Electronic Colloquium on Computational Complexity, Report No. 95 (2010)

A combinatorial analysis for the critical clause tree

Masaki Yamamoto*

Abstract

In [FOCS1998], Paturi, Pudlák, Saks, and Zane proposed a simple randomized algorithm for finding a satisfying assignment of a k-CNF formula. The main lemma of the paper is as follows: Given a satisfiable k-CNF formula that has a d-isolated satisfying assignment z, the randomized algorithm finds z with probability at least $2^{-(1-\mu_k/(k-1)+\epsilon_k(d))n}$, where $\mu_k/(k-1) = \sum_{i=1}^{\infty} 1/(i((k-1)i+1))$, and $\epsilon_k(d) = o_d(1)$. They estimated the lower bound of the probability in an analytical way, and used some asymptotics. In this paper, we analyze the same randomized algorithm, and estimate the probability in a combinatorial way. The lower bound we obtain is a little simpler: $2^{-(1-\mu_k(d)/(k-1))n}$, where $\mu_k(d)/(k-1) = \sum_{i=1}^{d} 1/(i((k-1)i+1))$. This value is a little bit larger than that of [FOCS98] although the two values are asymptotically same when $d = \omega(1)$.

1 Introduction

The satisfiability problem is, given a conjunctive normal form (abbrev. CNF) formula, a problem to determine whether the CNF formula is satisfiable. This is one of the well-known NP-complete problems even if the clause length is restricted to at most three. In this paper, we focus on the satisfiability problem where the clause length is at most a constant $k \geq 3$, which is denoted by k-SAT.

Initiated by Monien and Speckenmeyer [2], a number of exponential time algorithms for k-SAT have been proposed, and the exponential time complexity of k-SAT has been improved [2, 4, 3, 5, 1], to name but a few. In [4], Paturi, Pudlák, and Zane proposed a simple randomized algorithm for k-SAT. Given a k-CNF formula over n variables, the algorithm repeats the following procedure exponentially (in n) many times: Generate a permutation π on $\{1, ..., n\}$ and a 0/1 sequence $y \in \{0,1\}^n$. Then, apply the assignment y to variables in the order π as follows: Let $\varphi^{(1)} = \varphi$. For each $1 \leq i \leq n$, if a unit clause $(x_{\pi(i)})$ (resp. $(\bar{x}_{\pi(i)})$) appears in $\varphi^{(i)}$, then assign true (resp. false) to variable $x_{\pi(i)}$. Otherwise, assign $y_{\pi(i)}$ to $x_{\pi(i)}$. We define the new formula $\varphi^{(i+1)}$ as a formula obtained from $\varphi^{(i)}$ by assigning as above. We denote this procedure by $\texttt{assign}(\varphi, \pi, y)$. It is not difficult to see that given a satisfiable k-CNF formula φ that has a unique solution, the probability that $\texttt{assign}(\varphi, \pi, y)$ finds the unique solution is at least $2^{-(1-1/k)n}$. Thus, by repeating this procedure $\Omega(2^{(1-1/k)n})$ times, it finds the unique solution with high

*Tokai University, yamamoto@tokai-u.jp

ISSN 1433-8092

probability [1]. We denote this exponential-time algorithm by PPZ(φ). Note that PPZ(φ) does work for any CNF formula φ. Based on this algorithm, Paturi, Pudlák, Saks, and Zane [3] proposed an improved randomized algorithm: Given a k-CNF formula φ, it first preprocesses φ, and constructs a CNF formula. More precisely, it adds clauses to φ by resolution. Let $\texttt{resolve}(\varphi, s)$ be the preprocessing where resolution is applied to clauses of size at most $s/2$. Let φ_s be the CNF formula obtained by running $\texttt{resolve}(\varphi, s)$. Then, it runs PPZ($\varphi_s$). The main lemma of [3] is the following, where they say that for integer $d \geq 1$, a satisfying assignment z is d-isolated if there is no solution around z within the Hamming distance d.

Lemma 1.1. *[Paturi, Pudlák, Saks, and Zane [3]] Let φ be a satisfiable k-CNF formula. Let φ_s be a CNF formula obtained by running $\texttt{resolve}(\varphi, s)$. Suppose that φ has a d-isolated satisfying assignment z. Then, for $s \geq k^d$,*

$$\Pr_{\pi,y}\{\texttt{assign}(\varphi_s, \pi, y) \textit{ finds } z\} \geq 2^{-(1-\mu_k/(k-1)+\epsilon_k(d))n},$$

where

$$\begin{aligned} \mu_k &= \sum_{i=1}^{\infty} \frac{1}{i(i+1/(k-1))}, \\ \epsilon_k(d) &= \frac{3}{(d-1)(k-2)+2}. \end{aligned}$$

For proving this lemma, they [3] analyzed the probability that the assignment of a variable x is uniquely determined by the occurrence of the unit clause (x) or $(\bar{x})$. Let $C = (x_1 \vee \bar{x}_2 \vee \cdots \vee \bar{x}_k)$ be a clause of φ. Suppose that $z_1 = z_2 = \cdots = z_k = 1$. Then, C is "critical" at x_1 under z in a sense that what satisfies C under z is the assignment only to x_1. Thus, if $\pi^{-1}(i) < \pi^{-1}(1)$ for all $2 \leq i \leq k$ and $y_i = 1$ for all $2 \leq i \leq k$, the clause C becomes a unit clause (x_1), and hence the value of x_1 is uniquely determined so that C is satisfied. Intuitively, the probability that $\texttt{assign}(\varphi, \pi, y)$ finds z increases as the number of clauses increases keeping z to be a solution. That's why they [3] obtains φ_s instead of φ. From this, what we estimate is the probability for each variable x that there exists a critical clause in φ_s at x under z such that x is the last of all variables of C under the random order π. See the next section for the details.

