Local Enumeration and Majority Lower Bounds

Mohit Gurumukhani
Cornell University, Ithaca, NY, USA

Ramamohan Paturi
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, USA

Pavel Pudlák
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Michael Saks
Department of Mathematics, Rutgers University, Piscataway, NJ, USA

Navid Talebanfard
University of Sheffield, Sheffield, UK
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Abstract

Depth-3 circuit lower bounds and k-SAT algorithms are intimately related; the state-of-the-art Σ_3^k-circuit lower bound (Or-And-Or circuits with bottom fan-in at most k) and the k-SAT algorithm of Paturi, Pudlák, Saks, and Zane (J. ACM'05) are based on the same combinatorial theorem regarding k-CNFs. In this paper we define a problem which reveals new interactions between the two, and suggests a concrete approach to significantly stronger circuit lower bounds and improved k-SAT algorithms. For a natural number k and a parameter t, we consider the $\textsc{Enum}(k,\ t)$ problem defined as follows: given an n-variable k-CNF and an initial assignment α, output all satisfying assignments at Hamming distance $t(n)$ of α, assuming that there are no satisfying assignments of Hamming distance less than $t(n)$ of α. We observe that an upper bound $b(n,k,t)$ on the complexity of $\textsc{Enum}(k,\ t)$ simultaneously implies depth-3 circuit lower bounds and k-SAT algorithms:

- **Depth-3 circuits:** Any Σ_3^k circuit computing the Majority function has size at least $\binom{n}{\frac{n}{2}}/b(n,k,\frac{n}{2})$.
- **k-SAT:** There exists an algorithm solving k-SAT in time $O\left(\sum_{t=1}^{n/2} b(n,k,t)\right)$.

A simple construction shows that $b(n,k,\frac{n}{2}) \geq 2^{(1-O(\log(k)/k))n}$. Thus, matching upper bounds for $b(n,k,\frac{n}{2})$ would imply a Σ_3^k-circuit lower bound of $2^{\Omega(\log(k)n/k)}$ and a k-SAT upper bound of $2^{(1-\Omega(\log(k)/k))n}$. The former yields an unrestricted depth-3 lower bound of $2^{\omega(\sqrt{n})}$ solving a long standing open problem, and the latter breaks the Super Strong Exponential Time Hypothesis.

In this paper, we propose a randomized algorithm for $\textsc{Enum}(k,\ t)$ and introduce new ideas to analyze it. We demonstrate the power of our ideas by considering the first non-trivial instance of the problem, i.e., $\textsc{Enum}(3,\ \frac{n}{2})$. We show that the expected running time of our algorithm is 1.598^n, substantially improving on the trivial bound of $3^{n/2} \simeq 1.732^n$. This already improves Σ_3^3 lower bounds for Majority function to 1.251^n. The previous bound was 1.154^n which follows from the work of Håstad, Jukna, and Pudlák (Comput. Complex.'95).

By restricting ourselves to monotone CNFs, $\textsc{Enum}(k,\ t)$ immediately becomes a hypergraph Turán problem. Therefore our techniques might be of independent interest in extremal combinatorics.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases Depth 3 circuits, k-CNF satisfiability, Circuit lower bounds, Majority function

Digital Object Identifier 10.4230/LIPIcs.CCC.2024.17

Related Version *Full version*: https://arxiv.org/abs/2403.09134

Funding *Mohit Gurumukhani*: Supported by NSF CAREER Award 2045576 and a Sloan Research Fellowship.

Ramamohan Paturi: Partially supported by NSF grant 2212136.

Pavel Pudlák: Partially supported by grant EXPRO 19-27871X of the Czech Grant Agency and the institute grant RVO: 67985840.
Navid Talebanfard: This project has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101106684 — EXCICO. Views and opinions expressed are however those of author(s) only and do not necessarily reflect those of the European Union or REA. Neither the European Union nor the granting authority can be held responsible for them.

Funded by
the European Union

1 Introduction

Local search is a fundamental paradigm in solving the satisfiability problem: find an assignment close in Hamming distance to the initial assignment that satisfies the formula, if one exists. Papadimitriou [18] was the first to employ this idea in a randomized poly-time 2-SAT algorithm. Schöning [24] showed that a slight modification of this algorithm yields a running time of $(2-2/k)^n$ for k-SAT. Dantsin et al. [4] considered a deterministic version of local search and gave a deterministic $(2-2/(k+1))^n$ time algorithm. Brueggemann and Kern [2] and Kutzkov and Scheder [13] improved this deterministic local search procedure and obtained faster deterministic 3-SAT algorithms. Moser and Scheder [17] eventually considered a variant of the local search problem and used it to give a deterministic k-SAT algorithm matching the running time of Schöning's.

