq \Pr[f(x) = 1] \cdot \Pr[g(x) = 1] \ .$$

We are ready to prove the lemma by induction on d. Let u be the root of T and u' the root of T'. If $d = 0$, then $P(T, 0) = 0$, since $\pi(u) \geq \gamma$. The same holds for T'.

For the induction step, let $v_1, \ldots, v_\ell$ be the children of u in T, and $v'_1, \ldots, v'_\ell$ the children of u' in T'. Let $T_1, \ldots, T_\ell$ and $T'_1, \ldots, T'_\ell$ be the respective subtrees. Clearly $h(T, \gamma) \leq d$ if and only if for each $1 \leq i \leq \ell$, either (i) $\pi(v_i) < \gamma$ or (ii) $\pi(v_i) \geq \gamma$ and $h(T_i) \leq d - 1$. We calculate

$$\begin{aligned}
P(T, d) &= \Pr\left[\bigwedge_{i=1}^{\ell} h(T_i, \gamma) \leq d - 1\right] \\
&\geq \prod_{i=1}^{\ell} \Pr\left[h(T_i, \gamma) \leq d - 1\right] && (6*.4) \\
&= \prod_{i=1}^{\ell} (\gamma + (1 - \gamma) \cdot P(T_i, d - 1)) && (6*.5) \\
&\geq \prod_{i=1}^{\ell} (\gamma + (1 - \gamma) \cdot P(T'_i, d - 1)) && (6*.6) \\
&= \Pr\left[\bigwedge_{i=1}^{\ell} h(T'_i, \gamma) \leq d - 1\right] && (6*.7) \\
&= \Pr[h(T', \gamma) \leq d] \ . && (6*.8)
\end{aligned}$$

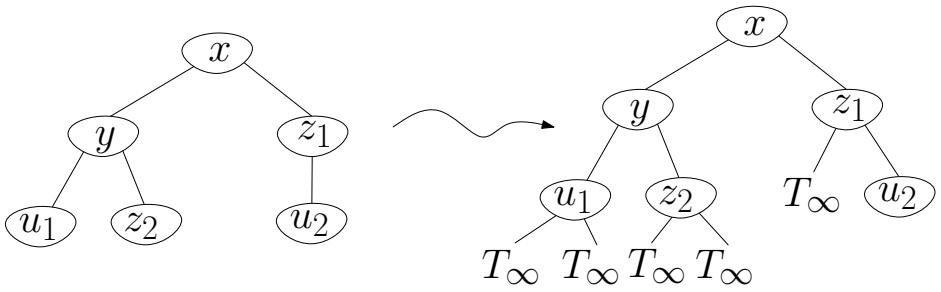

Figure 6*.8: Making the tree infinite can only increase Reachable(T, γ, π).

Here, (6*.4) follows from the FKG Inequality, since the events $[h(T_i, \gamma) \leq d-1]$ are monotone boolean functions in the events $[\pi(w) < \gamma]$. Inequality (6*.6) follows from induction, and (6*.7) holds since the events $[h(T_i', \gamma) \leq d-1]$ are independent, since all variable labels in T_i' are distinct. □

Let us condition the event $[x \in \text{Forced}(F, \alpha, \pi, D)]$ on $\pi(x) = \gamma$. For $d = \log_{k-1} D$ it holds that

$$\begin{aligned} \Pr[x \in \text{Forced}(F, \alpha, \pi, D) \mid \pi(x) = \gamma] &\geq \Pr[h(T_x, \gamma) \leq d \mid \pi(x) = \gamma] \\ &= \Pr[h(T_x, \gamma) \leq d \mid \pi(x) \geq \gamma] \\ &= P(T_x, d) \geq P(T_x', d)\ . \end{aligned}$$

Next, we extend T_x' to an infinite tree T_∞: Attach several infinite $(k-1)$-ary tree to every node in T_x' having fewer than $k-1$ children. Label each new node with a new variable. This gives an infinite $(k-1)$-ary tree T_∞, whose root is labeled with x.

Observation 6*.10 Reachable$(T_x', \gamma, \pi) \subseteq$ Reachable(T_∞, γ, π) *and therefore* $P(T_x', d) \geq P(T_\infty, d)$.

Note that $P(T_\infty, d)$ is some number depending on γ, k, and d, but not depending on the input formula F. Since T_∞ is infinite, there is a possibility that $h(T_\infty, \gamma)$ is unbounded.