For analyzing this probability, they introduced a labelled rooted tree. This is defined for each variable. Fix a variable x arbitrarily. Each node is labelled with a variable, and the root is labelled with x. We regard any node as a variable labelled the node. Then, roughly speaking, they observed that there exists a labelled rooted tree T_x such that any cut of T_x contains all variables of a critical clause of φ_s. They call it a "critical clause tree" for x. From this, what we estimate is the probability that there exists a cut in T_x such that x is the last of all variables of the cut under the random order π. See the next section for the details.

They estimated this probability in an analytical way, and used some asymptotics. In this paper, we estimate the probability in a combinatorial way. Through the combinatorial analysis, we obtain a little simpler lower bound on the probability. The following is our main result:

[1] Moreover, it is not too difficult to see that given a satisfiable k-CNF formula φ, it finds one of the satisfying assignments of φ in time $\Omega(2^{(1-1/k)n})$ with high probability. See [4] for the details.

Lemma 1.2. *Let φ be a satisfiable k-CNF formula. Let φ_s be a CNF formula obtained by running* `resolve`(φ, s). *Suppose that φ has a d-isolated satisfying assignment z. Then, for $s \geq k^d$,*

$$\Pr_{\pi,y}\{\texttt{assign}(\varphi_s, \pi, y) \text{ finds } z\} \geq 2^{-(1-\mu_k(d)/(k-1))n},$$

where

$$\mu_k(d) \;=\; \sum_{i=1}^{d} \frac{1}{i(i+1/(k-1))}.$$

The above formula of the lower bound of the probability is a little simpler than that of [3]. Moreover, as we will see in the concluding section, the value is a little bit larger than that of [3] although the two values are asymptotically same when $d = \omega(1)$. The purpose of this paper is to present another proof of the main lemma of [3], and to study combinatorial aspects of critical clause trees.

Our analysis

We give a combinatorial analysis of the probability. Fix a variable x arbitrarily. Let T_x be the critical clause tree for x. Our goal is to estimate the probability that for some cut A of T_x, all variables of A are before x under π. The idea of our combinatorial analysis is to exploit the following formula: for a finite sequence of events $E_1, E_2, ...,$

$$\Pr\left\{\bigvee_i E_i\right\} = \sum_i \Pr\left\{E_i \wedge \overline{(E_1 \vee \cdots \vee E_{i-1})}\right\}$$

Letting $F_i \stackrel{\text{def}}{=} E_i \wedge \overline{(E_1 \vee \cdots \vee E_{i-1})}$, we have $\Pr\{\bigvee_i E_i\} = \sum_i \Pr\{F_i\}$. Let $\mathcal{A}$ be the set of cuts of T_x. For each cut $A \in \mathcal{A}$, let E_A be the event that all variables of A are before x under π. Then, our goal is to estimate the probability $\Pr\{\bigvee_{A\in\mathcal{A}} E_A\}$. For using the above formula, we need to define a total order of cuts of T_x. The crucial point is the way of defining the order. See the section 3 for the precise definition. Let A_i be the ith cut under the total order, and let E_i be the event that all variables of A_i are before x under π. Then, $\Pr_\pi\{F_i\}$ can be expressed as a simple formula on $|A_i|$ and the number of nodes above A_i. (See Lemma 3.5.) Moreover, for any cuts A_i and A_j such that $|A_i| = |A_j|$, the numbers of nodes above A_i and A_j are same, and hence $\mathrm{P}\{F_i\} = \mathrm{P}\{F_j\}$. (See Lemma 3.7.) At this point, it remains to estimate the number of cuts of size i. We see that this number is related to the generalized Catalan number. (See Lemma 3.8.) Gathering the above lemmas, we can obtain our main result.

2 Preliminaries

2.1 PPSZ algorithm

Let $X = \{x_1, ..., x_n\}$ be a set of n Boolean variables. A *literal* is either a variable or the negation of a variable. A *clause* is a disjunction of literals. The size of a clause is the number of literals consisting of the clause. We alternatively consider a clause as the set

of literals, and hence we denote by $|C|$ the size of a clause C. We call a clause of size one a *unit clause*. A *conjunctive normal form* (abbrev. *CNF*) formula is a conjunction of clauses. We alternatively consider a CNF formula as the set of clauses, and hence we denote by $|\varphi|$ the number of clauses of a CNF formula φ. A *k-CNF* formula is a CNF formula that the size of every clause is at most k. Let φ be a k-CNF formula over X, and let t be a partial assignment to X. Then, we denote by $\varphi|_t$ a formula obtained from φ by applying t to φ.