Despite the success of local search, the fastest known k-SAT algorithm PPSZ and its improvements follow a different approach: pick a random variable x, if its value is not easily seen to be forced[1] then assign it randomly and continue (see [20, 19, 10, 8, 22]). The analysis of this simple yet powerful algorithm consists of a combinatorial theorem relating the size and the structure of the set of satisfying assignments of a k-CNF. The strength of this combinatorial theorem is further manifested by the fact that the state-of-the-art depth-3 circuit lower bounds are built on it [20, 19].

This curious interaction between lower bounds and algorithms has become less of a surprise over the years. Williams [27] initiated a whole new line of inquiry by showing that improved satisfiability algorithms for a circuit class automatically imply lower bounds for the same class. Conversely, almost all known circuit lower bound techniques have been adopted in satisfiability algorithms (see e.g. [11, 3]). Within this context the role of local search is unclear.

▶ **Question 1.** *Can we derive lower bounds from local search algorithms?*

This question is also motivated by a lack of progress in improving depth-3 circuit lower bounds and related upper bounds on k-SAT algorithms.

Depth-3 circuit lower bounds. A Σ_3^k circuit is an Or-And-Or circuit where the bottom fan-in is bounded by k, i.e., a disjunction of k-CNFs. We use $\Sigma_3^k(f)$ to denote the minimum number of k-CNFs in a Σ_3^k circuit computing a function f. The study of these circuits was advocated by Valiant [25] who showed that a strong enough Σ_3^k lower bound for every fixed k implies a super-linear lower bound for series-parallel circuits. Moreover, [7] showed that strong lower bounds even for small constant k such as $k=16$ imply various circuit lower

[1] For example if x appears in a unit clause, or if such a clause can be derived in small width resolution.

bounds including better general circuit lower bounds. The technique of [20] gives a lower bound of $\Omega(2^{n/k})$ for the parity function and it is known to be tight. In fact this results in a $\Omega(n^{\frac{1}{4}}2^{\sqrt{n}})$ lower bound for computing parity by unrestricted depth-3 circuits which is tight up to a constant factor. A further improvement comes from [19] which gives a lower bound of $2^{cn/k}$ where $c > 1$ for the BCH code. At this point, this is the best known lower bound for computing any explicit function by Σ_3^k circuits.

Majority is a natural candidate for going beyond the $2^{\Omega(\sqrt{n})}$ depth-3 circuit lower bound. The natural Σ_3^k circuit for computing Majority has size $2^{O(n\log(k)/k)}$ (which implies an unrestricted depth-3 size upper bound of $2^{O(\sqrt{n\log n})}$). Håstad, Jukna and Pudlák [9] introduced the intriguing notion of *k-limits* to capture the depth-3 complexity of various functions and proved a lower bound of $2^{\Omega(\sqrt{n})}$ where the constant factor in the exponent is improved over the constant one could obtain from Switching Lemma. Regarding Σ_3^k circuits computing Majority, their result implies size lower bounds of 1.414^n for $k = 2$, and 1.154^n for $k = 3$, and for $k \geq 4$ it yields nothing. The Σ_3^2 bound is known to be essentially tight [21]. More recently, [14] proved a tight lower bound of $2^{\Omega(n\log(k)/k)}$ for computing Majority by Σ_3^k circuits where each And gate depends on at most k variables. Further, [1] studied the effect of negations for Σ_3^k circuits computing majority. However, the question of proving tight lower bounds for computing Majority by depth-3 circuits (even for fan-in 3 circuits) remains open.

k-SAT upper bounds: Lack of progress in improving the savings beyond $\Omega(\frac{1}{k})$ for k-SAT algorithms led researchers to consider SSETH (Super Strong Exponential Time Hypothesis). SSETH is the hypothesis that k-SAT cannot be solved with *savings* asymptotically more than $1/k$, i.e., there is no $2^{(1-\epsilon_k)n}$ time k-SAT algorithm with $\epsilon_k = \omega(1/k)$. However, SSETH is known to be false on average [26, 15], that is, the satisfiability of almost all k-CNFs can be decided with much larger savings. It is thus not unreasonable to attempt to get such savings even in the worst-case. Yet we cannot hope to achieve such a savings for a large subclass of PPSZ-style algorithms [23].

