The PPSZ Algorithm for Constraint Satisfaction Problems on More Than Two Colors

Timon Hertli[1], Isabelle Hurbain[1], Sebastian Millius[1], Robin A. Moser[1], Dominik Scheder[2], and May Szedlák[1]

[1] ETH Zürich
[2] Shanghai Jiaotong University

Abstract. The PPSZ algorithm (Paturi et al., FOCS 1998) is the fastest known algorithm for k-SAT. We show how to extend the algorithm and its analysis to (d, k)-Clause Satisfaction Problems where each variable ranges over d different values. Given an input instance with a unique satisfying assignment, the resulting algorithm is the fastest known algorithm for (d, k)-CSP except when (d, k) is $(3, 2)$ or $(4, 2)$. For the general case of multiple satisfying assignments, our algorithm is the fastest known for all $k \geq 4$.

1 Introduction

In its full generality, the Constraint Satisfaction Problem is NP-complete, so most researchers believe that we will never find an efficient algorithm for it. Worse, even getting a substantial edge over trivial exhaustive search is deemed unlikely by most in the community. Far from despairing, people have tried several routes around this: finding heuristics that work well in practice [3]; designing algorithms that are fast as long as a certain complexity measure of the instance is small [14,1]; coming up with *moderately* exponential algorithms.

The study of moderately exponential algorithms has been especially fruitful in two areas: the study of algorithms for k-satisfiability (short k-SAT) and graph colorability. For example, the algorithm PPSZ solves 3-SAT in $O(1.308^n)$ instead of the trivial 2^n; Beigel and Eppstein [2] show how to solve 3-colorability in time $O(1.3289^n)$ instead of the trivial 3^n; Björklund and Husfeldt [4] solve k-colorability in time $O(2^n \text{poly}(n))$ instead of the trivial k^n.

We focus on the general constraint satisfaction problem where every variable takes on a value in $[d] := \{1, \ldots, d\}$ and the only structural restriction is that each constraint may depend on at most k variables. Every constraint can be written as the conjunction of at most d^k *clauses*, i.e., disjunctive constraints like $(x_1 \neq 3 \vee x_2 \neq 4 \vee x_1 \neq 2)$. Since d and k are considered constant values, we can re-write an instance as a conjunction of clauses. We call the resulting formula a (d, k)- *clause satisfaction formula* and the corresponding decision problem the *clause satisfaction problem*. We abbreviate both by (d, k)-ClSP. Note that k-SAT is the same as $(2, k)$-ClSP and d-colorability is a special case of $(2, d)$-ClSP.

In this paper we generalize the PPSZ k-SAT algorithm [10] and Hertli's analysis [7] to (d, k)-ClSP. While it is rather straightforward to adapt the algorithm

to handle values $d \geq 3$, analyzing its running time is much more challenging than in the Boolean ($d = 2$) case. This is in contrast to Schöning's random walk algorithm [13], where both algorithm and analysis generalize easily to $d \geq 3$.

1.1 Which Running Time Can We Expect

We measure the running time of an algorithm in terms of n, the number of variables in the input formula F. Since all algorithms known so far have exponential running time, and thus sub-exponential terms do not really influence the running time, we use the $O^*(\cdot)$ notation: $O^*(f(n))$ means $f(n) \cdot 2^{o(n)}$. Clearly we can solve (d, k)-ClSP by simply trying out all d^n possible assignments to the n variables. Checking whether a concrete assignment satisfies the formula can be done in polynomial time, so this yields a running time $O^*(d^n)$, a baseline against which we measure our algorithms.

Under the assumption that $P \neq NP$ we will not find a polynomial time algorithm for (d, k)-CSP (expect for the trivial case $d = 1$ and for $(d, k) = (2, 2)$, which is 2-SAT). The Exponential Time Hypothesis [8] states that there is some $c > 0$ such that every (randomized) 3-SAT algorithm runs in time at least $\Omega\left(2^{cn}\right)$. Thus, we cannot expect running times like $O(d^{\sqrt{n}})$ for (d, k)-ClSP. Indeed, a more sophisticated reduction by Traxler [15] actually shows that under the Exponential Time Hypothesis, every algorithm for $(d, 2)$-ClSP takes time at least $\Omega\left(d^{cn}\right)$, for some $c > 0$. In other words, (d, k)-ClSP becomes strictly more complex as d increases, even for $k = 2$. This stands in contrast to d-Colorability, which can be solved in $O^*(2^n)$ time [4], for every d. Thus, under the Exponential Time Hypothesis, $(d, 2)$-ClSP is strictly more complex than d-Colorability.

1.2 Previous Results

For k-SAT, the currently fastest known (randomized) algorithm is the PPSZ algorithm by Paturi, Pudlák, Saks and Zane [10]. For instances with a unique satisfying assignment they give an elegant running time analysis. We call this case *UniqueSAT* (or *UniqueClSP* for $d \geq 3$). For the general case (if the instance has multiple satisfying assignments), the analysis becomes much more difficult, and it took over ten years until Hertli [7] showed how to obtain the UniqueSAT time bound in the general case as well.

There are several moderately exponential algorithms for (d, k)-ClSP. Schöning's random walk algorithm [13] solves (d, k)-ClSP in time $O^*\left(\left(\frac{d(k-1)}{k}\right)^n\right)$. Beigel and Eppstein gave an algorithm for $(d, 2)$-ClSP running in time $\mathcal{O}((0.4518d)^n)$ for $d > 3$. Feder and Motwani [5] give an $(d, 2)$-ClSP algorithm based on the PPZ algorithm [11], the predecessor of the PPSZ algorithm, improving on the algorithm by Beigel and Eppstein for large d. Li, Li, Liu, and Xu [9] generalized this to (d, k)-ClSP, but with a sub-optimal weaker analysis. Scheder [12] showed how to use the full power of PPZ for (d, k)-ClSP.

A generic technique for turning any k-SAT algorithm into a (d,k)-ClSP is *downsampling*: for each variable x in F randomly forbid all but 2 colors. The resulting instance F' is a $(2,k)$-ClSP and can be solved by any off-the-shelf k-SAT algorithm A. We call this algorithm "downsampling + A". Note that if F is unsatisfiable then F' is; if F is satisfiable then F' is satisfiable with probability at least $(2/d)^n$.

1.3 Our Contribution

We generalize PPSZ to (d,k)-ClSP and analyze its running time. Our upper bound for UniqueClSP is of the form $O^*\left(d^{S_{d,k}n}\right)$ where $S_{d,k} < 1$ is some constant depending on the number of colors d and the arity k of the constraints. We have a complicated but more or less explicit formula for $S_{d,k}$ (involving a sum and an integral). However, there is an intuitive explanation "what $S_{d,k}$ is":

Consider the following random experiment: Let T be an infinite rooted tree in which every even-level vertex (this includes the root, which has level 0) has $k-1$ children, and every odd-level vertex has $d-1$ children (there are no leafs). Take $d-1$ disjoint copies of T, choose a value $p \in [0,1]$ uniformly at random and delete each odd-level vertex of the $d-1$ trees with probability p, independently. Let Y be the number of trees in which this deletion still leaves an infinite path starting at the root. Obviously, Y is a random variable and $0 \leq Y \leq d-1$. Define $S_{d,k} := \mathbf{E}\left[\log_d(1+Y)\right]$. We will devote a section in the appendix to the computation of $S_{d,k}$.

Theorem 1.1. *There exists randomized algorithm for Unique-(d,k)-ClSP with one-sided error that runs in time $\mathcal{O}^*(d^{S_{d,k}n})$.*

In the general case (when the input formula may have multiple satisfying assignments) we fail to match this running time for $k = 2, 3$. This failure may well be an artifact of our analysis and not reflect the true success probability of PPSZ. Let $G_{d,k} := \max\{S_{d,k}, 1 - \frac{1}{2\ln(d)}\}$.

Theorem 1.2. *There exists a randomized algorithm for (d,k)-ClSP with one-sided error that runs in time $O^*(d^{G_{d,k}n})$.*

It turns out that $G_{d,k} = S_{d,k}$ for $k \geq 4$, so for $k \geq 4$ our analysis yields the same performance bounds for the unique and the general case:

Lemma 1.3. *If $k \geq 4$ then $S_{d,k} \geq 1 - \frac{1}{2\ln(d)}$ and therefore $S_{d,k} = G_{d,k}$.*

In the general case, i.e., if F may have multiple satisfying assignments, we also solve an open problem of [7]: Hertli [7] made a slight (and natural) change to PPSZ to make his analysis go through. However, he conjectured that is modification is unnecessary, i.e., the same performance bound would hold for PPSZ as originally stated. We show that this is indeed the case. Furthermore, our proof actually gives a bound on the probability that *a specific satisfying assignment* α is returned, whereas [7] only gave a bound that *some* satisfying assignment is returned.

For certain values of d and k Theorem 1.2 gives a *worse* running time than the analysis of PPZ in [12]. Since PPSZ is at least as good as PPZ, we know that for certain parameters Theorem 1.2 gives a suboptimal bound. Still, in the tables below we list the numbers as given by Theorem 1.2 so that the reader can see immediately where this paper improves currently known best bounds.

d	k	PPSZ Unique	PPSZ General	BE [2]	FM [5]	Downsampling+ 2-SAT
3	2	1.434	1.820	**1.356**	1.5	1.5
4	2	1.849	2.427	**1.808**	2	2
5	2	**2.254**	3.033	**2.259**	2.5	2.5
6	2	**2.652**	3.640	**2.711**	2.994	3
10	2	**4.208**	6.066	**4.518**	4.529	5
15	2	**6.115**	9.098	6.777	**6.424**	7.5

Table 1. Constants c so that the algorithm for $(d, 2)$-ClSP runs in time $c^{n+o(n)}$

Asymptotics. We want to gauge the performance of several (d, k)-ClSP algorithms for large d. For this, we define the *savings* of an algorithm to be the largest c such that it solves (d, k)-ClSP in time $O^*\left(\frac{d^n}{2^{cn}}\right)$.

Theorem 1.4. *For $k \geq 2$ and large d the savings of PPSZ for (d, k)-ClSP converge to $\log_2(e)(1 - S_{2,k})$, and $1 - S_{2,k} = -\int_0^1 \ln(1 - r^{k-1})dr$.*

This means the savings for large d are a factor $\log_2(e) \approx 1.44$ larger than the savings for k-SAT. It should be mentioned that for large d the advantage of PPSZ over PPZ vanishes, i.e., their savings converge, for each fixed k. We compare the savings of several algorithms in Table 3.

1.4 Notation

We adapt the notational framework as used in [16]. Let V be a finite set of variables, each of which takes values in $[d] := \{1, \ldots, d\}$. A *literal* over $x \in V$ is of the form $(x \neq c)$ for $c \in \mathbb{N}$. A *clause* over V is a disjunction (OR) of finite set of literals over pairwise distinct variables from V. A formula F over V is a conjunction (AND) of clauses over V. It is sometimes convenient to view F as a *set* of clauses. By $\mathrm{vbl}(F)$ we denote the set of variables appearing in F.

A formula F is a (d, k)-ClSP if the variables can take on d values and every clause has at most k literals. We also write (d, k)-ClSP to denote the satisfiability decision problem on (d, k)-ClSP formulas.

A *assignment* on V is a function $\alpha : V \to [d]$. It satisfies the literal $(x \neq c)$ if $\alpha(x) \neq c$; it satisfies a clause if it satisfies at least one literal therein; finally, it satisfies a formula if it satisfies all its clauses.

A *partial assignment* α on V is a partial function $V \to [d]$. It is convenient to view α as a certain $(d, 1)$-CSP over V: for example $(x_1 = c_1) \wedge (x_2 = c_2)$ is

d	k	PPSZ Unique	PPSZ General	PPZ [12]	Downsampling+PPSZ
3	3	**1.901**	**1.901**	2.162	1.961
4	3	**2.479**	**2.479**	2.729	2.615
5	3	**3.049**	**3.049**	3.291	3.268
6	3	**3.614**	**3.640**	3.850	3.922
7	3	**4.175**	**4.246**	4.407	4.575
8	3	**4.733**	**4.853**	4.963	5.229
9	3	**5.289**	**5.459**	5.516	5.882
10	3	**5.844**	**6.066**	6.069	6.536
11	3	**6.397**	6.672	**6.621**	7.189
15	3	**8.602**	9.098	**8.821**	9.803
3	4	**2.153**	**2.153**	2.351	2.204
4	4	**2.823**	**2.823**	3.014	2.938
5	4	**3.487**	**3.487**	3.672	3.673
10	4	**6.761**	**6.761**	6.935	7.345
15	4	**10.006**	**10.006**	10.176	11.018
3	5	**2.310**	**2.310**	2.471	2.355
4	5	**3.040**	**3.040**	3.195	3.139
5	5	**3.764**	**3.764**	3.915	3.924
10	5	**7.348**	**7.348**	7.490	7.848
15	5	**10.906**	**10.906**	11.045	11.771

Table 2. Constants c so that the algorithm for (d,k)-ClSP runs in time $c^{n+o(n)}$

the partial assignment that sets x_1 to c_1 and x_2 to c_2. Two partial assignments α, β over V are *compatible* if they agree whereever they are defined; equivalently, if $\alpha \wedge \beta$, viewed as a $(d,1)$-CSP, is satisfiable. If α is a partial assignment on V then $\mathcal{U}_\alpha$ is the set of variables in V on which α is not defined. We denote by $\alpha[x=c]$ the (partial) assignment that sets x to c and agrees with α elsewhere.

By $\models$ we denote usual logical implication. That is, for two formulas F, G over a variable set V, the expression $F \models G$ means that every total assignment α that satisfies F also satisfies G.

By Unique (d,k)-ClSP we denote the promise problem of deciding whether a (d,k)-ClSP has exactly one or no satisfying assignment.

2 The PPSZ Algorithm

Definition 2.1 (D-implication). *Let F be a satisfiable ClSP formula over V, α_0 a partial assignment, and $D \in \mathbb{N}$. We say that (F, α_0) D-implies the literal $(x \neq c)$ and write $(F, \alpha_0) \vDash_D (x \neq c)$ if there is a subset G of F with $|G| \leq D$ such that $G \wedge \alpha_0$ implies $(x \neq c)$.*

Whether $(F, \alpha_0) \models_D (x \neq c)$ holds or not can be checked in time $O(|F|^D \cdot \text{poly}(n))$. If D is constant this is polynomial. If D is sufficiently slowly growing this is subexponential.

k	PPSZ (and PPZ)	Downsampling+PPSZ	Schöning
general k	$\log_2(e)(1-S_{2,k})$	$1-S_{2,k}$	$\log_2\left(\frac{k}{k-1}\right)$
2	1.44	1	1
3	0.885	0.613	0.585
4	0.642	0.445	0.415
5	0.504	0.349	0.322
$k\to\infty$	$\frac{\pi^2\log_2(e)}{6k}\approx\frac{2.371}{k}$	$\frac{\pi^2}{6k}\approx\frac{1.644}{k}$	$\frac{\log_2(e)}{k}\approx\frac{1.44}{k}$

Table 3. The savings of several (d,k)-ClSP algorithms. For PPSZ and PPZ the savings hold for $d\to\infty$. The savings of downsampling+PPSZ and of Schöning do not depend on d.

Definition 2.2 (Eligible values). *Let F be a satisfiable ClSP formula over V, α_0 a partial assignment, and $x\in\mathcal{U}_{\alpha_0}$ (i.e. an unassigned variable). Then*

$$\mathcal{A}(x,\alpha_0):=\{c\in[d]\mid(F,\alpha_0)\nvdash_D(x\neq c)\}\ .$$

That is, $\mathcal{A}(x,\alpha_0)$ is the set of colors not ruled out by D-implication.

Note that $\mathcal{A}(x,\alpha_0)$ also depends on F and D. However, F and D will not change throughout the analysis, so we will assume from now on that they are clear from the context.