We first present a simple randomized algorithm for SAT, denoted by PPZ, which was proposed by Paturi, Pudlák, and Zane [4]. The algorithm proposed by [3] is based on PPZ. Let φ be a CNF formula over X of n variables. The algorithm PPZ is shown in Fig. 1.

```
PPZ(φ, I)   // φ: a CNF formula over n variables, I: an integer

    for I times do
        pick a random permutation π on [n]
        pick a random 0/1 sequence y of length n
        t = assign(φ, π, y)
        if t satisfies φ, then output t

    output "not satisfiable"

assign(φ, π, y)

    set t = ∅
    for i : 1 ≤ i ≤ n
        if there is a unit clause x_π(i) in φ|_t, then set t_π(i) = 1
        else if there is a unit clause x̄_π(i) in φ|_t, then set t_π(i) = 0
        else t_π(i) = y_π(i)

    return t
```

Figure 1: PPZ algorithm

Given a satisfiable CNF formula φ, we estimate the probability that $\texttt{assign}(\varphi, \pi, y)$ returns a satisfying assignment of φ for a random permutation π and a random 0/1 sequence y. Let $\tau(\varphi)$ (resp. $\tau(\varphi, z)$) be the probability that $\texttt{assign}(\varphi, \pi, y)$ returns a satisfying assignment (resp. a satisfying assignment z) of φ (for random π and random y). Let $F(\varphi, \pi, y)$ be the set of variables the assignment of which is determined by the occurrence of a unit clause in running $\texttt{assign}(\varphi, \pi, y)$. Then, it is not difficult to see that $\tau(\varphi) = \sum_{z \in \mathrm{SAT}(\varphi)} \tau(\varphi, z)$, where $\mathrm{SAT}(\varphi)$ is the set of satisfying assignments of φ, and

$$\tau(\varphi, z) = \frac{1}{2^n n!} \sum_{\pi} 2^{|F(\varphi_s, \pi, z)|} = 2^{-n} \mathop{\mathrm{E}}_{\pi} \left[2^{|F(\varphi_s, \pi, z)|} \right].$$

See [3] for the proof of the above equations. By the convexity of the exponential function,

$$\tau(\varphi, z) \geq 2^{-n} \cdot 2^{\mathrm{E}_\pi[|F(\varphi,\pi,z)|]}.$$

By the following equation,

$$\mathrm{E}_\pi[|F(\varphi, \pi, z)|] = \sum_{x \in X} \Pr_\pi\{x \in F(\varphi, \pi, z)\}.$$

we will focus on lower bounds of $\Pr_\pi\{x \in F(\varphi, \pi, z)\}$ for any $x \in X$. For a satisfying assignment z of φ, we say that a clause $C \in \varphi$ is *critical at x under z* if C is not satisfied by an assignment z' which is obtained from z by flipping the value of x. For a satisfying assignment z of φ, we say that z is *d-isolated* if there is no satisfying assignment around z of distance at most d.

Let φ be a k-CNF formula over X of n variables. The algorithm proposed by [3] is PPZ(φ_s, I) where φ_s is a CNF formula obtained from φ by a "preprocessing". Let C and C' be two clauses for which there is a unique variable x such that C contains x and C' contains $\bar{x}$. Then, the *resolvent* $R(C, C')$ of C and C' is a clause $C \cup C' \setminus \{x\}$. We call such two clauses a *resolvable pair*. We say that a resolvable pair of C and C' is *s-bounded* if $|C| \leq s$, $|C'| \leq s$ and $|R(C, C')| \leq s$. The algorithm proposed by [3], denoted by PPSZ, is shown in Fig. 2. Similar to the analysis of PPZ, we will focus

PPSZ(φ, s, I) // φ: a k-CNF formula over n variables, s, I: integers

    $\varphi_s =$ `resolve`(φ, s)
    run PPZ(φ_s, I)

`resolve`(φ, s)

    let $\varphi_s = \varphi$
    while (there is a s-bounded resolvable pair (C, C') in φ_s
                such that $R(C, C') \notin \varphi$)
        $\varphi_s = \varphi_s \wedge R(C, C')$

    return φ_s

Figure 2: PPSZ algorithm

on lower bounds of $\Pr_\pi\{x \in F(\varphi_s, \pi, z)\}$ for any $x \in X$, where φ_s is given by running `resolve`(φ, s).

2.2 Tree cut

For estimating the probability $\Pr_\pi\{x \in F(\varphi_s, \pi, z)\}$, a labelled rooted tree was introduced by [3]. Such a rooted tree will be constructed for each variable. For defining it, we need some terminologies. The *degree* of a node in a rooted tree is the number of its children. The *depth* of a node is its distance from the root. A rooted tree is of uniform

depth d if every leaf is of depth d. A subset A of nodes is a *cut* if it does not include the root, and every path from the root to a leaf includes a node of A. In this paper, we only consider "minimal" cuts in the following sense: $A \setminus \{v\}$ is no longer a cut for any $v \in A$. That is, any two nodes v, v' of a cut, v is not an ancestor of v'.