It appears that making progress towards larger k-SAT savings as well as depth-3 circuit lower bounds requires new ideas. We argue that local search has the potential to achieve this goal, and as an evidence for this claim, we apply local search ideas to give a new Σ_3^3 lower bound for Majority function.

1.1 Local enumeration, k-SAT and Σ_3^k lower bounds for Majority function

The local search problem is formally defined as follows.

(k, t)-SAT. Given an n-variable k-CNF F, a parameter t, and an assignment α, decide if there is a satisfying assignment β of F such that $d(\alpha, \beta) \leq t(n)$, where $d(\cdot)$ is the Hamming distance.

Dantsin et al. [4] gave a simple branching algorithm which solves (k, t)-SAT in time $\text{poly}(n) \cdot k^t$. This already gives a non-trivial algorithm for 3-SAT: solve $(k, \frac{n}{2})$-SAT starting with the all-0 and all-1 assignments. To get a non-trivial algorithm for larger k, they used *covering codes*, i.e., a small and asymptotically optimal number $C(n, t)$ of Hamming balls of a given radius t that cover the entire n-dimensional Boolean cube. Then an upper bound of $C(n, r) \cdot k^t$ follows immediately for k-SAT by solving (k, t)-SAT starting with the centers

of each of the balls in the covering code. Setting $t = \frac{n}{k+1}$ minimizes this quantity. Thus improved upper bounds for $(k,\ t)$-SAT immediately imply improved k-SAT upper bounds, and indeed this is what [2, 13] did by proving an upper bound of c^t for some $c < 3$ for $(3,\ t)$-SAT. However, this improvement in local search is not sufficient to yield better upper bounds for k-SAT when we want to use the technique for large t. It is conceivable that improved bounds for large t combined with covering codes would yield improved k-SAT algorithms. This leads to the following question:

▶ **Question 2.** *What is the complexity of $(k,\ \epsilon n)$-SAT where $0 < \epsilon \leq \frac{1}{2}$?*

It is also natural to consider the enumeration problem for $(k,\ t)$-SAT: enumerate all satisfying assignments within Hamming distance t of an initial assignment. We note that even a weaker form of this problem already captures the circuit complexity of Majority function. For this purpose, we introduce the following class of parameterized problems.

Enum$(k,\ t)$. Given an n-variable k-CNF F and an initial assignment α, output all satisfying assignments of F at a Hamming distance t from α assuming that there are no satisfying assignments of F at a Hamming distance of less than t from α.

We observe that upper bounds on Enum$(k,\ t)$ imply depth-3 circuit lower bounds and k-SAT algorithms.

▶ **Proposition 1.** *Assume that Enum$(k,\ t)$ can be solved in randomized expected time $b(n, k, t)$. Then*

1. *any Σ_3^k circuit requires at least $\binom{n}{n/2}/b(n, k, \frac{n}{2})$ size for computing Majority function.*
2. *k-SAT can be solved in time $O(\sum_{t=1}^{n/2} b(n, k, t))$.*

Proof. 1) Consider a Σ_3^k circuit C that computes Majority function. We will write $C = \bigvee_{i=1}^{m} F_i$, where each F_i is a k-CNF. None of the F_is has a satisfying assignment with Hamming weight less than $n/2$. By assumption we can enumerate all satisfying assignments of Hamming weight exactly $n/2$ in expected time $b(n, k, \frac{n}{2})$. This in particular implies that the total number of such satisfying assignments for each F_i is at most $b(n, k, \frac{n}{2})$. Since F_is should together cover all assignments of Hamming weight exactly $n/2$ and since there are $\binom{n}{n/2}$ such assignments, the claim follows.