Let us describe PPSZ. Given a ClSP F, it starts with the empty assignment $\alpha_0=\emptyset$ and attempts to incrementally add variables to it, hoping that eventually α_0 becomes a satisfying (total) assignment. To achieve this, PPSZ chooses a uniformly random permutation π of V and iterates through V in the order dictated by π. When considering some $x\in V$ it computes $\mathcal{A}(x,\alpha_0)$. If this is empty then $F\wedge\alpha_0$ is unsatisfiable and PPSZ declares failure. Otherwise, it chooses some eligible color $c\in\mathcal{A}(x,\alpha_0)$ uniformly at random, adds $(x=c)$ to α_0, and continues. Below we give a pseudo-code for PPSZ. Our pseudo-code is recursive rather than iterative because this is more convenient for the analysis of the general (multiple satisfying assignments) case.

Algorithm 1 Top-Level-PPSZ(ClSP formula F)

Choose π u.a.r. from all permutations of $V(F)$.
Let α_0 be the empty assignment.
return PPSZ(F, π, α_0)

Note that $\mathcal{A}(x,\alpha_0)$ is the set of values that are "currently eligible for x". Now suppose α_0 is compatible with some satisfying assignment α, and the next assignment steps of PPSZ are all according to α. We are actually interested how $\mathcal{A}(x,\alpha_1)$ will look where α_1 is the "future" partial assignment just before x is processed. This motivates the following (recursive) definition.

Algorithm 2 PPSZ(F, π, α_0)

if α_0 is a total assignment **then**
  return α_0 if it satisfies F, else `failure`
end if
$x \leftarrow$ first variable of $\mathcal{U}_{\alpha_0}$ according to π
$c \leftarrow_{\text{u.a.r.}} \mathcal{A}(x, \alpha_0)$ (return `failure` if $\mathcal{A}(x, \alpha_0) = \emptyset$).
return PPSZ(F, π, $\alpha_0 \wedge (x = c)$)

Definition 2.3 (Ultimately Eligible Values). *Let π a permutation of the variables, α be a satisfying assignment, α_0 a partial assignment compatible with α, and x a variable in $\mathcal{U}_{\alpha_0}$. Let y be the first variable of $\mathcal{U}_{\alpha_0}$ according to π.*

- *If $y = x$ set $\mathcal{A}(x, \alpha_0, \alpha, \pi) := \mathcal{A}(x, \alpha_0)$.*
- *Otherwise, set $\mathcal{A}(x, \alpha_0, \alpha, \pi) := \mathcal{A}(x, \alpha_0 \wedge (y = \alpha(y)), \alpha, \pi)$.*

This definition allows us to write down an explicit formula for the success probability of PPSZ. We write

$$p(\alpha_0, \alpha) := \Pr_{\pi}[\text{PPSZ}(F, \pi, \alpha_0) \text{ returns } \alpha] \ .$$

This is the probability that PPSZ returns one particular assignment α. Observe that PPSZ returns α if and only if it always picks the "correct" value for every $x \in \mathcal{U}_{\alpha_0}$. For a fixed permutation π this happens with probability $\frac{1}{|\mathcal{A}(x, \alpha_0, \alpha, \pi)|}$. Therefore we obtain

$$p(\alpha_0, \alpha) = \mathop{\mathbf{E}}_{\pi}\left[\prod_{x \in \mathcal{U}_{\alpha_0}} \frac{1}{|\mathcal{A}(x, \alpha_0, \alpha, \pi)|}\right] \ . \tag{1}$$

$$\geq d^{-\sum_{x \in \mathcal{U}(\alpha_0)} \mathbf{E}_{\pi}[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|]} \ . \qquad \text{(by Jensen's Inequality)}$$

A large part of this paper will be devoted to estimating $\mathbf{E}_{\pi}[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|]$. Note that in general there is no non-trivial upper bound: If $F = 1$, the constant 1 formula over n variables, then $\mathcal{A}(x, \alpha_0, \alpha, \pi) = d$ for all x and π and $p(\alpha_0, \alpha) = d^{-|\mathcal{U}_{\alpha_0}|}$. In particular this is d^{-n} if we start with the empty assignment $\alpha_0 = \emptyset$. In the other extreme, if there is only one possible value of x, we can actually give a non-trivial upper bound.

Definition 2.4 (Frozen Variables). *Let α_0 a partial assignment. A variable $x \in \mathcal{U}(\alpha_0)$ is* frozen *(in F with respect to α_0) if there is a value $c \in [d]$ such that $F \wedge \alpha_0 \models (x = c)$.*

Here we are talking about "full implication" $\models$, not D-implication $\models_D$.

Lemma 2.5. *Let F be a (d, k)-ClSP formula, α a satisfying assignment, α_0 a partial assignment compatible with α, and x a variable in $\mathcal{U}_{\alpha_0}$. If x is frozen in F with respect to α_0 then*

$$\mathop{\mathbf{E}}_{\pi}[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|] \leq S_{d,k} + \epsilon_D \ ,$$

where ϵ_D is an error parameter that goes to 0 as D goes to infinity.

This lemma immediately implies Theorem 1.1.

Proof (of Theorem 1.1). Suppose F has exactly one satisfying assignment α. Let α_0 be the empty assignment. Note that every x is frozen in F with respect to α_0.

$$\begin{aligned} p(\alpha_0, \alpha) &\geq d^{-\sum_{x \in \mathcal{U}(\alpha_0)} \mathbf{E}_\pi[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|]} \\ &\geq d^{-n(S_{d,k} + \epsilon_D)} \ . \end{aligned}$$

By making D a slowly growing function in n, we can make sure that PPSZ runs in subexponential time and has success probability $O^*\left(d^{-S_{d,k}n}\right)$. Repeating this procedure $O^*\left(d^{S_{d,k}n}\right)$ times guarantees a constant success probability.

3 Understanding $|\mathcal{A}(x, \alpha_0, \alpha, \pi)|$: Proof of Lemma 2.5

During this whole section we fix a partial assignment α_0, a satisfying assignment α of F that is compatible with α_0, and a variable x. Without loss of generality we let $\alpha = (d, \ldots, d)$. Since x is frozen we have $F \wedge \alpha_0 \models (x = d)$. Similar to [10] we construct *critical clause trees*.

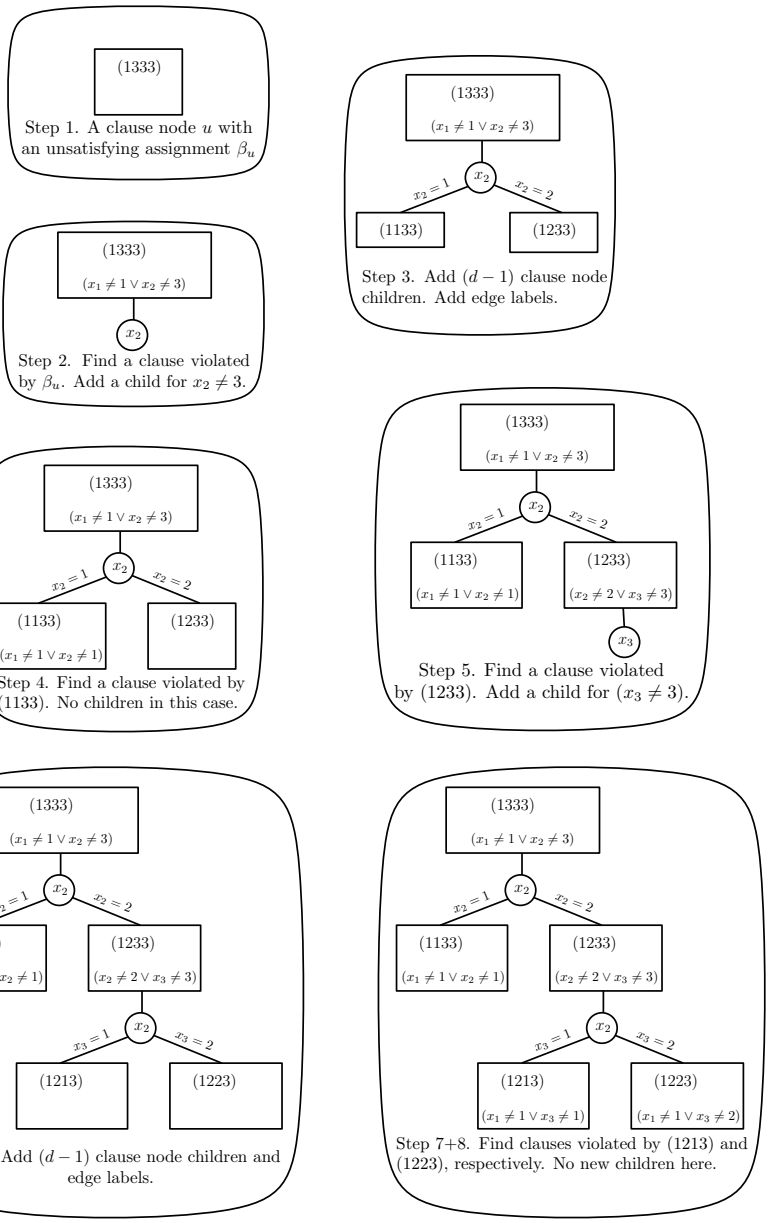
3.1 Construction of Critical Clause Trees

Consider a color $c \in \{1, \ldots, d-1\}$. The *critical clause tree* T_c has two types of nodes: *clause nodes* on even levels (this includes the root, which is on level 0) and *variable nodes* on odd levels. A clause node has a clause label clause-label$(c) \in F$ and an assignment label β_u; it will always hold that β_u is compatible with α_0 and violates clause-label(u); a clause node has at most $k-1$ children. A variable node v has a variable label var-label$(v) \in \mathcal{U}_{\alpha_0}$ and exactly $d-1$ children. Furthermore, each edge $e = (v, w)$ from a variable node v to a clause node w has an edge color edge-color$(e) \in [d-1]$. Here is how we construct T_c:

Create a a root vertex and set $\beta_{\text{root}} := \alpha[x = c]$.
`while` there is a leaf u without a clause label:
- Choose a clause C unsatisfied by β_u.
- Set clause-label$(u) := C$.
- `for all` literals $(y \neq d) \in C$:
  - Create a new child v of u. Set var-label$(v) = y$.
  - `for all` $i \in [d-1]$: Create a new child of w of v and set $\beta_w := \beta_u[y = i]$, edge-color$(v, w) = i$.

Proposition 3.1. *(1) The construction of T_c terminates. (2) Suppose u is a clause node in T_c, $C =$ clause-label(u) and $(y \neq i)$ is a literal in C. If $i = d$ then u has a child v with* var-label$(v) = y$. *If $i < d$ then u has an ancestor v with* var-label$(v) = y$. *(3) If* var-label$(v) =$ var-label(v') *then v is not an ancestor of v'. In other words, the set of variable nodes v with* var-label$(v) = y$ *is an anti-chain in T_c.*

Due to space constraints we will defer the proof of this proposition and of several lemmas below to the appendix.



Construction of a critical clause tree for $d = 3$, $V = \{x_1, x_2, x_3, x_4\}$, $\alpha = (3, 3, 3, 3)$, $\alpha_0 = \emptyset$, variable x_1, color $c = 1$ and formula $F = (x_1 \neq 1 \vee x_2 \neq 3) \wedge (x_1 \neq 1 \vee x_2 \neq 1) \wedge (x_2 \neq 2 \vee x_3 \neq 3) \wedge (x_1 \neq 1 \vee x_3 \neq 1) \wedge (x_1 \neq 1 \vee x_3 \neq 2)$.

Definition 3.2. *Let T_c be a critical clause tree and π be a permutation. A variable node v is* dead *if its variable label comes before x in π. It is* alive *if it is not dead. All clause nodes are alive, too. A node u is* reachable *if there is a path of alive nodes from the root to u.* Reachable(T_c, π) *is the set of all reachable vertices. Let $G(T_c, \pi)$ be the set of clause labels of the nodes in* Reachable(T_c, π).

Lemma 3.3 (Critical clause trees model local reasoning). *Let π be a permutation of the variables and $c \in [d-1]$ a color. Let β be the restriction of α to the variables coming before x in π. Then $G(T_c, \pi) \wedge \alpha_0 \wedge \beta \vDash (x \neq c)$.*

We encourage the reader to verify the lemma for the critical clause tree in the figure above, for example for $\pi = (x_2, x_1, x_3, x_4)$ or $\pi = (x_1, x_2, x_3, x_4)$.

Corollary 3.4. *If* $|\text{Reachable}(T_c, \pi)| \leq D$ *then $c \notin \mathcal{A}(x, \alpha_0, \alpha, \pi)$. In other words, PPSZ can eliminate color c for x by local reasoning.*

Proof (Proof of Lemma 3.3). We show the following equivalent statement: Let γ be a total assignment that is compatible with $\alpha_0 \wedge \beta$ and $\gamma(x) = c \neq d$. Then γ does not satisfy $G(T_c, \pi)$. We will prove this statement constructively by finding a clause that is violated by γ.

Set u to be the root of T_c.
`do` as long as possible:
- Let $C :=$ clause-label(u).
- `if` there is some $(y \neq d) \in C$ with $\gamma(y) = i \neq d$:
  - Let v be the child of u with var-label$(v) = y$.
  - Let w be the child of v such that edge-color$(v, w) = i$.
  - Set $u = w$ and continue.
- `else: return` u.

Let u be the vertex returned by this procedure. Consider any variable node v on the path from the root to u and let $y :=$ var-label(v). By construction $\beta_u(y) = \gamma(y) \neq d$. This means that y comes after x in π: Otherwise, $\gamma(y) = \alpha(y) = d$ by assumption on γ. So y comes after x, every ancestor v of u is alive, and u is reachable. Therefore $C :=$ clause-label$(u) \in G(T_c, \pi)$.

We claim that γ violates C: First consider a literal $(y \neq d) \in C$. If $\gamma(y) \neq d$, the above procedure would have continued, and not returned u. So $\gamma(y) = d$, and γ does not satisfy $(y \neq d)$. Second consider a literal $(z \neq i) \in C$ for some $i \neq d$. By Proposition 3.1 z appears as a variable label above u, and therefore $\gamma(z) = \beta_u(z)$. Since β_u violates C, it violates the literal $(z \neq i)$, thus γ violates it, too. We conclude that γ violates C. □

For $1 \leq c \leq d-1$ we define the indicator variable R_c which is 1 if $|\text{Reachable}(T_c, \pi)| > D|$ and 0 otherwise. By the above corollary we know that $R_c = 0$ implies $c \notin \mathcal{A}(x, \alpha_0, \alpha, \pi)$. Since $d \in \mathcal{A}(x, \alpha_0, \alpha, \pi)$ for all π we get $|\mathcal{A}(x, \alpha_0, \alpha, \pi)| \leq 1 + \sum_{c=1}^{d-1} R_c$. We now have to show that $\mathbf{E}\left[\log_d\left(1 + \sum_{c=1}^{d-1} R_c\right)\right] \leq S_{d,k} + \epsilon_D$.

Note that R_c depends on the number of reachable nodes. It is difficult to understand the worst-case behavior of the random variable $\sum R_c$. Let us therefore define a new ensemble of random variables:

$$P_c^h = \begin{cases} 1 & \text{if there exists a reachable vertex at depth } h \text{ in } T_c \ , \\ 0 & \text{else.} \end{cases}$$

Note that if $m := |\text{Reachable}|$ is very large, then there exist a reachable vertex at depth at least h, where h is logarithmic in m. The precise connection is: Let h be the largest even integer with $2(h/2)^{(k-1)(d-1)} \leq D$. Then $R_c \leq P_c^h$. So it suffices to bound $\mathbf{E}\left[\log_d\left(1+\sum_{c=1}^{d-1} P_c^h\right)\right]$ from above. Note that the behavior of $\sum P_c^h$ depends on (i) the shape of the critical clause trees; (ii) the concrete arrangement of variable labels in all $d-1$ trees. All can be pretty complex. Luckily, we can prove that in the worst-case, everything looks quite nice.

Lemma 3.5 (Independence Between Trees, Informal). *In the worst case, the trees $T_1, \ldots, T_{d-1}$ do not share any variable labels.*

This follows from a certain monotonicity argument and the concavity of $\log_d$.