Definition 2.1. *A rooted tree is said to be admissible if it has the following properties:*

- *the root is labelled with a variable.*
- *each node in the tree is either labelled with a variable or unlabelled.*
- *for any path from the root to a leaf, no two nodes in the path have the same label.*

Definition 2.2. *Let φ be a CNF formula over X, and let z be a satisfying assignment of φ. A rooted tree is said to be a critical clause tree with respect to φ, z, and a variable $x \in X$ if it is admissible, and it has the following properties:*

- *the root is labelled with x.*
- *for any cut A of the tree, φ has a critical clause C at x under z such that each variable of C except for x is a label of some node of A,*

We denote by $T_x(\varphi, z)$ such a critical tree with respect to φ, z, x.

Lemma 2.1. *[Paturi, Pudlák, Saks, and Zane: [3]] Let φ be a k-CNF formula over X, and let z be a d-isolated satisfying assignment of φ. For any variable $x \in X$, and for any $s \geq k^d$, there exists a critical clause tree $T_x(\varphi_s, z)$ which is of degree at most $k-1$ and of uniform depth d.*

By [3], considering the worst case, we may assume that all nodes of a critical clause tree are labelled with different variables. Moreover, we will show in the next section (just after the proof of Lemma 3.5) that the worst case is when every node of the tree is of degree exactly the maximum, i.e., $k-1$ for a k-CNF formula.

3 A combinatorial analysis

In this section, we prove Lemma 1.2. Given a satisfiable k-CNF formula φ over X, let x be an arbitrary variable of X. Suppose that φ has a d-isolated satisfying assignment z. We will estimate the probability $\Pr_\pi\{x \in F(\varphi_s, \pi, z)\}$, where φ_s is the output of $\texttt{resolve}(\varphi, s)$. Note from Lemma 2.1 that we are given a critical clause tree $T_x = T_x(\varphi_s, z)$ of degree at most $k-1$ and of uniform depth d. Moreover, we assume that all nodes of the tree are labelled with different variables. In what follows, we alternatively regard any node of T_x as a variable labelled the node. Thus, the root of T_x is x. We prove the following lemma in a combinatorial way, from which we easily derive Lemma 1.2.

Lemma 3.1. *Let φ be a satisfiable k-CNF formula, and let x be an arbitrary variable of φ. Let φ_s be a CNF formula obtained by running $\texttt{resolve}(\varphi, s)$. Suppose that φ has a d-isolated satisfying assignment z. Then, for $s \geq k^d$,*

$$\Pr_\pi\{x \in F(\varphi_s, \pi, z)\} \geq \sum_{i=1}^{d} \frac{1}{i((k-1)i+1)}.$$

As a first step to a combinatorial proof, we define a total order of cuts of T_x, denoted by $\leq_*$. Recall that we only consider minimal cuts. For any cut A of T_x, we denote by $T(A)$ the rooted subtree of T_x obtained by cutting below A. ($T(A)$ contains A.) Let A and A' be distinct cuts of T_x. (We define $A \leq_* A'$ for $A = A'$.) Let $T = T(A) \oplus T(A')$, where T is (a tree) induced by *edges* of T_x that are exclusively contained in $T(A) \cup T(A')$. Note that T is not empty, and does not contain the root of T_x. Thus, T is forest. Moreover, each (connected) component of T can be viewed as a rooted tree. Let W be the set of the roots of those rooted trees in T. Let $v \in W$ be the left-most node of W in T_x. Note that either A or A' contains v. Then, we define $A \leq_* A'$ if and only if A contains v. We show that $\leq_*$ is well-defined.

Proposition 3.2. *The set of cuts of T_x is totally ordered under $\leq_*$.*

Proof. It is obvious about anti-symmetry. It is also obvious about totality since we see from definition that the order between A and A' is uniquely determined for any distinct cuts A and A'. We show transitivity of $\leq_*$. Before that, we define some terminologies on rooted trees. Given a rooted tree, let v, v' be distinct nodes of the rooted tree such that v is not an ancestor of v', nor vice versa. Let w be the (unique) common ancestor of v and v' that has maximum depth. Let u (resp. u') be the child node of w that is an ancestor of v (resp. v'), or $u = v$ (resp. $u' = v'$). We say that v is *left* (resp. *right*) to v' if u is left (resp. right) to u'. Given a rooted tree, let v be a node of the rooted tree. Let L be the subtree of the rooted tree induced by all left nodes to v, all ancestors of v, and v itself. We call L the *left part* of v.