2) We can trivially check if there is a satisfying assignment of Hamming weight at most $n/2$ in $1 + \sum_{t=1}^{n/2} b(n, k, t)$ steps. In the same number of steps we can check if there is a satisfying assignment of Hamming weight at least $n/2$. ◀

Observe that $b(n, k, \frac{n}{2})$ cannot be too small: define the k-CNF $\mathsf{Maj}_{n,k}$ by partitioning the n variables into sets of size $2(k-1)$ and by including all positive clauses of size k from each of the parts. It is easy to see that every satisfying assignment of this formula must set at least $k-1$ variables in each part to 1 and the total number satisfying assignments with Hamming weight $n/2$ is $2^{(1-O(\log(k)/k))n}$, thus $b(n, k, \frac{n}{2}) \geq 2^{(1-O(\log(k)/k))n}$. It follows that a matching upper bound for Enum$(k,\ \frac{n}{2})$ refutes SSETH and gives Σ_3^k lower bound of $2^{\Omega(\frac{\log k}{k} n)}$ for Majority which in turn implies a $2^{\omega(\sqrt{n})}$ unrestricted depth-3 circuit lower bound, breaking a decades long barrier.

1.2 Our contributions

In this paper, we study Enum$(3,\ \epsilon n)$ and obtain new algorithms and lower bounds. Note that this is the first non-trivial instance of Enum$(k,\ t)$, since Enum$(2,\ t)$ can be solved in

2^t steps using a simple extension of the local search algorithm, and it is easy to see that this is tight for $t \leq n/2$ by considering the 2-CNF consisting of t disjoint monotone clauses.

▶ **Theorem 2** (Main result). *ENUM*$(3,\ t)$ *can be solved in expected time*

1. 3^t, *for* $t \leq \frac{n}{3}$,
2. $1.164^n \times 1.9023^t$, *for* $\frac{n}{3} < t \leq \frac{3n}{7}$,
3. $1.1962^n \times 1.7851^t$, *for* $\frac{3n}{7} < t \leq \frac{n}{2}$.

In particular, ENUM$(3,\ \frac{n}{2})$ *can be solved in expected time* 1.598^n.

Consequently, we get

▶ **Corollary 3.** $\Sigma_3^3(\mathsf{Maj}) \geq 1.251^{n-o(n)}$.

Our lower bound is the best known bound to compute Majority function by Σ_3^3 circuits. Note that $\mathsf{Maj}_{n,3}$ has $6^{n/4} \simeq 1.565^n$ satisfying assignments of Hamming weight $n/2$. Our bound is not too far from the optimal bound and it is a substantial improvement over $3^{n/2} \simeq 1.732^n$.

In the following, we explain how our approach to enumeration differs from the well-known approaches. The k^t algorithm used by [4] which solves $(k,\ t)$-SAT is a simple branching procedure. Without loss of generality assume that the initial assignment is all-0. If there is no monotone clause in the formula, then the all-0 assignment satisfies the formula. Otherwise select a monotone clause $C = x_1 \vee \ldots \vee x_k$. Then for each x_i, recursively solve $(k,\ (t-1))$-SAT for the formula restricted by $x_i = 1$. An obvious weakness of this algorithm is that if the depth of the recursion tree is more than n/k, then some assignments will be considered more than once in the tree thus leading to redundant computation.

Our starting point is to search the recursion tree (which we call a transversal tree following the hypergraph nomenclature) so each satisfying assignment (which we call a transversal in the following) within the ball is visited exactly once. We observe that such a non-redundant search can be conducted with any clause ordering and any fixed ordering of variables within the clause. During the search of the transversal tree, it is easy to decide whether a subtree contains any new satisfying assignments by considering the labels of the child edges of the nodes along the path. If there are no new satisfying assignments in a subtree, we will prune it. It turns out that this approach is isomorphic to the seminal method of Monien-Speckenmeyer [16] where for each i we recursively solve the problem under the restriction $x_1 = \ldots = x_{i-1} = 0, x_i = 1$. However, it is not clear how to improve upon the bound obtained by [16]. We show that by choosing the clause ordering carefully and randomly ordering clause variables, a better bound can be obtained. In other words, we will consider *randomized* Monien-Speckenmeyer trees with careful clause ordering. The crux of our contribution is a new analysis of randomized transversal trees.