Lemma 3.6 (Independence Within a Tree, Informal). *In the worst case, no variable label appears twice within a tree.*

A version of this lemma also appears in [10]. It follows from the FKG inequality [6] and the fact that P_c^h is monotone in each of the events "y comes after x in π". At this point we can forget all about variable and clause labels. Instead of thinking of π as a permutation on $\mathcal{U}_{\alpha_0}$, we think of it as assigning each variable x a random value $\pi(x) \in [0,1]$. With probability 1 this defines a permutation. Thus the ensemble $(P_1^h, \ldots, P_{d-1}^h)$ can be produced by the following random experiment: Select $p \in [0,1]$ uniformly at random (this corresponds to choosing $\pi(x)$). Then delete each odd-level node with probability p, independently (if $\pi(v) < \pi(x)$ then the node labeled v is dead). Now $P_c^h = 1$ if and only if after deletion, T_c contains a path of length h starting at its root.

Observation 3.7 (Deletion in Infinite Trees, Informal) *In the worst case, all T_c are infinite trees in which an even-level node has exactly $k-1$ children and an odd-level node exactly $d-1$.*

This "worst case" of infinite trees can of course not happen for an actual ClSP instance F. However, it is useful to imagine infinite trees in the analysis. Let us assume the trees $T_1, \ldots, T_{d-1}$ look as in the worst case outlined above, and write $Y^h := \sum_{c=1}^{d-1} P_c^h$. The distribution of Y^h does not depend on F, only on h, d, and k. We define P_c to 1 if T_c has an *infinite* path of alive vertices and set $Y := \sum_{c=1}^{d-1} Y_c$.

Lemma 3.8. $\mathbf{E}[\log_d(1+Y^h)]$ *converges to* $\mathbf{E}[\log_d(1+Y)] = S_{d,k}$ *as* $h \to \infty$.

Equivalently, $\mathbf{E}[\log_d(1+Y^h)] = S_{d,k} + \epsilon_D$ for some ϵ_D that converges to 0 as D grows. To sum up,

$$\mathop{\mathbf{E}}_{\pi}[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|] \leq \mathbf{E}\left[\log_d\left(1+Y^h\right)\right] = S_{d,k} + \epsilon_D \ .$$

4 General (d, k)-ClSP

The intuition behind the analysis of the general case is: Our partial assignment α_0 represents the current state of PPSZ (i.e. the variable assignments it has already made). If a variable x is frozen at this point in time (cf. Definition 2.4), then Lemma 2.5 gives us an upper bound on $\mathbf{E}[|\mathcal{A}(x, \alpha_0, \alpha, \pi)|]$. Otherwise, if x is not frozen, we have at least a $2/d$ chance of guessing a value for x that keeps F satisfiable.

Below we will carefully track how $\mathbf{E}[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|]$ changes over time after x becomes frozen. Surprisingly we only use one property of our D-implication mechanism: adding more information to α_0 can only decrease the number of eligible colors:

> Let $y \neq x$ and $c := \alpha(y)$. Then $\mathcal{A}(x, \alpha_0 \wedge y = c) \subseteq \mathcal{A}(x, \alpha_0)$.

4.1 Definitions and Notation

Through most of the analysis, we consider a certain "snapshot" of PPSZ. At this point in time it has already assigned some variables, and we represent this by the partial assignment α_0.

Definition 4.1. *Let F be a (d, k)-ClSP formula and α_0 a partial assignment. Let $x \in \mathcal{U}(\alpha_0)$.*

- *$\mathcal{A}(x, \alpha_0)$ is the set of eligible values as in Definition 2.2.*
- *$\mathcal{S}_{\alpha_0}(x)$ is the set of values $c \in [d]$ such that $F \wedge \alpha_0 \wedge (x = c)$ is satisfiable.*
- *$\mathcal{S}_{\alpha_0} := \{(x, c) \in \mathcal{U}(\alpha_0) \times [d] \big| c \in \mathcal{S}_{\alpha_0}(x)\}$.*

Note that a variable x is frozen if and only if $|\mathcal{S}_{\alpha_0}(x)| = 1$. Also, $\mathcal{S}_{\alpha_0}(x) \subseteq \mathcal{A}(\alpha_0, x)$. We partition the set $\mathcal{U}(\alpha_0)$ of yet unassigned variables into the parts: $\mathcal{U}(\alpha_0) = \mathrm{V}_{\mathrm{fo}}(\alpha_0) \dot\cup \mathrm{V}_{\mathrm{fr}}(\alpha_0) \dot\cup \mathrm{V}_{\mathrm{nf}}(\alpha_0)$ where

- $\mathrm{V}_{\mathrm{nf}}(\alpha_0) := \{x \in \mathcal{U}(\alpha_0) \mid |\mathcal{S}_{\alpha_0}(x)| \geq 2\}$, i.e., the set of non-frozen variables.
- $\mathrm{V}_{\mathrm{fo}}(\alpha_0) := \{x \in \mathcal{U}(\alpha_0) \mid |\mathcal{A}(\alpha_0, x)| = 1\}$, i.e., those variables for which the D-implication mechanism of PPSZ can rule out all but one value. Clearly, such a variable is also frozen. We call such a variable *forced*.
- $\mathrm{V}_{\mathrm{fr}}(\alpha_0) :=$ the set of frozen variables not in $\mathrm{V}_{\mathrm{fo}}(\alpha_0)$.

Lemma 2.5 guarantees that $\mathbf{E}_\pi[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|] \leq S_{d,k} + \epsilon_D$ whenever x is frozen. We write $S := S_{d,k} + \epsilon_D$ and $G := \max\{S, 1 - \frac{\log_d e}{2}\}$. As in [7] we define a *cost function*:

Definition 4.2. *Let α_0 be a partial and α a total assignment and x a variable. We define $\mathrm{cost}(\alpha_0, \alpha, x)$ as follows:*

- *If $x \notin \mathcal{U}(\alpha_0)$ or α_0, α are incompatible or α violates F, or x is forced with respect to α_0 then $\mathrm{cost}(\alpha_0, \alpha, x) = 0$;*
- *else if $x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$ then $\mathrm{cost}(\alpha_0, \alpha, x) = G$;*
- *else (if $x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$) then $\mathrm{cost}(\alpha_0, \alpha, x) = \mathbf{E}_\pi[\log_d(|\mathcal{A}(x, \alpha_0, \alpha, \pi)|)]$.*

We define $\mathrm{cost}(\alpha_0, \alpha) = \sum_{x \in \mathcal{U}(\alpha_0)} \mathrm{cost}(\alpha_0, \alpha, x)$.

Note that $\mathrm{cost}(\alpha_0, \alpha) \leq G \cdot n(\alpha_0)$ by Lemma 2.5.

4.2 A distribution over satisfying assignments

Let α_0 be a partial assignment such that $F \wedge \alpha_0$ is satisfiable. We define a (not computationally efficient) process that samples a random satisfying assignment:

> `while` $\mathcal{U}(\alpha_0) \neq \emptyset$:
> – Pick $(x, c) \in \mathcal{S}_{\alpha_0}$.
> – Add $(x = c)$ to α_0.
> `return` α_0.

Note that this process always outputs a total satisfiable assignment compatible with (the original) α_0. Let $\mathrm{Q}(\alpha_0, \alpha)$ be the probability that this process, started with α_0, outputs α. This defines a probability distribution over the set of satisfying assignments of F. Let $p(\alpha_0, \alpha)$ denote the probability that PPSZ(F, α_0) returns α .

Lemma 4.3. *Let α be a satisfying assignment, α_0 be a partial assignment compatible with α. Then $p(\alpha_0, \alpha) \geq \mathrm{Q}(\alpha_0, \alpha) \cdot d^{-\operatorname{cost}(\alpha_0, \alpha)}$.*

Proof (of Theorem 1.2). Let α_0 be the empty assignment. Then

$$\Pr[\mathrm{PPSZ}(F, \alpha_0) \text{ succeeds}] = \sum_{\alpha \in \mathrm{sat}(F)} p(\alpha_0, \alpha)$$

$$\geq \sum_{\alpha \in \mathrm{sat}_V(F)} \mathrm{Q}(\alpha_0, \alpha) \cdot d^{-\operatorname{cost}(\alpha_0, \alpha)} \geq \sum_{\alpha \in \mathrm{sat}_V(F)} \mathrm{Q}(\alpha_0, \alpha) \cdot d^{-Gn} = d^{-Gn} \ .$$

□

The rest of this section is devoted to proving Lemma 4.3. We prove $p(\alpha_0, \alpha) \geq \mathrm{Q}(\alpha_0, \alpha) \cdot d^{-\operatorname{cost}(\alpha_0, \alpha)}$ by induction over $|\mathcal{U}(\alpha_0)|$, the number of variables unassigned in α_0. If α_0 is total the statement holds trivially.

For the induction step suppose α_0 is not total. PPSZ randomly picks $x \in \mathcal{U}(\alpha_0)$ and $c \in |\mathcal{A}(x, \alpha_0)|$, adds $(x = c)$ to α_0 and continues. For the rest of this inductive proof, the meaning of α and α_0 will not change. We thus drop the α_0 from $\mathcal{S}_{\alpha_0}, \mathcal{S}_{\alpha_0}(x), \mathcal{A}(x, \alpha_0), \mathcal{U}_{\alpha_0}, \mathrm{V}_{\mathrm{nf}}(\alpha_0), \ldots$. We also write $\mathcal{S}, \mathcal{S}(x), \mathcal{A}(x)$ and write $s := |\mathcal{S}|, s(x) := |\mathcal{S}(x)|, a(x) := |\mathcal{A}(x)|$. Finally, since PPSZ adds $(x = c)$ to α_0, we have to look at partial assignments that extend α_0 by one variable. For this we write $\alpha_0^{x=c} := \alpha_0 \wedge (x = c)$. Most of the time we consider partial assignments that fix one additional variable x to $\alpha(x)$. We denote this by $\alpha_0^x := \alpha_0^{x=\alpha(x)}$.

Given the current partial assignment α_0, PPSZ randomly picks some $x \in \mathcal{U}$ and $c \in \mathcal{A}(x)$ and continues with $\alpha_0^{x=c}$. Thus

$$p(\alpha_0, \alpha) = \mathop{\mathbf{E}}_{x \in \mathcal{U}} \left[\mathop{\mathbf{E}}_{c \in \mathcal{A}(x)} [p(\alpha_0^{x=c}, \alpha)] \right] = \mathop{\mathbf{E}}_{x \in \mathcal{U}} \left[\frac{1}{a(x)} \cdot p(\alpha_0^{x=\alpha(x)}, \alpha) \right] \ ,$$

The second equality holds since $p(\alpha_0^{x=c}, \alpha) = 0$ for $c \neq \alpha(x)$. Applying the induction hypothesis to $\alpha_0^{x=\alpha(x)}$ (or a_0^x, for short):

$$p(\alpha_0, \alpha) \geq \mathop{\mathbf{E}}_{x \in \mathcal{U}} \left[\frac{\mathrm{Q}(\alpha_0^x, \alpha) d^{-\operatorname{cost}(\alpha_0^x, \alpha)}}{a(x)} \right]$$

We could apply Jensen's inequality to the above expectation. However, the argument of $\mathbf{E}$ above includes Q, which is not necessarily very concentrated around its expectation, and Jensen's inequality not seem to yield any usable bound. To circumvent this problem, we introduce a new probability distribution $\xi(x)$ over $\mathcal{U}(\alpha_0)$ that is proportional to $\mathrm{Q}(\alpha_0^x, \alpha)$. Note that

$$\mathrm{Q}(\alpha_0, \alpha) = \mathop{\mathbf{E}}_{(x,c) \in \mathcal{S}} \mathrm{Q}(\alpha_0^{x=c}, \alpha) = \frac{1}{s} \sum_{x \in \mathcal{U}} \sum_{c \in \mathcal{S}(x)} \mathrm{Q}(\alpha_0^{x=c}, \alpha) = \frac{1}{s} \sum_{x \in \mathcal{U}} \mathrm{Q}(\alpha_0^x, \alpha) \ ,$$

and therefore $\xi(x) := \frac{\mathrm{Q}(\alpha_0^x, \alpha)}{s \cdot \mathrm{Q}(\alpha_0, \alpha)}$ is a probability distribution over $\mathcal{U}$. Thus,

$$\begin{aligned} p(\alpha_0, \alpha) &\geq \mathop{\mathbf{E}}_{x \in \mathcal{U}} \left[\frac{1}{a(x)} \cdot \mathrm{Q}(\alpha_0^x, \alpha) \cdot 2^{-\operatorname{cost}(\alpha_0^x, \alpha)} \right] \\ &= \frac{s \cdot \mathrm{Q}(\alpha_0, \alpha)}{|\mathcal{U}|} \mathop{\mathbf{E}}_{x \sim \xi} \left[\frac{2^{-\operatorname{cost}(\alpha_0^x, \alpha)}}{a(x)} \right] \\ &\geq \frac{s \cdot \mathrm{Q}(\alpha_0, \alpha)}{|\mathcal{U}|} 2^{\mathbf{E}_{x \sim \xi}[-\operatorname{cost}(\alpha_0^x, \alpha) - \log_d a(x)]} \ . \qquad \text{(by Jensen's)} \end{aligned}$$

In order for our inductive prove to go through, the last expression should be at least $\mathrm{Q}(\alpha_0, \alpha) \cdot 2^{-\operatorname{cost}(\alpha_0, \alpha)}$. This happens if and only if

$$\begin{aligned} \frac{s}{|\mathcal{U}|} \cdot 2^{\mathbf{E}_{x \sim \xi}[-\operatorname{cost}(\alpha_0^x, \alpha) - \log_d a(x)]} \geq 2^{-\operatorname{cost}(\alpha_0, \alpha)} \iff \\ \log_d \frac{s}{|\mathcal{U}|} - \mathop{\mathbf{E}}_{x \sim \xi}[\operatorname{cost}(\alpha_0^x, \alpha) + \log_d a(x)] \geq -\operatorname{cost}(\alpha_0, \alpha) \iff \\ \mathop{\mathbf{E}}_{x \sim \xi}[\operatorname{cost}(\alpha_0, \alpha) - \operatorname{cost}(\alpha_0^x, \alpha)] - \mathop{\mathbf{E}}_{x \sim x}[\log_d a(x)] + \log_d \frac{s}{|\mathcal{U}|} \geq 0 \ . \end{aligned} \tag{2}$$

We defer the (quite demanding) proofs of the next three lemmas to the appendix.

Lemma 4.4. $\mathbf{E}_{x \sim \xi}[\operatorname{cost}(\alpha_0, \alpha) - \operatorname{cost}(\alpha_0^x, \alpha)] \geq \frac{1}{s} \left(G \sum_{y \in \mathrm{V_{nf}}} s(y) + \sum_{y \in \mathrm{V_{fr}}} \log_d a(y) \right)$.

Lemma 4.5. $\mathbf{E}_{x \sim \xi}[\log_d a(x)] \leq \frac{\sum_{x \in \mathcal{U}} \log_d a(x)}{s} + \frac{\sum_{x \in \mathrm{V_{nf}}} (s(x) - 1)}{s}$.

Lemma 4.6. $\log_d \frac{s}{|\mathcal{U}|} \geq \log_d(e) \frac{\sum_{y \in \mathrm{V_{nf}}} (s(y) - 1)}{s}$.