Let A, A', A'' be cuts of T_x such that $A \leq_* A'$ and $A' \leq_* A''$. We will show that $A \leq_* A''$. Let $T = T(A) \oplus T(A')$ and $T' = T(A') \oplus T(A'')$. As in definition, let W (resp. W') be the set of the roots of rooted trees in T (resp. T'). Let v (resp. v') be the left-most node of W (resp. W') in T_x. By definition, $v \in A$ and $v' \in A'$. First, we can discard the possibility that $v = v'$ or v' is an ancestor of v. Since $v \in A$, there are nodes of A' below v. On the other hand, $v' \in A'$. Thus, there are two nodes $u, u' \in A'$ such that u is an ancestor of u', contradicting to our assumption: we only consider minimal cuts. From this, we have three cases: (1) v is an ancestor of v', (2) v is left to v', and (3) v is right to v'.

Let $T'' = T(A) \oplus T(A'')$, and let W'' be the set of the roots of rooted trees in T''. Consider case (1), that is, v is an ancestor of v'. Let L (resp. L') be the left part of v (resp. v'). By definition, $T(A)$ and $T(A')$ (resp. $T(A')$ and $T(A'')$) are identical on L (resp. L'). Moreover, $L \subsetneq L'$. Thus, $T(A)$ and $T(A'')$ are identical on L, but they are different on the subtree below v. From this, we see that the left-most node of W'' in T_x is v. Since $v \in A$, we have $A \leq_* A''$. The above argument is similarly applied to case (2).

Consider case (3), that is, v is right to v'. Let L (resp. L') be the left part of v (resp. v'). Then, $T(A)$ and $T(A')$ (resp. $T(A')$ and $T(A'')$) are identical on L (resp. L'). Moreover, $L \supsetneq L'$. Thus, $T(A)$ and $T(A'')$ are identical on L', but they are different on parts below v'. From this, we see that the left-most node of W'' in T_x is v'. Moreover, $v' \in A$ since $T(A)$ and $T(A')$ are identical on L, and hence identical on $L' \subset L$. Thus, we have $A \leq_* A''$. Therefore, we show transitivity of $\leq_*$. □

We denote by V_i the ith cut under the total order $\leq_*$. Let π be the random permu-

tation on $[n]$. Let E_i be the event that all nodes of V_i are before x under π. Then,

$$x \in F(\varphi_s, \pi, z) \iff \bigvee_i E_i.$$

Thus, we will estimate the probability $\Pr_\pi\{\bigvee_i E_i\}$. Note that it suffices to consider permutations on variables labelled nodes of T_x, instead of all the variables $[n]$. For estimating $\Pr_\pi\{\bigvee_i E_i\}$, we exploit the following equation:

$$\bigvee_i E_i = \bigvee_i F_i, \quad \text{where } F_i \stackrel{\text{def}}{=} E_i \wedge \overline{(E_1 \vee \cdots \vee E_{i-1})}.$$

Note that any pair of F_i and F_j $(i \neq j)$ is disjoint. Thus,

$$\Pr_\pi\left\{\bigvee_i E_i\right\} = \Pr_\pi\left\{\bigvee_i F_i\right\} = \sum_i \Pr_\pi\{F_i\}.$$

Thus, we will estimate the probability $\Pr_\pi\{F_i\}$. Let U_i be the set of nodes above V_i except for the root. Before proving a lemma on the value of $\Pr_\pi\{F_i\}$, we present two propositions for the lemma.

Proposition 3.3. *For any two cuts V_i and V_j such that $T(V_j)$ is a subtree of $T(V_i)$, we have $V_j \leq_* V_i$, and hence $j \leq i$.*

Proof. It is obvious in case of $V_i = V_j$. Suppose that $V_i \neq V_j$. Let $T = T(V_i) \oplus T(V_j)$. Let W be the set of the roots of rooted trees of T. Note that $W \neq \emptyset$ since $V_i \neq V_j$. Let v be the left-most nodes of W in T_x. Since $T(V_j)$ is a subtree of $T(V_i)$, we have $v \in V_j$. Thus, $V_j \leq_* V_i$. □

Proposition 3.4. *Let V_i (resp. V_j) be ith (resp. jth) cuts, and let U_i (resp. U_j) be the set of nodes above V_i (resp. V_j). Then, for any i, j such that $j < i$, we have $V_j \cap U_i \neq \emptyset$.*

Proof. We prove its contraposition. Suppose that $V_j \cap U_i = \emptyset$. For any path from the root to a leaf, let $u \in V_i$ (resp. $v \in V_j$) be the node contained in the path. Since $V_j \cap U_i = \emptyset$, we have $u = v$ or u is an ancestor of v. This means that $T(V_i)$ is a subtree of $T(V_j)$. From Proposition 3.3, $V_i \leq_* V_j$, and hence $i \leq j$. □

Lemma 3.5. *For any $i \geq 1$,*

$$\Pr_\pi\{F_i\} = \frac{|V_i|! \cdot |U_i|!}{(|V_i| + |U_i| + 1)!}.$$

Proof. Fix i arbitrarily. The event F_i is the event that E_i occurs and any E_j for $1 \leq j \leq i-1$ does not occur. Note that E_i is the event that all nodes of V_i are before x. We consider permutations on $V_i \cup U_i \cup \{x\}$ rather than all nodes of T_x. The total number of permutations on $V_i \cup U_i \cup \{x\}$ is $(|V_i| + |U_i| + 1)!$. Now, we count the number of permutations such that all nodes of V_i are before x, and any E_j for $1 \leq j \leq i-1$ does not occur.