Connection to hypergraph Turán problems. Recall that a transversal in a hypergraph is a set of vertices that intersects every hyperedge. The recent work [6] gives a connection between depth-3 circuits and transversals[2]. Here we find another connection. Given n, t, and k, let $R^+(n, t, k)$ be the maximum number of transversals of size t in an n-vertex k-graph with no transversal of size $t-1$. Since a k-graph can be viewed as a monotone k-CNF,

[2] [6] results are stated in terms of cliques which are dual to transversals.

our algorithm enumerates minimum size transversals and thus gives new upper bounds for $R^+(n,t,k)$. Mantel's theorem, which is a special case of Turán's seminal theorem, gives the exact value $R^+(n,n-2,2) = n^2/4$, and the Turán problem for 3-graphs can be phrased as showing $R^+(n,n-3,3) = (5/9+o(1))n^3$ (see [12] for a thorough survey of this and related problems). Our technique allows us to derive new bounds for $R^+(n,t,3)$, where $t = \Theta(n)$. Although we currently do not get any useful results for $t = n - o(n)$, we hope that our techniques can be extended to make progress for this regime of parameters.

Enumeration algorithms for CNFs with bounded negations. As an additional application of our enumeration techniques, we get the following enumeration algorithm:

▶ **Theorem 4.** *Let F be a CNF of arbitrary width where either each clause contains at most 3 negative literals or each clause contains at most 3 positive literals. Then, we can enumerate all minimal satisfiable solutions of F in time $O(1.8204^n)$.*

2 Preliminaries

In this section, we introduce concepts and notation that we use for the rest of the paper. Let $F = (X, \mathcal{C})$ be a k-CNF with variable set X and clause set $\mathcal{C}$. We view the satisfying assignments of F as subsets of variables which are set to 1.

▶ **Definition 5** (Transversals). *A set $S \subseteq X$ is a transversal for F if the assignment that sets the variables in S to 1 and the variables in $X \setminus S$ to 0 is a satisfying assignment of F. The size of a transversal S is defined as $|S|$.*

We say S is a minimal transversal for F if no subset of S is a transversal. We call the corresponding satisfying asssignment a minimal satisfying assignment.

Our use of transversals for discussing satisfying assignments is motivated by the notion of transversals in hypergraphs. We view a monotone k-CNF (where all literals in every clause are positive) as a k-graph where every hyperedge has size at most k. This leads to a 1-1 correspondence between the transversals of a monotone k-CNF and those of the corresponding hypergraph.

In this paper, we are primarily interested in minimum-size transversals.

▶ **Definition 6** (Transversal number). *For a satisfiable k-CNF F, we define* transversal number $\tau(F)$ *to be the cardinality of the minimum-size transversal of F. We use $\Gamma(F)$ to denote the set of all minimum-size transversals of F and $\#\Gamma(F)$ to denote the cardinality of $\Gamma(F)$.*

Let $t \in [n]$ and $\alpha \in \{0,1\}^n$ be such that every satisfying assignment of F is at a distance of at least t from α. We reduce the Enum$(k,\ t)$ problem for F from the initial assignment α to the Enumz(k) problem: enumerate all minimum-size transversals of a k-CNF[3]. Indeed, let G be the k-CNF formula (over the variable set X) where its clause set is obtained from that of F by the following replacement of literals: For each variable $v \in X$, if α sets v to 1, then we swap occurrences of the positive and the negative literals corresponding to v in $\mathcal{C}$. Otherwise, if α sets v to 0, we leave the corresponding literals as they are. G is also a k-CNF

[3] This problem has been previously considered by, e.g. [5] with a different name Min-Ones k-SAT and a non-trivial algorithm is given independent of $\tau(F)$. Here we focus on running times with fine dependence on $\tau(F)$.

and for all $y \in \{0,1\}^n$, $G(y) = F(y \oplus \alpha)$. Clearly, there exists a transversal within distance t from 0^n in G if and only if there exists a transversal within distance t from α in F.

We now prove the following useful proposition that allows us to assume all clauses of F have width exactly k.

▶ **Proposition 7.** *For every k-CNF $F = (X, \mathcal{C})$ with $\tau(F) \leq n - k$, there exists a k-CNF G where every clause has width exactly k and $\tau(G) = \tau(F)$ and the set of transversals of G includes the set of transversals of F. Furthermore, if F is monotone, G is monotone.*

Proof. Assume that F has a clause C of width $1 \leq k' < k$. Let F' be the formula obtained from F by removing the clause C and adding a clause $C' = C \cup C''$ for each $S \subseteq X \setminus C, |S| = k - k'$, where C'' is a monotone clause over variables in S. The proposition follows since every transversal of F is a transversal of F' and any transversal of F' which is not a transversal of F must have size at least $n - k + 1$. ◀