Let $(*)$ denote the left-hand side of inequality (2).

$$\begin{aligned} s \cdot (*) &\geq G \sum_{y \in \mathrm{V_{nf}}} s(y) + \sum_{y \in \mathrm{V_{fr}}} \log_d a(y) - \sum_{x \in \mathcal{U}} \log_d a(x) - \sum_{x \in \mathrm{V_{nf}}} (s(x) - 1) + \log_d(e) \sum_{y \in \mathrm{V_{nf}}} (s(y) - 1) \\ &= \sum_{y \in \mathrm{V_{nf}}} (G s(y) - s(y) + 1 + \log_d(e)(s(y) - 1)) - \sum_{y \in \mathcal{U}} \log_d a(y)(1 - \mathbb{I}_{y \in \mathrm{V_{fr}}}) \ . \end{aligned}$$

Note that $\log_d a(y)(1 - \mathbb{I}_{y \in \mathrm{V_{fr}}})$ is equal to 0 if $y \in \mathrm{V_{fr}} \cup \mathrm{V_{fo}}$ and at most 1 if $y \in \mathrm{V_{nf}}$. Thus it suffices to show that $\sum_{y \in \mathrm{V_{nf}}} (Gs(y) - s(y) + \log_d(e)(s(y) - 1)) \geq 0$. We will show that every summand is non-negative for each $y \in \mathrm{V_{nf}}$:

$$Gs(y)s(y) + \log_d(e)(s(y) - 1 \geq 0 \quad \Leftrightarrow G \geq 1 - \log_d(e) \cdot \frac{s(y) - 1}{s(y)} \ .$$

The last inequality holds because $\frac{s(y)-1}{s(y)} \geq 1/2$ for $y \in \mathrm{V_{nf}}$ and $G \geq 1 - \frac{\log_d(e)}{2}$ by definition. This finishes the proof.

5 Conclusion and Open Problems

We have shown how to apply the PPSZ algorithm to (d, k)-ClSPs. In the unique case we established correlation inequalities showing that PPSZ behaves as expected. This improves the fastest known running time for Unique (d, k)-ClSP algorithm for almost all values (d, k). These results transfer to the general case for $k \geq 4$.

In our analysis of the general case we only distinguish frozen and non-frozen variables. That is, for non-frozen variables we make no difference between variables with two, three, or even d viable values. A more fine-grained analysis could give an improved result for the general case. However we do not know how to analyze the transition between different types of "non-frozen-ness".

We conjecture that the running time in the general case is no worse than in the unique case and that the current discrepancy for $k = 2, 3$ is a shortcoming of our analysis, not the algorithm.

References

1. E. Allender, S. Chen, T. Lou, P. A. Papakonstantinou, and B. Tang. Width parameterized SAT: time-space tradeoffs. *Theory of Computing*, (to appear).
2. R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. *J. Algorithms*, 54(2):168–204, 2005.
3. A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.
4. A. Björklund and T. Husfeldt. Inclusion–exclusion algorithms for counting set partitions. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 575–582. IEEE Computer Society, 2006.
5. T. Feder and R. Motwani. Worst-case time bounds for coloring and satisfiability problems. *J. Algorithms*, 45(2):192–201, 2002.
6. C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Comm. Math. Phys.*, 22:89–103, 1971.
7. T. Hertli. 3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General. *SIAM J. Comput.*, 43(2):718–729, 2014.

8. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity. *J. Comput. System Sci.*, 63(4):512–530, 2001. Special issue on FOCS 98 (Palo Alto, CA).
9. L. Li, X. Li, T. Liu, and K. Xu. From k-sat to k-csp: Two generalized algorithms. *CoRR*, abs/0801.3147, 2008.
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. Scheder. Ppz for more than two truth values - an algorithm for constraint satisfaction problems. *CoRR*, abs/1010.5717, 2010.
13. U. Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 410–414. IEEE Computer Society, 1999.
14. S. Szeider. On fixed-parameter tractable parameterizations of SAT. 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 188–202. Springer, 2003.
15. P. Traxler. The time complexity of constraint satisfaction. In M. Grohe and R. Niedermeier, editors, *Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings*, volume 5018 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 2008.
16. E. Welzl. Boolean Satisfiability – Combinatorics and Algorithms (Lecture Notes), 2005. `www.inf.ethz.ch/~emo/SmallPieces/SAT.ps`.

A Analysis of Critical Clause Trees

Let us briefly review the setup of Section 3. Our goal is to prove Lemma 2.5, stating that $\mathbf{E}[\log_d(|\mathcal{A}(x, \alpha_0, \alpha, \pi)|)] \leq S_{d,k} + \epsilon_D$ for some ϵ_D that converges to 0 as D grows. Towards a proof we defined a collection of *critical clause trees* $T_1, \ldots, T_{d-1}$. Such a tree has *clause nodes* on even levels (which includes the root) and *variable nodes* on odd levels. Every variable node v has a variable label var-label(v). We first prove some facts about critical clause trees:

Proposition 3.1, restated. *(1) The construction of T_c terminates. (2) Suppose u is a clause node in T_c, $C =$ clause-label(u) and $(y \neq i)$ is a literal in C. If $i = d$ then u has a child v with* var-label(v) $= y$. *If $i < d$ then u has an ancestor v with* var-label(v) $= y$. *(3) If* var-label(v) $=$ var-label(v') *then v is not an ancestor of v'. In other words, the set of variable nodes v with* var-label(v) $= y$ *is an anti-chain in T_c.*

Proof. We start by showing that the process is well-defined. Every assignment label β_u occurring in the tree has $\beta_u(x) \neq d$—indeed, we never modify β by changing a value back to d. Thus, β_u violates F and we can always find a clause label for a clause node. Next let us show (2). Suppose u is a clause node. If $(z \neq d)$ is a literal in $C :=$ clause-label(u) then by construction, u will get a child v with var-label(v) $= z$. Now suppose $y \neq i$ is a literal in C. Since β violates C it follows that $\beta_u(y) = i$. It is clear from the construction that β_u is changed bit by bit when adding a new clause node and that there is some variable node v with var-label(v) $= y$ that is an ancestor of u. This shows (2). Furthermore, once $\beta_u(y) = i < d$, this also implies $\beta_w(y) = i$ for all clause nodes w that are descendants of u (since the process never changes a value back to d). Thus we see that clause-label(w) does not contain the literal $(y \neq d)$. This in turn means no variable node v in the subtree of u has var-label(v) $= y$. We therefore see that the variable nodes along a descending path in T_c have distinct variable labels. Thus it contains at most $|V|$ variable nodes, which shows (1), that the process terminates. Also it means that the variable nodes v in T_c with var-label(v) $= y$ form an antichain, which shows (3). □

A.1 Analyzing Critical Clause Trees

For an integer h we define the following random variables $R_1^h, \ldots, R_{d-1}^h$: For a (uniformly random) permutation π of the variables, call a variable node v *dead* if var-label(v) comes before x in π. Otherwise, call it *alive*. All clause nodes are alive, too. Let R_c^h be 1 if T_c contains a path of length h which starts at the root and contains only alive vertices. For h being the largest even integer with $2(h/2)^{(k-1)(d-1)} \leq D$ we have shown in Section 3 that $|\mathcal{A}(x, \alpha_0, \alpha, \pi)| \leq 1 + \sum_{c=1}^{d-1} P_c^h$ and to proof of Lemma 2.5 we have to prove an upper bound on $\mathbf{E}[\log_d(1 + \sum_{c=1}^{d-1} P_c^h)]$. In this section we will do this formally.

A.2 An Alternative View of Permutations

Instead of viewing π as a uniformly random permutation of the variables V of F, we think of it as a uniformly random function $V \to [0,1]$. With probability one this defines a permutation on V. It has a clear advantage: when we condition on $[\pi(x) = p]$ the events $[\pi(u) < \pi(x)]$ for $u \in V$ become independent.

Lemma 3.5 [Independence Between Trees], restated.
Informal Statement: *In the worst case, the trees $T_1, \ldots, T_{d-1}$ do not share any variable labels.*

Formal Statement: *Define $\tilde{P}_1^h, \ldots, \tilde{P}_{d-1}^h$ as follows: Sample $p \in [0,1]$ randomly, and then set each $\tilde{P}_c^h$ independently to 1 with probability $\Pr[P_c^h = 1 \mid \pi(x) = p]$. Then*

$$\mathbf{E}\left[\log_d\left(1 + \sum_{c=1}^{d-1} P_c^h\right)\right] \leq \mathbf{E}\left[\log_d\left(1 + \sum_{c=1}^{d-1} \tilde{P}_c^h\right)\right] .$$

Proof. We prove that the lemma still holds when we condition on $\pi(x) = p$ for every $p \in [0,1]$. Let $L \subseteq V$ be the set of variable labels occurring in the trees $T_1, \ldots, T_{(d-1)}$. The expressions $[\pi(y) \geq \pi(x)]$ for $y \in L$ are now independent binary random variables with expectation $1 - p$ each. Recall that P_c^h is the indicator variable that there is a path of length h that starts at the root at contains only alive nodes. This is a monotone boolean function in the boolean variables $[\pi(y) \geq p]$. The lemma will follow from the following more general lemma.

Lemma A.1 (Concave Correlation Lemma). *Let $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}_0^+$ be monotone boolean functions. Let $X \sim \{0,1\}^n$ be sampled by setting each coordinate to 1 with some probability q. Let $X_1, \ldots, X_k$ be independent copies of X (that is, each X_i has the same distribution as X, but all X_i are independent). Then*

$$\mathbf{E}[g(f_1(X) + \cdots + f_k(X))] \leq \mathbf{E}[g(f_1(X_1) + \cdots + f_k(X_k))] ,$$

for every concave function $g : \mathbb{R}_0^+ \to \mathbb{R}$.

Proof. Write $X = YZ$ where $Y \in \{0,1\}^{n-1}$ and $Z = \{0,1\}$. We first show that we can "make the last bit" independent, i.e.,

$$\mathbf{E}[g(f_1(X) + \cdots + f_k(X))] \leq \mathbf{E}[g(f_1(YZ_1) + \cdots + f_k(YZ_k))] , \qquad (3)$$

where $Z_1, \ldots, Z_k$ are independent binary random variables with expectation q each. The statement of the lemma will then follow by fixing a choice of each Z_i and then applying (3) to the second-last bit, and so on (note fixing the last bit to some value gives rise to certain functions $f_1', \ldots, f_k' : \{0,1\}^{n-1} \to \mathbb{R}_0^+$, which are again monotone).

So we only have to prove (3). Fix a choice for $Y \in \{0,1\}^{n-1}$. This makes $f_i(Y,Z)$ a monotone one-bit function, which is either 0, 1, or Z. To ease notation we assume that $f_1(Y,Z) = \cdots = f_\ell(Y,Z) = Z$, followed by a indices i for which $f_i(Y,Z) = 1$, followed by $k-\ell-a$ ones that are 0. Thus it suffices to show

$$\mathbf{E}[g(\ell Z + a)] \leq \mathbf{E}[g(Z_1 + \cdots + Z_\ell + a)] \tag{4}$$

Let $h(x) := g(x+a)$. This is concave, too. Note that Z and Z_1 have the same distribution. Therefore,

$$\begin{aligned}
\mathbf{E}[g(\ell Z + a)] &= \mathbf{E}[h(\ell Z)] = \mathbf{E}[h(\ell Z_1)] \\
&= \mathbf{E}\left[\mathbf{E}[h(\ell Z_1)|Z_1 + \cdots + Z_\ell]\right] \\
&\leq \mathbf{E}\left[h(\mathbf{E}[\ell Z_1|Z_1 + \cdots + Z_\ell])\right] && \text{(by Jensen's Inequality)} \\
&= \mathbf{E}\left[h\left(\ell \cdot \frac{Z_1 + \cdots + Z_\ell}{\ell}\right)\right] && \text{(by symmetry)} \\
&= \mathbf{E}[h(Z_1 + \cdots + Z_\ell)]\ .
\end{aligned}$$

This finishes the proof. □

We now apply the concave correlation lemma with $n = |L|$, $f_c = P_c^h$, and $g(t) = \log_d(1+t)$. This finishes the proof of Lemma 3.5. □

Next we have to prove that in the worst case, no label appears twice in a tree $T_x^{(c)}$. That is, $\mathbf{E}[R_c]$ is maximized if all variable labels of T_c are distinct. Paturi, Pudlák, Saks, and Zane [10] do this using the FKG inequality [6]. Sure enough, this works here, too, but notation becomes a mess. It is actually easier to do it "by hand".

Lemma 3.6 [Independence Within a Tree], restated.
Informal Statement: *In the worst case, no variable label appears twice within a tree.*

Formal Statement: *Let $\hat{T}$ be a tree that looks exactly like $T^{(c)}$ but in which all variable labels are distinct. Let $\hat{P}_c^h$ be the event that is defined analogously to P_c^h but referring to $\hat{T}_c$ instead of T_c. Then* $\Pr[P_c^h = 1] \leq \Pr[\hat{P}_c^h = 1]$.

Proof. We show that the statement is true conditioned on $\pi(x) = p$ for any $p \in [0,1]$. L be the set of variable labels in T_c. Note that x does not occur as a variable label in T_c. Our indicator variable P_c^h is in fact a monotone boolean function in the variables $[\pi(y) \geq p]$, $y \in L$. Each of those Boolean variables is 1 with probability $1-p$. To ease notation we will write P instead of P_c^h (within this proof, c and h are anyway fixed).

Let $y \in L$ be a label that occurs several times in T. Let v be some variable node of T with var-label$(v) = y$. We create a new variable $y' \notin L$ and construct a new tree T' that looks like T, but we label v with y' instead of y. Let P' denote the analog of P for this tree T'. We want to show that $\mathbf{E}[P] \leq \mathbf{E}[P']$. Once we

have shown this, we can iteratively replace non-unique labels by new labels, in the end arriving at some tree T'' in which every odd-level vertex has a unique label. Then we will be done.

We will in fact show that $\mathbf{E}[P] \leq \mathbf{E}[P']$ holds even conditioned on some choice for $\pi(z) \in [0,1]$ for every $z \in L \setminus \{y\}$. Under this restriction, P becomes a monotone function in $[\pi(y) \geq p]$, and P' becomes a monotone function in $[\pi(y) \geq p]$ and $[\pi(y') \geq p]$. There are several cases:

1. P' becomes constant (0 or 1). Then the restriction reduces P to the same constant, and we are done.
2. P' becomes $[\pi(y) \geq p]$ or becomes $[\pi(y') \geq p]$. Since y' labels a vertex labeled y in T, P reduces to $[\pi(y) \geq p]$ and we are done as well.
3. P' becomes $[\pi(y) \geq p] \vee [\pi(y') \geq p]$. Then P becomes $[\pi(y) \geq p]$ and $\mathbf{E}[P] = p \leq 2p - p^2 = \mathbf{E}[P']$.
4. P' becomes $[\pi(y) \geq p] \wedge [\pi(y') \geq p]$.