Claim 1. *Given that all nodes of V_i are before x,*

$$\overline{E_1 \vee \cdots \vee E_{i-1}} \iff \textit{all of } U_i \textit{ are after } x$$

Proof. ($\Rightarrow$) We prove its contraposition. That is, let $u \in U_i$ be a node that are before x. Let $V_i' \subset V_i$ be the set of nodes that are descendants of u. (Note that $V_i' \neq \emptyset$.) Let $W = \{u\} \cup (V_i \setminus V_i')$. Then, W is a cut such that $W \neq V_i$. Note here that all nodes of W are before x. Since $T(W)$ is a subtree of $T(V_i)$, by Proposition 3.3, we have $W \leq_* V_i$. Since $W \neq V_i$, we have $W = V_j$ for some $j < i$. Thus, since all nodes of V_j are before x, we have that E_j occurs.

($\Leftarrow$) We prove its contraposition. Suppose that $E_1 \vee \cdots \vee E_{i-1}$ holds. Consider that E_j occurs for some $j < i$. This means that all nodes of V_j are before x. On the other hand, from Proposition 3.4, we have $V_j \cap U_i \neq \emptyset$, which means that some node of U_i is before x. □

Thus, any permutation that we count in is a permutation such that all nodes of V_i are before x, and all nodes of U_i are after x. The number of those permutations is $|V_i|! \cdot |U_i|!$. This proves the lemma. □

From this lemma, we have

$$\Pr\left\{\bigvee_i E_i\right\} = \sum_i \frac{|V_i|! \cdot |U_i|!}{(|V_i| + |U_i| + 1)!}. \tag{1}$$

For estimating the value of the right-hand-side of the above, we will show that the worst case (i.e., the lowest value of $\Pr\{\bigvee_i E_i\}$) is when every node of T_x is of degree exactly $k-1$. This is proved using (1) as follows.

Suppose that T_x is not such a rooted tree. Let v be an arbitrary node the degree of which is less than $k-1$. Add a new child node to v, and then add a path to the child node so that the resulting rooted tree, denoted by T_x', is of uniform depth. Note that we may assume that all nodes of T_x' are labelled with different variables. We will show that the case of T_x' is worse than that of T_x. Let N be the number of cuts of T_x. Let $S \subset [N]$ be the set of indices i such that V_i contains neither v nor ancestors of v. Similar to V_i and E_i for T_x, let V_i' be the ith cut of T_x' under $\leq_*$, and let E_i' be the event that all nodes of V_i' are before x. Similar to N and S for T_x, let N' be the number of cuts of T_x', and let $S' \subset [N']$ be the set of indices i such that V_i' contains neither v nor ancestors of v. We easily see the following two facts.

Fact 1. *The two sets $\{V_i : i \in [N] \setminus S\}$ and $\{V_i' : i \in [N'] \setminus S'\}$ are identical.*

Fact 2. *For any $i \in S'$, let V be the projection of V_i' onto the nodes of T_x. Then, there is a unique $j \in S$ such that $V = V_j$. Moreover, $V_i' \setminus V_j = \{u\}$ for some u that is on the path added to T_x.*

From these two facts, it is not difficult to see that

$$\Pr\left\{\bigvee_{i \in [N']} E_i'\right\} = \sum_{i \in [N']} \frac{|V_i'|! \cdot |U_i'|!}{(|V_i'| + |U_i'| + 1)!}$$

$$= \sum_{i\in[N']\setminus S'} \frac{|V_i'|!\cdot|U_i'|!}{(|V_i'|+|U_i'|+1)!} + \sum_{i\in S'} \frac{|V_i'|!\cdot|U_i'|!}{(|V_i'|+|U_i'|+1)!}$$

$$= \sum_{i\in[N]\setminus S} \frac{|V_i|!\cdot|U_i|!}{(|V_i|+|U_i|+1)!} + \sum_{i\in S}\sum_{j=0}^{d'} \frac{(|V_i|+1)!\cdot(|U_i|+j)!}{((|V_i|+1)+(|U_i|+j)+1)!},$$

where d' is the length of the path added to T_x. Thus, it suffices to show that for any $i \in S$ and for any $d' \geq 0$,

$$\frac{|V_i|!\cdot|U_i|!}{(|V_i|+|U_i|+1)!} \geq \sum_{j=0}^{d'} \frac{(|V_i|+1)!\cdot(|U_i|+j)!}{((|V_i|+1)+(|U_i|+j)+1)!}.$$

Proposition 3.6. *For any $d' \geq 0$, and for $a \geq 0$ and $b \geq 0$,*

$$\frac{a!\cdot b!}{(a+b+1)!} \geq \sum_{j=0}^{d'} \frac{(a+1)!\cdot(b+j)!}{(a+1+b+j+1)!}.$$