3 Transversal Trees and Tree Search

In this section, we present an algorithm called TreeSearch for solving Enumz(k), i.e., to enumerate all minimum-size transversals of a satisfiable k-CNF F. Let $t = \tau(F)$ be the transversal number of F. Our algorithm considers a tree, called *transversal tree*, of depth t where each minimum-size transversal corresponds to at least one leaf node at depth t. Our algorithm TreeSearch traverses the tree to enumerate the leaves at depth t corresponding to minimum-size transversals. However, a leaf at depth t need not correspond to a minimum-size transversal and furthermore two distinct leaves at depth t may correspond to the same minimum-size transversal. For these reasons, enumerating all leaves can take significantly more time than the total number $\#\Gamma(F)$ of minimum-size transversals. We deal with this issue by pruning subtrees so that TreeSearch would only encounter minimum-size transversals that are not encountered elsewhere. While our pruning approach is isomorphic to that of Monien-Speckenmeyer [16], our analysis and bound crucially depend on our choice of clause ordering and random ordering of the child nodes of the transversal tree. In the following, we define the required concepts and present our constructions.

▶ **Definition 8** ((k, X)-trees). *For $k \geq 1$ and a set X of variables, a (k, X)-tree is a directed k-ary tree T with a node and edge labeling Q and a tree-edge ordering π which satisfies the following properties.*

1. *Each edge is directed from parent to child. Each non-leaf node has at most k children. Children of a node are ordered from left to right according to π.*
2. *Each (tree) edge e is labeled with a variable $q_e \in X$. Each node v is labeled with a set $Q_v \subseteq X \cup \{\bot\}$.*
3. *For each node, labels of child edges are distinct. If $e = (u, v)$ is an edge, then $Q_v = Q_u \cup \{q_e\}$ or $Q_v = Q_u \cup \{q_e, \bot\}$.*
4. *Labels of edges along any path are distinct.*
5. *All leaves v where $\bot \notin Q_v$ are at the same level.*

Let T be a (k, X)-tree with root r and labeling Q. For node v of T, let T_v denote the subtree T_v of T rooted at v. If u and v are nodes of T such that u is an ancestor of v, P_{uv} denotes the unique path from u to v in T.

▶ **Definition 9** (Shoot of a tree path). *If u and v are nodes of T such that u is an ancestor of v, the subgraph consisting of all the child edges of the nodes along the unique path from u to v is called the* shoot $S_{uv} = S^T_{uv}$ *from u to v. In particular $P_{uv} \subseteq S_{uv}$.*

For a path P_{uv}, the labels of the edges along the path are called *path variables* of P_{uv}. For a shoot S_{uv}, labels of the shoot edges are called shoot variables.

▶ **Definition 10** (Transversal tree). *Let $F = (X, \mathcal{C})$ be a k-CNF on the variable set X. A (k, X)-tree T rooted at r with labeling Q and tree-edge ordering π is a* transversal *tree for F if*

1. *$Q_r = \emptyset$ if F has no empty clause and otherwise $Q_r = \{\perp\}$, and*
2. *for every node v, each minimum-size transversal of F which is an extension of Q_v will appear as the label of a leaf in the subtree rooted at v.*

It is easy to see that a transversal tree T for a satisfiable k-CNF F has depth $\tau(F)$ and every leaf v of T such that $\perp \notin Q_v$ is at depth $\tau(F)$. We also note that any subtree of a transversal tree is also a transversal tree.

▶ **Definition 11** (Valid and invalid leaves). *Let T be a transversal tree for a satisfiable k-CNF F. We say that a leaf v of T is* valid *if Q_v is a minimum-size transversal of F. Otherwise it is* invalid.

For a transversal tree T of a satisfiable k-CNF F, let $\Gamma(T)$ denote the collection of minimum-size transversals associated with the valid leaves of T. The definition of transversal tree implies the following basic fact.

▶ **Fact 1.** $\Gamma(F) = \Gamma(T)$.

3.1 Construction of Transversal Trees

In this section, we show how to construct transversal trees for a satisfiable k-CNF $F = (X, \mathcal{C})$. The construction produces a labeling Q on nodes and edges. We will not specify a tree-edge ordering in the construction. However, we will later select a left-right ordering where child nodes are ordered randomly and independently for each node. The construction depends on the ordering of clauses. Let Π denote an ordering of the clauses in F.