We claim that Case 4 is impossible: Indeed, fixing $\pi(z)$ for all $z \in L \setminus \{y\}$ decides which vertices of T' are dead or alive, except those labeled y or y'. However, since the set of vertices labeled y or y' in T' are those labeled y in T, they form an antichain in T (and thus in T', too). Therefore, the existence of an alive path of length h in T' is a *disjunction* of (zero, one, or two) the literals $[\pi(y) \geq p]$ and $[\pi(y') \geq p]$, never a conjunction. This finishes the proof. □

With Lemma 3.5 and Lemma 3.6 we can now assume that all variable labels in $T_1, \ldots, T_{d-1}$ are distinct. Recall that in T_c every clause-level vertex has at most $k-1$ children, and every odd-level vertex has at most $d-1$ children. Obviously, in the worst case T_c every even-level vertex has *exactly* $k-1$ children (up to level h, after which it does not matter). This motivates the following random experiment:

Let $T_1, \ldots, T_{d-1}$ be infinite trees in which every even-level vertex has degree $k-1$ and every odd-level vertex has degree $d-1$. Sample $p \in [0,1]$ randomly and delete every odd-level vertex with probability p, independently. Let T'_c be the component of containing the root after deletion. We define random variables Y_c^h as follows: If T'_c contains a vertex on level h (after deletion), set $Y_c^h = 1$, otherwise $Y_c^h = 0$. The last few paragraphs can now be summarized in the following lemma:

Lemma A.2 (Deletion in Infinite Trees).

$$\mathbf{E}\left[\log_d\left(1 + \sum_{c=1}^{d-1} R_c\right)\right] \leq \mathbf{E}\left[\log_d\left(1 + \sum_{c=1}^{d-1} Y_c^h\right)\right] . \tag{5}$$

We are now in a much more comfortable situation since the distribution of $(Y_1^h, \ldots, Y_{d-1}^h)$ is independent of the input formula F. Note that for each c it holds that $1 \geq Y_c^h \geq Y_c^{h+1} \geq 0$ and therefore $Y_c := \lim_{h\to\infty} Y_c^h$ exists. A moment

of thought shows that $Y_c = 1$ if and only if after deletion, T_c has arbitrarily long paths (equivalently, an infinite path) starting at the root. Let $\mathcal{E}^h$ be the event $(Y_1^h, \ldots, Y_{d-1}^h) \neq (Y_1, \ldots, Y_h)$. That is, after deletion some tree contains a path of length h starting at the root, but does not contain an infinite such path. By a union bound we have, for any $c \in [d-1]$:

$$\Pr[\mathcal{E}^h] \leq d \cdot \Pr[Y_c^h = 1 \wedge Y_c = 0] = d(\Pr[Y_c^h = 1] - \Pr[Y_c = 1] \ .$$

By the Monotone Convergence Theorem, $\lim_{h\to\infty} \Pr[Y_c^h = 1] = \Pr[Y_c = 1]$ and therefore $\Pr[\mathcal{E}^h] \to 0]$. To summarize:

$$\begin{aligned}
\mathbf{E}\left[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi)|\right] &\leq \mathbf{E}\left[\log_d \left(1 + \sum_{c=1}^{d-1} R_c\right)\right] \\
&\leq \mathbf{E}\left[\log_d \left(1 + \sum_{c=1}^{d-1} Y_c^h\right)\right] \\
&\leq \mathbf{E}\left[\log_d \left(1 + \sum_{c=1}^{d-1} Y_c\right)\right] + \Pr[\mathcal{E}^h] \\
&= S_{d,k} + \Pr[\mathcal{E}^h] \ .
\end{aligned}$$

The last inequality holds because even when $\mathcal{E}$ holds, $\log_d \left(1 + \sum_{c=1}^{d-1} Y_c^h\right)$ is at least 0 and $\log_d \left(1 + \sum_{c=1}^{d-1} Y_c\right)$ is at most 1. The last equality holds because the random variables $Y_1, \ldots, Y_{d-1}$ are exactly as in the definition of $S_{d,k}$. Since h increases as D does, we can set $\epsilon_D := \Pr[\mathcal{E}^h]$, which goes to 0 as D grows. This finishes the proof of Lemma 2.5.

B Computing $S_{d,k}$

We have proved Theorem 1.1, which states that PPSZ finds a unique satisfying assignment with probability $O^*\left(d^{-S_{d,k}\cdot n}\right)$. In this section we will show how to compute $S_{d,k}$. Let us start by repeating the definition of $S_{d,k}$.

Let $T_{d,k}$ be the infinite tree in which every even-level vertex has $k-1$ children and every odd-degree vertex has $d-1$ children. Sample $p \in [0,1]$ uniformly at random and delete every odd-level vertex with probability p. Let Y be the indicator variable for the "survival", i.e., $Y = 1$ iff the component containing the root is infinite. Now, for $p \in [0,1]$ uniformly at random, perform this random experiment $d-1$ times independently, and let $Y_1, \ldots, Y_{d-1}$ be the corresponding "survival indicator variables". Then

$$S_{d,k} := \mathbf{E}\left[\log_d\left(1 + Y_1 + \cdots + Y_{d-1}\right)\right] \ .$$

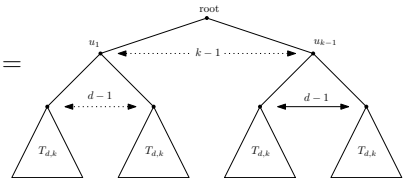
When we condition on some specific value for p, the Y_i become independent binary random variables with probability y_p each.

$$\begin{aligned}
S_{d,k} &= \int_0^1 \mathbf{E}\left[\log_d\left(1 + Y_1 + \cdots + Y_{d-1}\right) | \, p\right] dp \\
&= \int_0^1 \sum_{j=1}^{d-1} \log_d(1+j) \Pr[Y_1 + \cdots + Y_{d-1} = j \mid p] dp \\
&= \int_0^1 \sum_{j=1}^{d-1} \log_d(1+j) \binom{d-1}{j} y_p^j (1-y_p)^{d-1-j} dp \\
&= \sum_{j=1}^{d-1} \log_d(1+j) \binom{d-1}{j} \int_0^1 y_p^j (1-y_p)^{d-1-j} dp \qquad (6)
\end{aligned}$$

B.1 Computing y_p

It remains to compute y_p, the probability of the survival event. Most of this standard Galton-Watson branching process theory. In fact, it will be easier to compute $z_p := 1 - y_p$, the probability, conditioned on p, that the component of the root is finite after deletion. Note that $T_{d,k}$ has the following simple recursive structure:

$T_{d,k} :=$

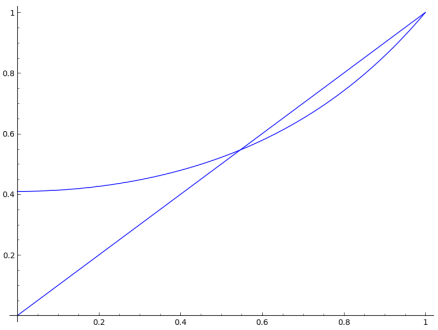


Let us say "extinction happens" if the component of the root is finite after deletion. For extinction to happen, the following must happen at every child v_i of the root: (1) the vertex v_i is deleted, or (2) extinction happens in each of the $d-1$ subtrees rooted at the children of v_i. Conditioned on p, the event (1) has probability p, and (2) has probability z_p^{d-1}. Thus, we get an equation for z_p:

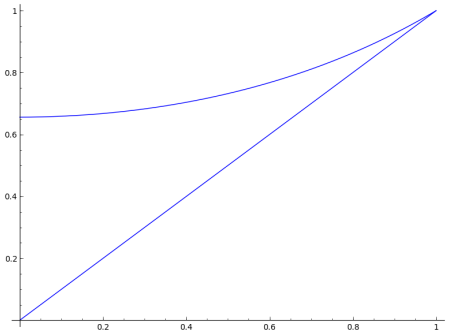
$$z_p = \left(p + (1-p)z_p^{d-1}\right)^{k-1} \quad . \tag{7}$$

Lemma B.1. *The extinction probability z_p is the smallest solution in $[0,1]$ of equation (7). Furthermore, $z_p = 1$ if and only if $p \geq 1 - \frac{1}{(d-1)(k-1)}$.*

Proof. Let $f(z) = \left(p + (1-p)z^{d-1}\right)^{k-1}$. Clearly, z_p is a solution of $f(z) = z$. How many solutions does $f(z) = z$ have? Note that $f(0) = p^{k-1}$, $f(1) = 1$, and f is convex. Thus it has at least one and at most two solutions in $[0,1]$:



Two roots for $d = 3, k = 5, p = 0.8$



One root for $d = 3, k = 5, p = 0.8$

We see that $z = 1$ is the unique root of $f(z) = z$ in $[0,1]$ if and only if $f'(1) \leq 1$. Let us compute

$$f'(z) = (k-1)\left(p + (1-p)z^{d-1}\right)^{k-2}(1-p)(d-1)z^{d-2}$$
$$f'(1) = (k-1)(1-p)(d-1) \ .$$

So $f'(1) \leq 1$ if and only if $p \geq 1 - \frac{1}{(d-1)(k-1)}$.

It remains to show that if $p < 1 - \frac{1}{(d-1)(k-1)}$ and thus $f(z) = z$ has some root $z^* < 1$, then actually $z_p = z^*$ and not $z_p = 1$. Recall that $Y^{(h)}$ is the indicator variable which is 1 if after deletion, the component of T containing the root contains a vertex at level h. Let $y_p^{(h)} := \Pr[Y^{(h)}|p]$, $Z^{(h)} := 1 - Y^{(h)}$ and $z_p^{(h)} := 1 - y_p^{(h)}$. Since the root is on level 0 and never gets deleted, we have $z_p^0 = 0$. As we have seen above, $z_p^{(h)} \to z_p$, and obviously $z_p^{(h)}$ is increasing in h. Let us derive a recurrence for the $z_p^{(h)}$. Note that Z^h holds iff the following holds

for all children v of the root: (1) v is deleted, or (2) after deletion, each child w of v is the root of a subtree of height at most $h-2$. Observe that (1) holds with probability p and (2) with probability $\left(z_p^{h-2}\right)^{d-1}$. Thus,

$$z_p^{(h)} = \left(p + (1-p)\left(z_p^{(h-2)}\right)^{d-1}\right)^{k-1} = f\left(z_p^{(h-2)}\right) \ .$$

Thus we obtain $z_p = \lim_{h \to infty} z_p^{(h)}$ as the limit of the fixed point iteration of f, starting at 0. From the pictures above it should be clear that this converges to the smallest root.

Formally, let $z^* < 1$ be a root of $f(z) = z$, which exists and is unique if $p < 1 - \frac{1}{(k-1)(d-1)}$. We now prove by induction that $z_p^{(h)} \leq z^*$ for all even h. It is true for $h = 0$. Let us assume that $z_p^{(h)} \leq z^*$. Since $f(z)$ is increasing in z, we have $z_p^{(h+2)} = f\left(z^{(h)}\right) \leq f(z^*) = z^*$.

B.2 Making $S_{d,k}$ More Explicit

The reader who is happy with solving equation (7) numerically and then numerically integrating (6) to obtain $S_{d,k}$ may well skip the remainder of the section. Otherwise, he or she might want to continue as we outline how to compute $S_{d,k}$ somewhat more explicitly.

Note that we are given z_p implicitly, as the solution of a polynomial equation. This cannot be solved explicitly in general. However, we can compute the inverse function, namely $\rho(z)$: For given $z \in [0,1)$, $\rho(z)$ is the (unique) value p for which (7) holds. We can continuously extend ρ at point 1 by defining $\rho(1) := 1 - \frac{1}{(d-1)(k-1)}$. So ρ is continuous differentiable and monotone on $[0,1]$. We obtain:

$$\rho(z) = \frac{\sqrt[k-1]{z} - z^{d-1}}{1 - z^{d-1}} \ .$$

We abbreviate $m = (k-1)(d-1)$ and define $h_j : [0,1] \to [0,1]$ by $h_j(t) = t^{d-1-j}(1-t)^j$. We can now succinctly re-write (6):

$$\begin{aligned}
S_{d,k} &= \sum_{j=1}^{d-1} \log_d(1+j) \binom{d-1}{j} \int_0^1 y_p^j (1-y_p)^{d-1-j} dp \\
&= \sum_{j=1}^{d-1} \log_d(1+j) \binom{d-1}{j} \int_0^1 (1-z_p)^j z_p^{d-1-j} dp \\
&= \sum_{j=1}^{d-1} \log_d(1+j) \binom{d-1}{j} \int_0^1 h_j(z_p) dp \\
&= \sum_{j=1}^{d-1} \log_d(1+j) \binom{d-1}{j} \int_0^{1-1/m} h_j(z_p) dp \ .
\end{aligned}$$

The last line requires some justification: If $p \geq 1 - 1/m$, then $z_p = 1$ and thus $h_j(z_p) = 0$, thus values of p greater than $1 - 1/m$ contribute nothing to our expectation. We want to calculate the integral $\int_0^{1-1/m} h_j(z_p)dp$. Let us define $\zeta(p) := z_p$ to emphasize that we are talking about a function on the interval $[0, 1 - 1/m]$. We do not have an explicit formula for the integrand $h_j(\zeta(p))$, but we know the inverse function of ζ, namely $\rho(z)$. Thus we can apply the following substitution rule from calculus:

Lemma B.2. *Let I be a closed interval and $f : I \to \mathbb{R}$ be a continuous function and $\Phi : [a, b] \to \mathbb{R}$ a continuous differentiable function with $\phi([a, b]) \subseteq I$. Then*

$$\int_{\phi(a)}^{\phi(b)} f(x)dx = \int_a^b f(\phi(t))\phi'(t)dt \ .$$

We write $\zeta(p) := z_p$, $f(p) = h_j(\zeta(p))$, and $\phi(z) = \rho(z)$. Then f is continuous on $I = [0, 1 - 1/m]$ and ρ is continuous differentiable (and monotone) on $[a, b] = [0, 1]$. Note that $\phi([0, 1]) = [\phi(0), \phi(1)] = [0, 1 - 1/m] = I$, so all conditions of the lemma are satisfied. Therefore

$$\begin{aligned}
\int_0^{1-1/m} h_j(\zeta(p))dp &= \int_{\phi(a)}^{\phi(b)} f(p)dx \\
&= \int_a^b f(\phi(z))\phi'(z)dz \\
&= \int_0^1 h_j(\zeta(\rho(z)))\phi'(z)dz \\
&= \int_0^1 h_j(z)\phi'(z)dz \ .
\end{aligned}$$

Note that now we have an explicit formula for the integrand $h_j(z)\phi'(z)$. To simplify notation, we write $\Delta = d - 1$ and $\kappa := \frac{1}{k-1}$. We evaluate the integrand:

$$\begin{aligned}
h_j(z)\phi'(z) &= z^{d-1-j}(1-z)^j \left(\frac{\sqrt[k-1]{z} - z^{d-1}}{1 - z^{d-1}} \right)' \\
&= z^{\Delta - j}(1-z)^j \ \cdot \ \frac{z^{\kappa+\Delta}(\Delta - \kappa) + \kappa z^\kappa - \Delta z^\Delta}{z\left(1 - z^\Delta\right)^2}
\end{aligned}$$

Putting everything together we obtain

$$\begin{aligned}
S_{d,k} &= \sum_{j=1}^{d-1} \log_d(1+j) \binom{d-1}{j} \int_0^{1-1/m} h_j(z_p)dp \\
&= \sum_{j=1}^{\Delta} \log_{\Delta+1}(1+j) \binom{\Delta}{j} \int_0^1 z^{\Delta-j}(1-z)^j \ \cdot \ \frac{z^{\kappa+\Delta}(\Delta - \kappa) + \kappa z^\kappa - \Delta z^\Delta}{z\left(1 - z^\Delta\right)^2} dz
\end{aligned}$$

Although we still cannot give an elementary expression for the integral (not to speak of the sum), this is now explicit enough to be numerically calculated with

the help of a mathematics toolbox program, like maple or sagemath. We include the sage code for computing $S_{d,k}$:

```
% begin sagecode
def compute_S_DeltaKappa(Delta, kappa):
    total = 0
    j = 1
    while (j <= Delta):
        term1 = log(1 + j) / log(Delta + 1) * binomial(Delta, j)

        def helper(z):
            return z^(Delta-j)*(1-z)^j * (z^(kappa+Delta) * (Delta - kappa) +
                    kappa * z^kappa - Delta * z^Delta) / (z * (1-z^Delta)^2)

        term2 = numerical_integral(helper, 0, 1)[0]
        total = total + term1 * term2
        j = j+1

    return total

def compute_S(d,k):
    return compute_S_DeltaKappa(d-1, 1 / (k-1)).n(30)

def compute_basis(d,k):
    return (d^compute_S(d,k)).n(30)


% end sagecode
```

C Remaining Proofs for General (d, k)-ClSP

In case F has multiple satisfying assignments, we want to prove that the success probability of PPSZ is at least $d^{-G_{d,k}}$. In Section 4 we reduced this task to the one of showing the following inequality (2):

$$\mathop{\mathbf{E}}_{x\sim\xi}[\mathrm{cost}(\alpha_0,\alpha)-\mathrm{cost}(\alpha_0^x,\alpha)]-\mathop{\mathbf{E}}_{x\sim\xi}\left[\log_d a(x)\right]+\log_d\frac{s}{|\mathcal{U}|}\geq 0\ . \tag{8}$$

For definitions of cost, Q, ξ, $a(x)$ and so on please refer to Section 4. In this section we will prove Lemmas 4.4, 4.5, and 4.6, which bound the first, second, and third term of (2). To get started we have to collect some facts about cost and our distribution Q over satisfying assignments. In particular we want to understand how $\mathrm{cost}(\alpha_0,\alpha,y)$ and $\mathrm{Q}(\alpha_0,\alpha)$ change when passing from α_0 to α_0^x, that is, when fixing another variable x according to α.