Proof. We re-define a as $a-1$. (Thus, $a \geq 1$.) Then, the inequality is equivalent to

$$\frac{b}{a} \geq \sum_{j=0}^{d'} \frac{(b+j)(b+j-1)\cdots(b+1)b}{(a+b+j+1)(a+b+j)\cdots(a+b+1)}. \tag{2}$$

For $d'=0$, the right-hand-side of (2) is $b/(a+b+1)$, which is less than b/a. Suppose that $d' \geq 1$. Then, the right-hand-side of (2) is

$$\frac{b}{a+b+1} + \sum_{j=1}^{d'} \frac{(b+j)(b+j-1)\cdots(b+1)b}{(a+b+j+1)(a+b+j)\cdots(a+b+1)}$$

$$= \frac{b}{a+b+1}\left(1 + \sum_{j=1}^{d'} \frac{(b+j)(b+j-1)\cdots(b+1)}{(a+b+j+1)(a+b+j)\cdots(a+b+2)}\right).$$

Thus, (2) with $d' \geq 1$ is equivalent to

$$\frac{b}{a} \geq \frac{b}{a+b+1}\left(1 + \sum_{j=1}^{d'} \frac{(b+j)(b+j-1)\cdots(b+1)}{(a+b+j+1)(a+b+j)\cdots(a+b+2)}\right),$$

which is equivalent to

$$\frac{b+1}{a} \geq \sum_{j=1}^{d'} \frac{(b+j)(b+j-1)\cdots(b+1)}{(a+b+j+1)(a+b+j)\cdots(a+b+2)}.$$

Applying the above arguments repeatedly, (2) is reduced to the following inequality:

$$\frac{b+d'}{a} \geq \frac{b+d'}{a+b+d'+1}.$$

It is obvious that the above inequality holds, and hence this proves the proposition. □

From the above, we see that the case of T_x is worse than that of T'_x. Applying this argument repeatedly until every node of the resulting rooted tree is of degree exactly $k-1$, we see that the case of the final tree is the worst. Thus, in what follows, we assume that every node of T_x is of degree $k-1$.

Lemma 3.7. *Let V be an arbitrary cut of T_x, and let U be the set of nodes above V. Then,*

$$|V| = (k-2)(|U|+1)+1.$$

Proof. We prove it by induction on $|U|$. It is obvious for $|U|=0$ since the cut V such that $|U|=0$ is the set of all children of x. Suppose that the equation in the lemma holds for any cut such that the number of nodes above the cut is ℓ. Let V be an arbitrary cut such that $|U|=\ell+1$. Note that there exists a node u in U such that all children of u are in V. This is because otherwise we can find a path from the root to a leaf that does not contain any node of V. Let V' be the cut obtained from V by adding u and erasing all children of u. Let U' be the set of nodes above V'. Note that $|U'|=\ell$. By induction, $|V'|=(k-2)(\ell+1)+1$. Thus,

$$\begin{aligned} |V| &= |V'|+(k-1)-1 \\ &= (k-2)(\ell+1)+1+k-2 \\ &= (k-2)((\ell+1)+1)+1 \\ &= (k-2)(|U|)+1)+1. \end{aligned}$$

□

In the above arguments, we have focused on cuts of T_x. For any cut, we call the set of nodes above the cut its *upper set*. From now on, we switch our attention from cuts to their upper sets. Note that there is a one-to-one correspondence between cuts and their upper sets. Furthermore, observe that any upper set and the root x (uniquely) induce a connected subtree of T_x, which is also rooted at x, and any two upper sets induce distinct connected subtrees. On the other hand, for any connected subtree of T_x which is rooted at x, there is an upper set that induces the connected subtree. Thus, there is also a one-to-one correspondence between upper sets and connected subtrees of T_x that are rooted at x. From this, since the size of connected subtrees of T_x which are rooted at x takes the value from $\{1,2,...\}$, we see that the size of an upper set takes the value from $\{0,1,2,,...\}$ while the size of a cut takes from $\{(k-2)i+k+1 : i \in \{0,1,2,,...\}\}$.

From this observation, for estimating the right-hand-side of (1), we classify cuts according to the size of their upper sets. Let s_i be the number of cuts V such that the number of nodes above V is i. The following lemma is obtained from the "generalized Catalan number". See Appendix for a proof.

Lemma 3.8. *For $i \geq 0$,*

$$s_i = \frac{((k-1)(i+1))!}{((k-2)(i+1)+1)! \cdot (i+1)!}.$$

Theorem 3.9. *For any $d \geq 1$,*

$$\Pr\left\{\bigvee_i E_i\right\} \geq \sum_{i=1}^{d} \frac{1}{i((k-1)i+1)}.$$

Proof. From the equation (1) and Lemma 3.8,

$$\begin{aligned}
\Pr\left\{\bigvee_{i\in[N]} E_i\right\} &= \sum_{i\in[N]} \frac{|V_i|!\cdot|U_i|!}{(|V_i|+|U_i|+1)!} \\
&\geq \sum_{i=0}^{d} \frac{((k-2)(i+1)+1)!\cdot i!}{((k-2)(i+1)+1+i+1)!}\cdot s_i \\
&= \sum_{i=0}^{d} \frac{((k-2)(i+1)+1)!\cdot i!}{((k-2)(i+1)+1+i+1)!}\cdot\frac{((k-1)(i+1))!}{((k-2)(i+1)+1)!\cdot(i+1)!} \\
&= \sum_{i=0}^{d} \frac{1}{(i+1)((k-1)(i+1)+1)} \\
&\geq \sum_{i=1}^{d} \frac{1}{i((k-1)i+1)}.
\end{aligned}$$