Exercise 6*.1 *Show that if* $h(T_\infty, \gamma) \geq d$ *for all* $d \in \mathbf{N}$, *i.e.*, T_∞ *contains* arbitrarily long *paths of reachable nodes starting at the root, then* T_∞ *contains an* infinite *path of reachable nodes starting at the root.*

We write $h(T_\infty, \gamma) = \infty$ as a shorthand for "there exist arbitrarily long paths of reachable nodes starting at the root", and $h(T_\infty, \gamma) < \infty$ if there are no such paths. We define

$$P_\gamma(T_\infty) := \Pr[h(T_\infty, \gamma) < \infty] \ .$$

The following lemma is sounds deceptively obvious, but is not entirely trivial to prove.

Lemma 6*.11 *For each $\gamma \in [0,1]$, it holds that*

$$\lim_{d\to\infty} P_\gamma(T_\infty, d) = P_\gamma(T_\infty) \ .$$

This means that there are numbers $\epsilon_{\gamma,d}$ such that for every fixed $\gamma \in [0,1]$, it holds that $P_\gamma(T_\infty, d) \geq P_\gamma(T_\infty) - \epsilon_{\gamma,d}$ and $\lim_{d\to\infty} \epsilon_{\gamma,d} = 0$. Let us combine everything we have learned so far:

$$\begin{aligned} & \Pr[x \in \text{Forced}(F, \alpha, \pi, D) \mid \pi(x) = \gamma] \\ \geq \ & P_\gamma(T_\infty, d) \\ \geq \ & P_\gamma(T_\infty) - \epsilon_{\gamma,d} \ , \end{aligned}$$

and by the law of total probability, we conclude that

$$\begin{aligned} & \Pr[x \in \text{Forced}(F, \alpha, \pi, D)] \\ \geq \ & \int_0^1 P_\gamma(T_\infty) d\gamma - \int_0^1 \epsilon_{\gamma,d} d\gamma \ . \end{aligned}$$

If $D = D(n)$ is some slowly growing function, then $d = \log_2 D$ grows, too. One has to check that if $\epsilon_{\gamma,d}$ goes to 0 for every fixed γ, then also $\int_0^1 \epsilon_{\gamma,d}\ d\gamma$ goes to 0. We write $R_k := \int_0^1 P_\gamma(T_\infty) d\gamma$. This is a number that only depends on k. We have

$$\Pr[x \in \text{Forced}(F, \alpha, \pi, D)] \geq R_k - o(1) \ ,$$

$$\Pr[\text{ppsz}(F, \beta, \pi, D) = \alpha] \geq 2^{-(1-R_k)(n+o(n))} \quad .$$

We can find an equation for $P_\gamma(T_\infty)$, much like in the proof of Lemma 6*.8. Let u be the root of T and $v_1, \ldots, v_{k-1}$ be its children, and let $T_1, \ldots, T_{k-1}$ be the corresponding subtrees. The crucial observation is that all T_i are

isomorphic to T_∞ itself, and therefore $P_\gamma(T_\infty) = P_\gamma(T_i)$. On the other hand, observe that $h(T_\infty, \gamma) < \infty$ if and only if for every $1 \leq i \leq k-1$ either (i) $\pi(v_i) < \gamma$ or (ii) $\pi(v_i) \geq \gamma$ and $h(T_i, \gamma) < \infty$. Therefore

$$P_\gamma(T_\infty) \quad = \quad (\gamma + (1-\gamma)P_\gamma(T_\infty))^{k-1} \ . \tag{6*.9}$$

For $k = 3$, (6*.9) has the two solutions $P_\gamma(T_\infty) = 1$ and $P_\gamma(T_\infty) = \gamma^2/(1-\gamma)^2$. Without knowing which one is the true value, we can at least say that $P_\gamma(T_\infty) \geq \min(1, \gamma^2/(1-\gamma)^2$ (in fact, we will show later that this holds with equality) and thus

$$\int_0^1 P_\gamma(T_\infty) = \int_0^1 \min\left(1, \frac{\gamma^2}{(1-\gamma)^2}\right) = 2 - 2\ln 2 \geq 0.6137 \ .$$

Thus

$$\Pr[\mathrm{ppsz}(F, \beta, \pi, D) = \alpha] \geq 2^{-0.3863n - o(n)} \quad \geq \Omega\left(1.3071^{-n}\right) \ .$$

Theorem 6*.12 *Let D be a slowly growing function. Then for any 3-CNF formula F over n variables that has a unique satisfying assignment α, $\mathrm{ppsz}(F, \pi, \beta, D)$ returns α with probability $\Omega(1.3071^{-n})$.*

Bibliography

[FKG71] C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Comm. Math. Phys.*, 22:89–103, 1971.

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

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

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