Lemma C.1. *Let α_0 and α be fixed and compatible. For a variable variable $x\in\mathcal{U}$ let α_0^x denote the assignment $\alpha_0\cup\{x\mapsto\alpha(x)\}$. The following statements hold:*

(i) $\mathrm{Q}(\alpha_0^x,\alpha)\geq\mathrm{Q}(\alpha_0,\alpha)$*, and* $\mathrm{Q}(\alpha_0^x,\alpha)=\mathrm{Q}(\alpha_0,\alpha)$ *if x is frozen.*
(ii) $\mathrm{cost}(\alpha_0^x,\alpha,y)\leq\mathrm{cost}(\alpha_0,\alpha,y)$ *for any variable y.*

When choosing $x\in\mathcal{U}$ uniformly at random and setting it according to α, then

(iii) $\mathbf{E}_{x\in\mathcal{U}}[\mathrm{Q}(\alpha_0^x,\alpha)]=\frac{|\mathcal{S}_{\alpha_0}|}{|\mathcal{U}|}\,\mathrm{Q}(\alpha_0,\alpha)$.
(iv) the cost of a frozen non-forced variable $y\in\mathrm{V}_{\mathrm{fr}}$ decreases on expectation as

$$\mathop{\mathbf{E}}_{x\in\mathcal{U}}[\mathrm{cost}(\alpha_0^x,\alpha,y)]\leq\mathrm{cost}(\alpha_0,\alpha,y)-\frac{\log_d|a(y)|}{|\mathcal{U}|}.$$

Proof. (i) We prove the claim by induction over the size of α_0. The claim holds trivially if α_0 is a complete assignment. Otherwise we "unwrap" the definition of $\mathrm{Q}(\alpha_0,\alpha)$ and get

$$\begin{aligned}
|\mathcal{S}_{\alpha_0}|\cdot\mathrm{Q}(\alpha_0,\alpha)&=|\mathcal{S}_{\alpha_0}|\cdot\mathop{\mathbf{E}}_{(y,c)\in\mathcal{S}_{\alpha_0}}[\mathrm{Q}(\alpha_0^{y=c},\alpha)]=\sum_{(y,c)\in\mathcal{S}_{\alpha_0}}\mathrm{Q}(\alpha_0^{y=c},\alpha)\\
&=\sum_{y\in\mathcal{U}}\mathrm{Q}(\alpha_0^y,\alpha)\qquad\qquad\text{(since }\mathrm{Q}(\alpha_0^{y=c},\alpha)=0\text{ if }c\neq\alpha(y)\text{)}\\
&=\mathrm{Q}(\alpha_0^x,\alpha)+\sum_{y\in\mathcal{U}\setminus\{x\}}\mathrm{Q}(\alpha_0^y,\alpha)\ .
\end{aligned}$$

By the induction hypothesis we have $\mathrm{Q}(\alpha_0^y,\alpha)\leq\mathrm{Q}(\alpha_0^{y,x},\alpha)$ and thus

$$\begin{aligned}
|\mathcal{S}_{\alpha_0}|\cdot\mathrm{Q}(\alpha_0,\alpha)&\leq\mathrm{Q}(\alpha_0^x,\alpha)+\sum_{y\in\mathcal{U}\setminus\{x\}}\mathrm{Q}(\alpha_0^{y,x},\alpha)\\
&=\mathrm{Q}(\alpha_0^x,\alpha)+|\mathcal{S}_{\alpha_0^x}|\,\mathrm{Q}(\alpha_0^x,\alpha)\ .
\end{aligned}$$

The last equality follows from "wrapping in" the definition of $\mathrm{Q}(\alpha_0^x, \alpha)$. Recall that $\mathcal{S}_{\alpha_0}$ is the set of pairs (y, c) in $\mathcal{U}_{\alpha_0} \times [d]$ such that $F \wedge \alpha_0 \wedge (y = c)$ is satisfiable. Clearly $\mathcal{S}_{\alpha_0} \supseteq \mathcal{S}_{\alpha_0^x}$, but obviously $(x, \alpha(x)$ is in $\mathcal{S}_{\alpha_0}$ but not in $S_{\alpha_0^x}$, since x is already set in $S_{\alpha_0^x}$! Thus, $|\mathcal{S}_{\alpha_0}| \geq |\mathcal{S}_{\alpha_0^x}| + 1$ and therefore

$$|\mathcal{S}_{\alpha_0}| \cdot \mathrm{Q}(\alpha_0, \alpha) \leq \mathrm{Q}(\alpha_0^x, \alpha) + |\mathcal{S}_{\alpha_0^x}|\, \mathrm{Q}(\alpha_0^x, \alpha) =\leq |\mathcal{S}_{\alpha_0}| \cdot \mathrm{Q}(\alpha_0^x, \alpha)\ .$$

If x is frozen, then all inequalities in this proof can be replaced by equalities, which proves the equality statement in that case.

(ii) We consider the three cases: the variable y is non frozen, frozen or forced. Note that if $x = y$, the statement holds trivially.
If $y \in \mathrm{V_{nf}}$, then $\mathrm{cost}(\alpha_0, \alpha, y) = S$. Since the cost of a variable is always less than S, the statement holds.
If $y \in \mathrm{V_{fr}}$ or $y \in \mathrm{V_{fo}}$, then $\mathrm{cost}(\alpha_0, \alpha, y)$ is the expected logarithm of the number of non-forbidden values for y in the remainder of the PPSZ run. If we now fix another variable x to $\alpha(x)$, then this expectation cannot decrease, because adding a value assignment cannot allow a value that was forbidden.

(iii) We have

$$|\mathcal{S}_{\alpha_0}| \cdot \mathrm{Q}(\alpha_0, \alpha) = \sum_{(x,c) \in \mathcal{S}_{\alpha_0}} \mathrm{Q}(\alpha_0^{x=c}, \alpha) = \sum_{x \in \mathcal{U}} \mathrm{Q}(\alpha_0^x, \alpha) = |\mathcal{U}| \cdot \mathop{\mathbf{E}}_{x \in \mathcal{U}}[\mathrm{Q}(\alpha_0^x, \alpha)],$$

which proves the statement.

(iv) For a fixed, frozen, non-forced variable, we have that

$$\mathrm{cost}(\alpha_0, \alpha, y) = \mathop{\mathbf{E}}_{\pi}[\log_d(|\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|)].$$

Let π be a random permutation on V and let z be the variable that comes next (after α_0 has been assigned) in π. By the law of total expectation, we have that

$$\mathop{\mathbf{E}}_{\pi}[\log_d(|\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|)] = \mathop{\mathbf{E}}_{z}[\mathop{\mathbf{E}}_{\pi}[\log_d |\mathcal{A}(y, \alpha_0, \alpha, \pi, D)|] \mid z \text{ first in } \pi].$$

If $y = z$, then the expression in the expectation is $\log_d |\mathcal{A}(y, \alpha_0)|$. After that step, the cost of y is 0, so the overall cost in that case decreases by at least $\log_d |\mathcal{A}(y, \alpha_0)|$. This happens with probability $\frac{1}{|\mathcal{U}|}$, which yields the desired result.

□

Lemma 4.4, restated. *Choose $x \sim \xi$ and let $\alpha_0^x := \alpha_0 \cup \{x \mapsto \alpha(x)\}$. Then*

$$\mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0, \alpha) - \mathrm{cost}(\alpha_0^x, \alpha)] \geq \frac{1}{s} \left(G \sum_{y \in \mathrm{V_{nf}}} s(y) + \sum_{y \in \mathrm{V_{fr}}} \log_d a(y) \right)\ .$$

Proof. Recall that $\mathrm{cost}(\alpha_0, \alpha) = \sum_{y \in \mathcal{U}} \mathrm{cost}(\alpha_0, \alpha, y)$. Therefore

$$\begin{aligned}
\mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0^x, \alpha) - \mathrm{cost}(\alpha_0, \alpha)] &= \sum_{y \in \mathcal{U}} \left(\mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0, \alpha, y) - \mathrm{cost}(\alpha_0^x, \alpha, y)] \right) \\
&= \sum_{y \in \mathrm{V_{nf}}} \mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0, \alpha, y) - \mathrm{cost}(\alpha_0^x, \alpha, y)] \qquad (9) \\
&+ \sum_{y \in \mathrm{V_{fr}}} \mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0, \alpha, y) - \mathrm{cost}(\alpha_0^x, \alpha, y)] \ . \qquad (10)
\end{aligned}$$

The last equality holds since $\mathrm{cost}(\alpha_0^x, \alpha, y) = \mathrm{cost}(\alpha_0, \alpha, y) = 0$ for all $y \in \mathrm{V_{fo}}$. Let us first bound (9). We fix a variable $y \in \mathrm{V_{nf}}$. With probability $\xi(y)$, it happens that $x = y$ and then $\mathrm{cost}(\alpha_0, \alpha, y) = G$ and $\mathrm{cost}(\alpha_0^x, \alpha, y) = 0$. Otherwise $x \neq y$ and $\mathrm{cost}(\alpha_0, \alpha, y) \geq \mathrm{cost}(\alpha_0^x, \alpha, y)$, i.e., the cost does not increase. Thus,

$$\mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0, \alpha, y) - \mathrm{cost}(\alpha_0^x, \alpha, y)] \geq G \cdot \xi(y) = \frac{G\,\mathrm{Q}(\alpha_0^y, \alpha)}{s \cdot \mathrm{Q}(\alpha_0, \alpha)} \ .$$

Thus, summing over all y we obtain

$$(9) \geq \frac{G}{s \cdot \mathrm{Q}(\alpha_0, \alpha)} \cdot \sum_{y \in \mathrm{V_{nf}}} \mathrm{Q}(\alpha_0^y, \alpha) \ .$$

Claim: $\sum_{y \in \mathrm{V_{nf}}} \mathrm{Q}(\alpha_0^y, \alpha) = \mathrm{Q}(\alpha_0, \alpha) \sum_{y \in \mathrm{V_{nf}}} s(y)$.

This claim immediately shows that $(9) \geq \frac{G}{s} \cdot \sum_{y \in \mathrm{V_{nf}}} s(y)$.

Proof (Proof of the claim). Point (iii) of Lemma C.1 states that

$$s \cdot \mathrm{Q}(\alpha_0, \alpha) = \sum_{y \in \mathcal{U}} \mathrm{Q}(\alpha_0^y, \alpha)$$

Expanding s and noting that $\mathrm{Q}(\alpha_0^y, \alpha) = \mathrm{Q}(\alpha_0, \alpha)$ for all $y \in \mathcal{U} \setminus \mathrm{V_{nf}}$ we obtain

$$\mathrm{Q}(\alpha_0, \alpha) \cdot \sum_{y \in \mathcal{U}} s(y) = \sum_{y \in \mathrm{V_{nf}}} \mathrm{Q}(\alpha_0^y, \alpha) + \sum_{y \in \mathcal{U} \setminus \mathrm{V_{nf}}} \mathrm{Q}(\alpha_0, \alpha) \ .$$

Therefore we see that

$$\sum_{y \in \mathrm{V_{nf}}} \mathrm{Q}(\alpha_0^y, \alpha) = \mathrm{Q}(\alpha_0, \alpha) \sum_{y \in \mathcal{U}} (s(y) - \mathbb{I}_{y \notin \mathrm{V_{nf}}}) = \mathrm{Q}(\alpha_0, \alpha) \sum_{y \in \mathrm{V_{nf}}} s(y) \ ,$$

since $s(y) = 1$ for all $y \in \mathcal{U} - \mathrm{V_{nf}}$.

Now let us turn to (10). Again we fix some $y \in \mathrm{V_{fr}}$ and calculate:

$$\begin{aligned}
\mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0^x, \alpha, y)] &= \sum_{x \in \mathcal{U}} \frac{\mathrm{Q}(\alpha_0^x, \alpha)}{s \cdot \mathrm{Q}(\alpha_0, \alpha)} \cdot \mathrm{cost}(\alpha_0^x, \alpha, y) \\
&= \frac{|\mathcal{U}|}{s} \cdot \mathop{\mathbf{E}}_{x \in_{\mathrm{u.a.r.}} \mathcal{U}} \left[\frac{\mathrm{Q}(\alpha_0^x, \alpha)}{\mathrm{Q}(\alpha_0, \alpha)} \cdot \mathrm{cost}(\alpha_0^x, \alpha, y) \right] \ . \qquad (11)
\end{aligned}$$

We then invoke the following correlation inequality:

Proposition C.2. *Let $A, B \in \mathbb{R}$ be random variables and let A_ℓ be soem lower bound on A and B^u be some upper bound on B. Then*

$$\mathbf{E}[A \cdot B] \leq A_\ell \, \mathbf{E}[B] + B^u \, \mathbf{E}[A] - A_\ell B^u.$$

Proof. We can write $\mathbf{E}[A \cdot B] = \mathbf{E}[(A-a) \cdot B] + a \, \mathbf{E}[B]$ and then use $B \leq b$ and $A \geq a$ to obtain $\mathbf{E}[A \cdot B] \leq b \, \mathbf{E}[A - a] + aE[B] = b\bar{a} - ba + a\bar{b}$, as claimed. □

Set $A = \frac{\mathrm{Q}(\alpha_0^x, \alpha)}{\mathrm{Q}(\alpha_0, \alpha)}$ and $B = \mathrm{cost}(\alpha_0^x, \alpha, y)$. From Lemma C.1 we know that $A_\ell := 1$ is a lower bound on A and $B^u := \mathrm{cost}(\alpha_0, \alpha, y)$ is an upper bound on B; furthermore, that $\mathbf{E}[A] = \frac{s}{|\mathcal{U}|}$ and $\mathbf{E}[B] = \mathrm{cost}(\alpha_0, \alpha, y) - \frac{\log_d a(y)}{|\mathcal{U}|}$. We get that

$$\begin{aligned}
&\mathop{\mathbf{E}}_{x \in_{\text{u.a.r.}} \mathcal{U}} \left[\frac{\mathrm{Q}(\alpha_0^x, \alpha)}{\mathrm{Q}(\alpha_0, \alpha)} \cdot \mathrm{cost}(\alpha_0^x, \alpha, y) \right] \\
=& \left(\mathrm{cost}(\alpha_0, \alpha, y) - \frac{\log_d a(y)}{|\mathcal{U}|} \right) + \mathrm{cost}(\alpha_0, \alpha, y) \cdot \frac{s}{|\mathcal{U}|} - \mathrm{cost}(\alpha_0, \alpha, y) \\
=& \frac{s \cdot \mathrm{cost}(\alpha_0, \alpha, y) - \log_d a(y)}{|\mathcal{U}|}
\end{aligned}$$

Plugging this into (11) gives

$$\mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0^x, \alpha, y)] \leq \mathrm{cost}(\alpha_0, \alpha, y) - \frac{\log_d a(y)}{s}$$

and finally

$$(10) = \sum_{y \in \mathrm{V_{fr}}} \mathop{\mathbf{E}}_{x \sim \xi}[\mathrm{cost}(\alpha_0, \alpha, y) - \mathrm{cost}(\alpha_0^x, \alpha, y)] = \frac{1}{s} \cdot \sum_{y \in \mathrm{V_{fr}}} \log_d a(y) \ .$$

This finishes the proof of the lemma. □

Lemma 4.5, restated.

$$\mathop{\mathbf{E}}_{x \sim \xi}[\log_d a(x)] \leq \frac{\sum_{x \in \mathcal{U}} \log_d a(x)}{s} + \frac{\sum_{x \in \mathrm{V_{nf}}} (s(x) - 1)}{s} \ .$$

Proof. Writing out the expectation we get

$$\begin{aligned}
\mathop{\mathbf{E}}_{x \sim \xi}[\log_d a(x)] &= \sum_{x \in \mathcal{U}} \xi(x) \log_d a(x) \\
&= \sum_{x \in \mathcal{U}} \frac{\mathrm{Q}(\alpha_0^x, \alpha)}{s \cdot \mathrm{Q}(\alpha_0, \alpha)} \log_d a(x) \\
&= \frac{|\mathcal{U}|}{s} \mathop{\mathbf{E}}_{x \in_{\text{u.a.r.}} \mathcal{U}} \left[\log_d a(x) \cdot \frac{\mathrm{Q}(\alpha_0^x, \alpha)}{\mathrm{Q}(\alpha_0, \alpha)} \right].
\end{aligned}$$

We use the correlation inequality from Proposition C.2 with $A = \frac{\mathrm{Q}(\alpha_0^x,\alpha)}{\mathrm{Q}(\alpha_0,\alpha)}$, $A_l = 1$, $\mathbf{E}[A] = \frac{s}{|\mathcal{U}|}$, $B = \log_d a(x)$, $B^u = 1$, $\mathbf{E}[B] = \sum_{x\in\mathcal{U}} \frac{\log_d a(x)}{|\mathcal{U}|}$, and we get that

$$\begin{aligned}
\mathop{\mathbf{E}}_{x\sim\xi}[\log_d a(x)] &\le \frac{|\mathcal{U}|}{s} \cdot \mathop{\mathbf{E}}_{x\in_{\mathrm{u.a.r.}}\mathcal{U}} \left[\log_d a(x) \cdot \frac{\mathrm{Q}(\alpha_0^x,\alpha)}{\mathrm{Q}(\alpha_0,\alpha)}\right] \\
&\le \frac{|\mathcal{U}|}{s} \cdot \left(1 \cdot \sum_{x\in\mathcal{U}} \frac{\log_d a(x)}{|\mathcal{U}|} + \frac{s}{|\mathcal{U}|} \cdot 1 - 1 \cdot 1\right) \\
&= \frac{\sum_{x\in\mathcal{U}} \log_d a(x) + s - |\mathcal{U}|}{s} \,.
\end{aligned}$$

Finally observe that $s - |\mathcal{U}| = \sum_{x\in\mathcal{U}}(s(x)-1) = \sum_{x\in\mathrm{V_{nf}}}(s(x)-1)$, which yields the desired result. □

Lemma 4.6, restated. $\log_d \frac{s}{|\mathcal{U}|} \ge \log_d(e)\frac{\sum_{y\in\mathrm{V_{nf}}}(s(y)-1)}{s}$.