□

4 Concluding remarks

We have estimated the probability that the randomized procedure finds a d-isolated satisfying assignment. Below, we compare two exponents of the probabilities in this paper and [3]:

$$\begin{aligned}
&\sum_{i=1}^{d} \frac{1}{i((k-1)i+1)} - \left(\sum_{i=1}^{\infty} \frac{1}{i((k-1)i+1)} - \frac{3}{(d-1)(k-2)+2}\right) \\
=\ & \frac{3}{(d-1)(k-2)+2} - \sum_{i=d+1}^{\infty} \frac{1}{i((k-1)i+1)} \\
\geq\ & \frac{3}{(d-1)(k-2)+2} - \frac{1}{k-1}\sum_{i=d+1}^{\infty} \frac{1}{i^2} \\
\geq\ & \frac{3}{(d-1)(k-2)+2} - \frac{1}{d(k-1)} \quad \left(\because \sum_{i=d+1}^{\infty} \frac{1}{i^2} \leq \frac{1}{d}\right) \\
>\ & 0.
\end{aligned}$$

From this, we see that our result is a little bit better than [3] although the two results are asymptotically same when $d = \omega(1)$.

References

[1] K. Iwama and S. Tamaki, "Improved upper bounds for 3-SAT", In Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA04), p. 328, 2004.

[2] B. Monien and E. Speckenmeyer, "Solving satisfiability in less than 2^n steps", Discrete Applied Mathematics 10, pp. 287-295, 1985.

[3] R. Paturi, P. Pudlák, M. Saks, and F. Zane, "An Improve Exponential-Time Algorithm for k-SAT", In Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS1998), pp. 628-637, 1998. (This is the preliminary version of J. of the ACM 52(3), pp. 337-364, 2005.)

[4] R. Paturi, P. Pudlák, and F. Zane, "Satisfiability coding lemma", In Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS1997), pp. 566-574, 1997.

[5] U. Schöning, "A probabilistic algorithm for k-SAT and constraint satisfaction problems", In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS1999), pp. 410-414, 1999.

[6] J. H. van Lint and R. M. Wilson, A Course in Combinatorics, Cambridge University Press, 2001.

Appendix

A proof of Lemma 3.8

For $i \geq 0$, let b_i be the number of $(k-1)$-ary rooted trees of i nodes including the root. We define $b_0 = 1$ for convenience. We first observe that $s_{i-1} = b_i$ for $i \geq 1$, and hence we estimate b_i for $i \geq 0$. For this, let $B(z) = \sum_{i=0}^{\infty} b_i z^i$ be the generating function for b_i. Then, it is easy to see that

$$B(z) = 1 + z \cdot \{B(z)\}^{k-1}.$$

For analyzing $B(z)$, we make use of the Lagrange inversion formula (see [6], for example):

Theorem 4.1 (Lagrange inversion formula)**.** *Let $f(w)$ be an analytic function at $w = 0$ with $f(0) \neq 0$. Suppose that $z = w/f(w)$. Then, $w = \sum_{i=1}^{\infty} c_i(0) z^i$, where*

$$c_i(w) = \frac{1}{i!} \left(\frac{d}{dw} \right)^{i-1} (f(w))^i.$$

Let $C(z) = B(z) - 1$. Then,

$$z = \frac{C(z)}{(C(z)+1)^{k-1}}.$$

Applying the above theorem to $C(z)$ with $w = C(z)$ and $f(w) = (w+1)^{k-1}$, we obtain

$$\begin{aligned} c_i(w) &= \frac{1}{i!} \left(\frac{d}{dw} \right)^{i-1} (w+1)^{(k-1)i} \\ &= \frac{1}{i!} \underline{((k-1)i)^{i-1}} (w+1)^{(k-1)i-(i-1)}. \end{aligned}$$

From this, we obtain

$$\begin{aligned} c_i(0) &= \frac{1}{i!} \underline{((k-1)i)^{i-1}} \\ &= \frac{((k-1)i)!}{((k-2)i+1)! \cdot i!}. \end{aligned}$$

Thus,

$$C(z) = \sum_{i=1}^{\infty} c_i(0) z^i = \sum_{i=0}^{\infty} \frac{((k-1)i)!}{((k-2)i+1)! \cdot i!} \cdot z^i.$$

From this, we have $b_0 = 1$, and

$$b_i = \frac{((k-1)i)!}{((k-2)i+1)! \cdot i!}.$$

Therefore,

$$s_i = b_{i+1} = \frac{((k-1)(i+1))!}{((k-2)(i+1)+1)! \cdot (i+1)!}.$$

ECCC ISSN 1433-8092

http://eccc.hpi-web.de