Proof. We use the inequality $\log_d\left(\frac{b}{a}\right) \ge \log_d(e) \cdot \frac{b-a}{b}$, which holds whenever $b \ge a$:

$$\begin{aligned}
\log_d\left(\frac{s}{|\mathcal{U}|}\right) &\ge \log_d(e) \cdot \frac{s - |\mathcal{U}|}{s} \\
&= \log_d(e) \cdot \frac{\sum_{y\in\mathcal{U}}(s(y)-1)}{s} \,.
\end{aligned}$$

Then note that $s(y) - 1 = 0$ for all $y \in \mathcal{U} \setminus \mathrm{V_{nf}}$. This is the claimed bound.

D Proof of Lemma 1.3

Lemma 1.3, restated. *If $k \geq 4$ then $S_{d,k} \geq 1 - \frac{1}{2\ln(d)}$ and $S_{d,k} = G_{d,k}$.*

Recall that $S_{d,k} = \mathbf{E}[\log_d(1+Y)]$ for Y defined in Section 1.3. A quick calculation shows that Lemma 1.3 is equivalent to

Theorem D.1. $\mathbf{E}[\ln(1+Y)] \geq \ln(d) - 1/2$.

It is difficult to obtain an explicit formula for the distribution of Y. Thus, our first goal will be to replace Y be a nicer random variable X such that Y dominates X. For this, we look at a different but equivalent random experiment defining the random variable Y.

D.1 The Random Experiment

Let T be a (possibly countably infinite) rooted tree. Choose $\pi : V(T) \to [0,1]$ uniformly at random. If every infinite path starting at the root contains a vertex v with $\pi(v) < \pi(\text{root})$, we say that *extinction* occurs and denote this event by $\mathcal{E}(T)$. If extinction does not occur, that is, if there is an infinite path starting with the root on which every vertex has a π-value of at least r, we say *survival* occurs, and denote this event by $\mathcal{S}(T)$. For two numbers $a, b \in [0,1]$ let $a \vee b := a+b-ab$. The following lemma is folklore in the theory of Galton-Watson branching processes:

Proposition D.2. *Let T_m be the infinite rooted m-regular tree, and let $r \in [0,1]$. Then $\Pr[\mathcal{E}(T_m) | \pi(\text{root}) = r]$ is the smallest root of the equation $x = (r \vee x)^m$.*

We denote this probability by $E_r(T_m)$. Let $d, k \in \mathbb{N}$, set $m = (d-1)(k-1)$ and let $T_{d,k}$ be the tree consisting of $k-1$ copies of T_m attached to a common root. So the root of $T_{d,k}$ has degree $k-1$, and every other vertex of $T_{d,k}$ has degree $m = (d-1)(k-1)$.

Proposition D.3. $\Pr[\mathcal{E}(T_{d,k}) | \pi(\text{root}) = r] = (r \vee E_r(T_m))^{k-1}$.

From now on we write $E_r := E_r(T_{d,k})$ and $S_r := 1 - E_r$. So S_r is the probability of survival in $T_{d,k}$. We define random variables $Y_1, \ldots, Y_{d-1}$ as follows: Sample $R \in [0,1]$ uniformly at random; then set each Y_i to be 1 with probability S_R and 0 with probability E_R. Set $Y = Y_1 + \cdots + Y_{d-1}$.

Observation D.4 *The random variable Y we just defined has the same distribution as the variable Y defined in Section 1.3.*

In fact, the only reason we re-define Y is that it will be easier to work with $T_{d,k}$ than with the odd-level even-level trees defined in Section 1.3.

D.2 Proof Outline.

It should be clear that increasing k makes survival more likely. Thus $\mathbf{E}[\ln(1+Y)]$ increases with k, and it suffices to prove Theorem D.1 for $k = 4$.

The first challenge is that we have no convenient formula for S_r, the probability of survival. Let X_c be 1 if in T_c, $\pi(root) \leq \pi(v)$ for at least one child v of the root. Note that $Y_c \leq X_c$. Let $X = X_1 + \cdots + X_{d-1}$, so $Y \leq X$. We will show that in some sense, Y is "not much smaller" than X. This allows us to relate $\mathbf{E}[\ln(1 + Y)]$ and $\mathbf{E}[\ln(1 + X)]$. The advantage is that the distribution of X is much easier to understand than that of Y.

In a second step we want to show that $\mathbf{E}[\ln(1 + X)] - \ln(d)$ is increasing in d. The problem: This is not true. As a way around this, we show that in fact it decreases very little, and the total decrease above some point d_0 sums up to something in $o(d_0)$. For this we use the fact that $(X_1 + \cdots + X_n)/n$ is "more concentrated" than $(X_1 + \cdots + X_{n-1})/(n-1)$, in a way we make formal towards the end.

In the end, we manage to bound $E[\ln(1+Y)] - \ln(d)$ from below by something *increasing* in d. For $d = 130$ this "something" is larger than $-1/2$, and thus we have shown Theorem D.1 $d \geq 130$. For $d \leq 130$ we use numerical computation to show that it holds.

D.3 An Upper Bound on $S_r(T_{d,k})$

Lemma D.5. $S_r \geq 1 - \left(\frac{rm}{m-1}\right)^{k-1}$.

Proof. Note that for $r \geq 1 - 1/m$ we have $S_r = 0$, and the right-hand side is at most 0. Thus, the lemma holds in this case, and we can focus on the case $r \leq 1 - 1/m$.

For two numbers $a, b \in [0, 1]$ we define $a \vee b := a + b - ab$. Note that $\frac{rm}{m-1} = r \vee p$ for $p = \frac{r}{(1-r)(m-1)}$, and $p \in [0, 1]$ for $0 \leq r \leq 1 - 1/m$. Proposition D.3 states that $E_r(T_{d,k}) = (r \vee E_r(T_m))^k$ and the statement of the lemma is equivalent to showing that $(r \vee E_r(T_m))^k \leq (r \vee p)^k$. Since $\vee$ is monotone in both arguments, this holds once we can show that $E_r(T_m) \leq p = \frac{r}{(1-r)(m-1)}$. Recall that $E_r(T_m)$ is the smallest root x of

$$x - (r \vee x)^m$$

is 0. At $x = 0$ the above expression is negative and therefore it is negative for all $x \in [0, E_r(T_m))$. Thus, if we can show that $p - (r \vee p)^m \geq 0$ we show that

$p \geq E_r(T_m)$ and are done. In fact:

$$\begin{aligned} p - (r + (1-r)p)^m &= \frac{r}{(1-r)(m-1)} - \left(r + (1-r)\frac{r}{(1-r)(m-1)}\right)^m \\ &= \frac{r}{(1-r)(m-1)} - \left(\frac{rm}{m-1}\right)^m \geq 0 \iff \\ \left(\frac{rm}{m-1}\right)^m &\leq \frac{r}{(1-r)(m-1)} \iff \\ r^{m-1}(1-r) &\leq \frac{1}{m}\left(\frac{m-1}{m}\right)^{m-1} . \end{aligned}$$

Elementary calculus shows that the left hand side achieves its maximum for $r = \frac{m-1}{m}$, at which point it equals the right-hand side. This finishes the proof. □

We want to show that $\mathbf{E}[\ln(1+Y)] \geq \ln(d) - 1/2$. Conditioned on $R = r$ for a specific $r \in [0,1]$, the Y_i are independent binary random variables with expectation S_r each. In what follows, we abbreviate $\mathbf{E}[\ldots|R=r]$ by $\mathbf{E}[\ldots|r]$ for expressions involving Y_i or X_i. Note that $\mathbf{E}[X_i|r] = 1 - r^{k-1}$.

Lemma D.6. *Let $r \leq 1 - 1/m$. Then $\mathbf{E}[\ln(1+Y)|r] \geq \mathbf{E}[\ln(1+X|\frac{mr}{m-1})]$.*

Proof. This holds since $\mathbf{E}[Y_i|r] = S_r \geq 1 - \left(\frac{mr}{m-1}\right)^{k-1} = \mathbf{E}[X_i|\frac{mr}{m-1}]$. □

With this lemma we can "sandwich" $\mathbf{E}[\ln(1+Y)]$ by two expressions involving $\mathbf{E}[\ln(1+X)]$:

Lemma D.7 (Sandwich Lemma). $\mathbf{E}[\ln(1+X)] - \frac{\ln(d)}{(d-1)(k-1)} \leq \mathbf{E}[\ln(1+Y)] \leq \mathbf{E}[\ln(1+X)]$.

Proof. The latter inequality is obvious since $Y \leq X$ for each point in the probability space. So let us prove the first inequality:

$$\begin{aligned} \mathbf{E}[\ln(1+Y)] &= \int_0^1 \mathbf{E}[\ln(1+Y)|r]dr \\ &= \int_0^{1-\frac{1}{m}} \mathbf{E}[\ln(1+Y)|r]dr \\ &\geq \int_0^{1-\frac{1}{m}} \mathbf{E}\left[\ln(1+X)\,\middle|\,\frac{rm}{m-1}\right] dr \\ &= \frac{m-1}{m}\int_0^1 \mathbf{E}[\ln(1+X)|r]dr \\ &\geq \int_0^1 \mathbf{E}[\ln(1+X)|r]dr - \frac{\ln(d)}{m} \\ &= \mathbf{E}[\ln(1+X)] - \frac{\ln(d)}{m} . \end{aligned}$$

The last inequality holds since the integrand is always at most $\ln(d)$. □

It is still difficult to bound $\mathbf{E}[\ln(1+X)]$ since we have no clear measure of how concentrated $1+X_1+\cdots+X_{d-1}$ is around its mean, even when conditioned on a particular r. We have just shown that

$$\mathbf{E}[\ln(1+Y)] - \ln(d) \geq \mathbf{E}\left[\ln\left(\frac{1+X}{d}\right)\right] - \frac{\ln(d)}{m} \ . \tag{12}$$

To prove Theorem D.1 we want to show that the above expression is at least $-1/2$. What we have gained so far is that the distribution of $(X_i)_{i\geq 1}$ is simpler to understand and does not depend on d.

An impractical approach Let $\mu(r) := \mathbf{E}\left[\frac{1+X}{d}\middle| r\right]$. We can show that (i) $\int_0^1 \mathbf{E}[\ln(\mu(r))]dr \approx -0.445$ as d grows; (ii) that $\frac{1+X}{d} \geq (1-\epsilon)\mu(r)$ except with probability $e^{-C\epsilon^2 d}$; conclude after some calculations that $\mathbf{E}[\ln(1+Y)] - \ln(d) \geq -0.445 + \ln(1-\epsilon) - \ln(d)e^{-C\epsilon^2 d}$. For all sufficiently large d, everything is less than $-1/2$. The trouble is the "sufficiently large" would mean $d \geq 100,000$, meaning we'd have verify our inequality numerically for all $d \leq 100,000$.

A more practical approach. We will show that as d grows, the random variable $\frac{1+X}{d-1}$ becomes "more concentrated" in a certain sense, while its expectation stays the same. This will *almost* allow us to argue that (12) increases with d (only *almost* because in (12) we have d in the denominator, not $d-1$). Then we simply have to verify that (12) $\geq -1/2$ holds for some value d_0 and infer that it holds for all $d \geq d_0$. For $d < d_0$ we can use numerical computation to verify that $\mathbf{E}[\ln(1+Y)] \geq \ln(d) - 1/2$.

First, although (12) is not necessarily increasing in d, we will show that it is "almost increasing". We define $f_r(d) = \mathbf{E}\left[\ln\left(\frac{1+X_1+\cdots+X_{d-1}}{d-1}\right)\middle| r\right]$ (yes, this is different from the integrand above) and $F(d) = \int_0^1 f_r(d)dr = \mathbf{E}\left[\ln\left(\frac{1+X_1+\cdots+X_{d-1}}{d-1}\right)\right]$.

Lemma D.8. *Let $d_0 \geq 3$. Then $F(d+1) \geq F(d_0+1) - \frac{H_{d_0}+1}{d_0}$ for all $d \geq d_0$.*

With this lemma we get, for all $d \geq d_0 \geq 3$:

$$\begin{aligned}
&\mathbf{E}[\ln(1+Y)] - \ln(d) && (13)\\
&\geq \mathbf{E}\left[\ln\left(\frac{1+X}{d}\right)\right] - \frac{\ln(d)}{m} && \text{(by (12) above)}\\
&= F(d) + \ln\left(\frac{d-1}{d}\right) - \frac{\ln(d)}{(d-1)(k-1)}\\
&\geq F(d_0) - \frac{H_{d_0}+1}{d_0} + \ln\left(\frac{d-1}{d}\right) - \frac{\ln(d)}{(d-1)(k-1)} && \text{(by Lemma D.8)}\\
&\geq F(d_0) - \frac{H_{d_0}+1}{d_0} + \ln\left(\frac{d_0-1}{d_0}\right) - \frac{\ln(d_0)}{(d_0-1)(k-1)} \ . && (14)
\end{aligned}$$

The program sagemath can give an explicit formula for the expression in (14), in terms of the so-called β-function[3]. Namely, for $k = 4$ we have

$$\begin{aligned}\Pr[X = j] &= \int_0^1 \binom{d-1}{j} \left(1 - r^3\right)^j \left(r^3\right)^{d-1-j} dr \\ &= \frac{1}{3} \cdot \binom{d-1}{j} \cdot \beta(d - j - 2/3, j + 1) \ .\end{aligned}$$

Thus we can "explicitly" compute (14) for any concrete integer d_0 (if β is sufficiently explicit) and then evaluate it numerically. Using sagemath, numerical evaluation suggests that (14) is greater than -0.483 for $d \geq 200$; this proves the lemma for all $d \geq 200$. For smaller values of d we compute $\mathbf{E}\left[\ln\left(\frac{1+X}{d}\right)\right] - \ln(d)/m$ numerically; it reaches a minimum of -0.453 at $d = 17$. This shows that $\mathbf{E}[\ln(1 + Y)] - \ln(d) \geq -1/2$ for all $k \geq 4$ and all d.

D.4 Proof of Lemma D.8

We start by bounding the difference between $f_r(d)$ and $f_r(d + 1)$ for fixed r. Until further notice, everything is conditioned on this specific value of r. We write $f(d), f(d + 1)$ for brevity.

$$\begin{aligned}f(d) &= \mathbf{E}\left[\ln\left(\frac{1+X}{d-1}\right)\right] \\ &= \mathbf{E}\left[\ln\left(\frac{1}{d} + \frac{X}{d-1}\right)\right] + \mathbf{E}\left[\ln\left(\frac{1+X}{d-1}\right)\right] - \mathbf{E}\left[\ln\left(\frac{1}{d} + \frac{X}{d-1}\right)\right]\end{aligned}$$

Lemma D.9 (Concentration of Binomial Distributions). *Let $(X_i)_{i\geq 1}$ be independent binary random variables with expectation p each, and define $W_n := \frac{X_1+\cdots+X_n}{n}$. Then for every concave function $f : [0, 1] \to \mathbb{R}$, the sequence $\mathbf{E}[f(W_n)]$ is increasing in n.*

Proof. Suppose $x = \frac{j}{n+1}$ for some integer $j \in \{0, \ldots, n + 1\}$, so $\Pr[W_{n+1} = x]$ is positive. Note that $\mathbf{E}[W_n | W_{n+1} = x] = x$, and therefore $\mathbf{E}[W_n | W_{n+1}] = W_{n+1}$. In other words, the sequence $W_k, W_{k-1}, W_{k-2}, \ldots, W_1$ is a martingale. Thus, together with Jensen's inequality we obtain:

$$\begin{aligned}\mathbf{E}[f(W_n)] &= \mathbf{E}[\mathbf{E}[f(W_n|W_{n+1})]] \\ &\leq \mathbf{E}[f(\mathbf{E}[W_n|W_{n+1}])] && \text{(Jensen's Inequality)} \\ &= \mathbf{E}[f(W_{n+1})] \ .\end{aligned}$$

[3] `https://en.wikipedia.org/wiki/Beta_function`

Since $x \mapsto \ln\left(\frac{1}{d}+x\right)$ is a concave function on $[0,1]$, we conclude that $\mathbf{E}\left[\ln\left(\frac{1}{d}+\frac{X_1+\cdots+X_{d-1}}{d-1}\right)\right] \leq \mathbf{E}\left[\ln\left(\frac{1}{d}+\frac{X_1+\cdots+X_d}{d}\right)\right] = f(d+1)$. Thus,

$$\begin{aligned}
f(d) &= \mathbf{E}\left[\ln\left(\frac{1+X}{d-1}\right)\right] \\
&= \mathbf{E}\left[\ln\left(\frac{1}{d}+\frac{X}{d-1}\right)\right] + \mathbf{E}\left[\ln\left(\frac{1+X}{d-1}\right) - \ln\left(\frac{1}{d}+\frac{X}{d-1}\right)\right] \\
&\leq f(d+1) + \mathbf{E}\left[\ln\left(\frac{1+X}{d-1}\right) - \ln\left(\frac{1}{d}+\frac{X}{d-1}\right)\right] \qquad \text{(Lemma D.9)} \\
&= f(d+1) + \mathbf{E}\left[\ln\left(1+\frac{1}{(d-1)(X+1)+X}\right)\right] \qquad \text{(short calculation)} \\
&\leq f(d+1) + \mathbf{E}\left[\frac{1}{(d-1)(X+1)}\right] \qquad \text{(since } \ln(1+x) \leq x \text{ and } X \geq 0\text{)} \\
&= f(d+1) + \frac{1}{d-1}\,\mathbf{E}\left[\frac{1}{X+1}\right] .
\end{aligned}$$

The inequality we have just derived holds for f_r, for every $r \in [0,1]$. To obtain an inequality for F, we have to integrate over r:

$$F(d) \leq F(d+1) + \frac{1}{d-1}\int_0^1 \mathbf{E}\left[\left.\frac{1}{X+1}\right| r\right] dr \ . \tag{15}$$

We will bound the integral from above. We sample $U_1 \ldots U_{d-1}$ as follows: Choose $r \in [0,1]$ and then set each U_i to 1 with probability $1-r$, uniformly at random, and set $U = U_1 + \cdots + U_{d-1}$. Clearly $\mathbf{E}[U_i|r] = 1-r \leq 1-r^3 = \mathbf{E}[X_i|r]$ and therefore $\mathbf{E}\left[\frac{1}{X+1}|r\right] \leq \mathbf{E}\left[\frac{1}{U+1}\right]$. Here is a combinatorial interpretation of U: Choose d values $u_1, \ldots, u_d$ uniformly at random and count how many are larger than u_d. The equivalence becomes clear once we condition on $u_d = r$: For $i < d$ we have $\Pr[u_i \geq u_d | u_d = r] = 1-r$. By symmetry, U is uniformly over $\{0, \ldots, d-1\}$, and therefore

$$\mathbf{E}\left[\frac{1}{1+U}\right] = \frac{1}{d}\sum_{i=1}^{d}\frac{1}{i} = \frac{H_d}{d} \ .$$

To summarize, we obtain

$$F(d) \leq F(d+1) + \frac{\ln(d)+1}{d(d-1)} \leq F(d+1) + \frac{H_d}{d(d-1)} \ . \tag{16}$$

By summation, we obtain for $d \geq d_0$:

$$\begin{aligned} F(d+1) &\geq F(d_0+1) - \sum_{k=d_0+1}^{d} \frac{H_k}{k(k-1)} \\ &= F(d_0+1) - \left(\frac{1}{d_0} - \frac{1}{d}\right)(H_{d_0}+1) + \frac{H_d - H_{d_0}}{d} \\ &\qquad\qquad \text{(several lines of computation)} \\ &\geq F(d_0+1) - \frac{H_{d_0}+1}{d_0} \ . \end{aligned}$$

This proves Lemma D.8.

E Asymptotic Behavior of PPSZ and PPZ for large d—Proof of Theorem 1.4

We want to define a notion of "savings", i.e., the advantage a (d,k)-ClSP algorithm has over the trivial d^n algorithm.

Definition E.1. *We say the* savings *of some algorithm for (d,k)-ClSP are c if its running time is $O^*(d^n/2^{cn})$.*

Observation E.2 *The savings of PPSZ for $(2,k)$-ClSP (i.e., for k-SAT) are $E_{2,k} := 1 - S_{2,k}$.*

Here, $E_{2,k}$ is the "extinction probability" in an infinite $k-1$-ary tree with $\pi(\text{root})$ uniformly at random in $[0,1]$.

Theorem 1.4 *For large d, the savings of PPSZ for (d,k)-ClSP converge to $\log_2(e)E_{2,k}$ where $E_{2,k} = -\int_0^1 \ln(1-r^{k-1})dr$.*

Thus, we see that the savings for large d are a constant factor ≈ 1.44 larger than for $d = 2$.

E.1 PPZ for Large d

Lemma E.3. *For large d, the savings of PPSZ converge to those of PPZ.*

Proof. The running time of PPZ is at most $d^{\mathbf{E}[\log_d(1+X)n]} = e^{\mathbf{E}[\ln(1+X)n]}$. Thus, the savings c of PPZ are given by

$$\frac{\ln(d) - \mathbf{E}[\ln(1+X)]}{\ln(2)} \ .$$

Similarly, the savings of PPSZ are $\frac{\ln(d)-\mathbf{E}[\ln(1+Y)]}{\ln(2)}$. By the Sandwich Lemma (Lemma D.7) from the previous section we get

$$|\mathbf{E}[\ln(1+Y)] - \mathbf{E}[\ln(1+X)]| \leq \frac{\ln(d)}{(k-1)(d-1)} \ .$$

Thus, the difference converges to 0 as d grows and so do the savings. □

From now on we can focus on the savings of PPZ. First we show that one additional application of Jensen's inequality is tight in the limit:

Lemma E.4. *The difference between $\mathbf{E}[\ln(1+X)] = \int_0^1 \mathbf{E}[\ln(1+X)|r]dr$ and $\int_0^1 \ln(1+\mathbf{E}[X|r])dr$ converges to 0 for large d.*

Proof. By Jensen's inequality, $\mathbf{E}[\ln(1+X)|r] \le \ln(1+\mathbf{E}[X|r])$, so the difference is always at most 0. Let us now prove an upper bound. Fix some $r \le 1 - \frac{1}{\sqrt[4]{d-1}}$ and write $\mathbf{E}[X|r] =: E$. Then for every $\epsilon \ge 0$ we have

$$\mathbf{E}[\ln(1+X)|r] \ge \ln(1+E(1-\epsilon)) \cdot (1 - \Pr[X \le E(1-\epsilon)]) \tag{17}$$

We bound the probability using Chebyshev's inequality:

$$\begin{aligned}
\Pr[X \ge E(1-\epsilon)] &\le \frac{\mathrm{Var}(X)}{\epsilon^2 E^2} \\
&\le \frac{d-1}{\epsilon^2 (d-1)^2 (1-r^{k-1})^2} \\
&\le \frac{1}{\epsilon^2 (d-1)(1-r)^2} && \text{(since } k \ge 2\text{)} \\
&\le \frac{1}{\epsilon^2 \sqrt{d-1}} && \text{(since } 1 - r \ge \tfrac{1}{\sqrt[4]{d-1}}\text{)}
\end{aligned}$$

We bound the first term of (17):

$$\ln(1+E(1-\epsilon)) = \ln(1+E) + \ln\left(1 - \frac{\epsilon E}{1+E}\right) \ge \ln(1+E) - 2\epsilon$$

for sufficiently small ϵ. Altogether we get the following bound:

$$\begin{aligned}
\mathbf{E}[\ln(1+X)|r] &\ge (\ln(1+E) - 2\epsilon) \cdot \left(1 - \frac{1}{\epsilon^2 \sqrt{d-1}}\right) \\
&\ge \ln(1+E) - 2\epsilon - \frac{\ln(d)}{\epsilon^2 \sqrt{d-1}} && \text{(since } \ln(1+E) \le \ln(d)\text{)}
\end{aligned}$$

This holds whenever $r \le 1 - \frac{1}{\sqrt[4]{d-1}}$. For larger values of r we simply observe that $\mathbf{E}[\ln(1+X)|r] \ge 0$ and $\ln(1+\mathbf{E}[X|r]) \le \ln(d)$. Altogether we get

$$\int_0^1 \mathbf{E}[\ln(1+X)|r]dr - \int_0^1 \ln(1+\mathbf{E}[X|r])dr \ge -2\epsilon - \frac{\ln(d)}{\epsilon^2\sqrt{d-1}} - \frac{\ln(d)}{\sqrt[4]{d-1}} \ .$$

For an appropriate choice of ϵ we see that the right-hand side converges to 0. □

We plug this convergence result into our estimate for $\mathbf{E}[\ln(1+X)]$:

$$\begin{aligned}
\mathbf{E}[\ln(1+X)] - \ln(d) &\approx \int_0^1 \ln(1+\mathbf{E}[X|r])dr - \ln(d) \\
&= \int_0^1 \ln(1+(d-1)(1-r^{k-1}))dr - \ln(d) \\
&= \int_0^1 \ln(d-(d-1)r^{k-1})dr - \ln(d) \\
&= \int_0^1 \ln\left(1 - \frac{d-1}{d} r^{k-1}\right) dr \ .
\end{aligned}$$

Let us abbreviate $a := k-1$.

Lemma E.5. *The integral* $\int_0^1 \ln\left(1-\frac{d-1}{d}r^a\right)dr$ *is decreasing in* d *and converges from above to* $\int_0^1 \ln\left(1-r^a\right)dr$.

Lemma E.6. $-\int_0^1 \ln\left(1-r^a\right)dr = E_{2,k}$.

Putting everything together we obtain an equation for the savings of PPSZ:

$$\lim_{d\to\infty}\frac{\ln(d)-\mathbf{E}[\ln(1+X)]}{\ln(2)} = \frac{E_{2,k}}{\ln(2)} \ ,$$

which proves the theorem.

E.2 Proof of Lemma E.6

Part of the following calculation is actually similar to the one computing the asymptotical running time of PPSZ for large k (as opposed to large d) in the original PPSZ paper [10]. Using the Taylor expansion $\ln(1-x) = -\sum_{n=1}^{\infty}\frac{x^n}{n}$ we obtain

$$\begin{aligned}
-\int_0^1 \ln\left(1-r^a\right)dr &= \int_0^1 \sum_{n=1}^{\infty}\frac{r^a n}{n}dr \\
&= \sum_{n=1}^{\infty}\int_0^1 \frac{r^a n}{n}dr \\
&= \sum_{n=1}^{\infty}\frac{1}{n(an+1)} \ .
\end{aligned}$$

Let us briefly focus on a single term in the sum:

$$\frac{1}{n(an+1)} = \frac{1}{n}-\frac{a}{an+1} = \frac{1}{n}-\frac{1}{n+1/a} = \int_0^1\left(t^{n-1}-t^{n+1/a-1}\right)dt \ .$$

Plugging this into our above calculation we get

$$\begin{aligned}
-\int_0^1 \ln\left(1-r^a\right) &= \sum_{n=1}^{\infty}\int_0^1\left(t^{n-1}-t^{n+1/a-1}\right)dt \\
&= \sum_{n=0}^{\infty}\int_0^1 t^n\left(1-t^{1/a}\right)dt \\
&= \int_0^1\left(1-t^{1/a}\right)\sum_{n=0}^{\infty}t^n dt \\
&= \int_0^1\frac{1-t^{1/a}}{1-t}dt \ .
\end{aligned}$$

We claim that this last expression is $E_{2,k} = 1-S_{2,k}$, the extinction probability in the $k-1$-ary tree. Recall that

$$S_{2,k} = \mathbf{E}[\log_2(1+Y_1)] = \Pr[Y_1 = 1] \ .$$

Here, Y_1 is the indicator variable that the tree T_1 after random deletion contains an infinite path starting at the root. This is the "survival probability". The extinction probability is $E_{2,k} := 1 - S_{2,k}$. Conditioned on $\pi(\text{root}) = r$ we see that $\Pr[Y_1 = 0 | \pi(\text{root}) = r]$ is the smallest root Q of the equation

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

Let us denote this value by $Q(r)$. We do not have a closed formula for $Q(r)$. However, we can easily solve the above equation for $r = r(Q)$:

$$r(Q) = \frac{Q^{1/(k-1)} - Q}{1 - Q} = \frac{Q^{1/a} - Q}{1 - Q} \ .$$

Note that $Q(0) = 0$, $Q(1) = 1$ and Q is non-decreasing for $r \in [0, 1]$. Thus, by a geometric consideration it becomes clear that

$$1 - S_{2,k} = E_{2,k} = \int_0^1 Q(r)dr = 1 - \int_0^1 r(Q)dQ = \int_0^1 \frac{1 - Q^{1/a}}{1 - Q} dQ = \int_0^1 \frac{1 - t^{1/a}}{1 - t} dt \ .$$

This finishes the proof of Lemma E.6. □

E.3 Proof of Lemma E.5

Write $L_d := \int_0^1 \ln\left(1 - \frac{d-1}{d} r^a\right) dr$ and $R := \int_0^1 \ln\left(1 - r^a\right) dr$ (L and R stand for left and right). By comparing the integrand pointwise it is clear that L_d is decreasing in d and $L_d \geq R$ for every d. Thererfore $\lim_{d\to\infty} L_d$ exists and is at least R.

We show that $\lim_{d\to\infty} \leq R$. Set $\epsilon := 1/d$ and consider any $\delta \in [0, 1)$:

$$L_d = \int_0^1 \ln\left(1 - (1-\epsilon) r^a\right) dr \leq \int_0^{1-\delta} \ln\left(1 - (1-\epsilon) r^a\right) dr$$

since the integrand is negative. On $[0, 1-\delta]$ the integrand converges uniformly to $\ln(1 - r^a)$ as $\epsilon \to 0$ and thus

$$\lim_{d\to\infty} L_d \leq \int_0^{1-\delta} \ln\left(1 - r^a\right) dr \ .$$

This inequality holds for every δ. For $\delta \to 0$ the right-hand side converges to R, which proves the lemma. □