We start with the tree T with just one node r (the root node) with the label $Q_r = \emptyset$. Assume that we are about to expand a non-leaf node v. By construction, v is at depth less than $\tau(F)$ and $\perp \notin Q_v$. Since v is at depth less than $\tau(F)$, Q_v is not a transversal. Select the first monotone clause $C_v = \{a_1, \ldots, a_{k'}\}$ according to the clause order Π, where $k' \leq k$. Such a monotone clause must exist since every clause is non-empty and since otherwise an all-0 assignment will satisfy the formula contradicting the fact that Q_v is not a transversal. Also, it must be the case that $C_v \cap Q_v = \emptyset$ as none of the variables from Q_v can appear in C_v. For each $a \in C_v$,

1. Create a child node v_a for v and label the edge (v, v_a) by a.
2. Simplify the clauses by setting the variables along the path P_{rv_a} to 1. If there is an empty clause, set $Q_{v_a} = Q_v \cup \{a, \perp\}$ and v_a will not be expanded and thus will become a leaf node. Otherwise, label v_a by $Q_{v_a} = Q_v \cup \{a\}$.
3. If the level of the node is $\tau(F)$, make it a leaf node.
4. Order the child nodes of v left-right according to a tree-edge ordering.

▶ **Proposition 12.** *The tree T with the labeling as described above is a transversal tree for F.*

Proof. Indeed each non-leaf node has at most k children. All leaves v with $\perp \notin Q_v$ must be at the same level $\tau(F)$ since any node v at a level smaller than $\tau(F)$ and does not contain $\perp$ in Q_v can be expanded and since the construction stops at level $\tau(F)$. It is easy to verify that the labeling Q has the requisite properties. For every v with $\perp \notin Q_v$ and every extension Y

of Q_v to a minimum-size transversal, there exists a leaf with label Y in the subtree T_v since there is a child edge (v, v') of v with label a for some $a \in Y \cap C_v \neq \emptyset$ where C_v is the clause used to expand v. Inductively we can construct a path from v' to a leaf which ultimately has Y as the label. ◀

3.2 TreeSearch: An Algorithm for Enumerating the Valid Leaves of T

Our goal is to search the transversal tree to enumerate all minimum-size transversals. However, visiting all leaf nodes may take at least $k^{\tau(F)}$ time. To improve the efficiency of the search, we prune the tree during our search while guaranteeing the enumeration of each minimum-size transversal exactly once.

Let F be satisfiable k-CNF. Let Π be a clause ordering for F. Let T be transversal tree for F constructed using Π and some tree-edge ordering π. We note that for any tree-edge ordering π, the edges of any shoot S_{uv} (where u is any ancestor of v) are situated in one of three ways with respect to the tree path from u to v: 1) to the left of the tree path, 2) to the right of the tree path, or 3) along the tree path. Our key insight is that for any node v we can determine whether the subtree T_v potentially contains any *new* minimum-size transversals by considering the labels of the edges in the shoot S_{rv}.

Our TreeSearch starts with the root node r of T. Assume that we are currently visiting the node v. Let T_v be a subtree of T rooted at v. If v is not a leaf, let C_v be the monotone clause used for expanding the node v (based on the clause order Π) and $a_1, \ldots, a_{k'}$ be the ordering of its variables according to π for some $k' \leq k$. For $1 \leq i \leq k'$, let v_{a_i} be the i-th child node of v. The edge $e_i = (v, v_{a_i})$ is labeled with a_i. The search procedure TreeSearch starting at node v works as follows:

- If v is a leaf, output Q_v if it is a transversal. In any case, return to the parent.
- Otherwise, process the children of v in order. Let $T_i = T_{v_{a_i}}$ be the transversal tree rooted at the child v_{a_i} for $a_i \in C_v$. Prune the subtree T_i if and only if the shoot S_{rv} contains an edge $e' = (u', v')$ where u' is ancestor of v such that $Q_{e_i} = Q_{e'}$, and the edge e' appears to the left of the path P_{rv}. Search the tree T_i if it is not pruned.

▶ **Fact 2.** *For any clause ordering Π and tree-edge ordering π, TreeSearch outputs all minimum-size transversals of F exactly once.*

3.3 Canonical Clause Ordering and Random Tree Edge Ordering

The time complexity of TreeSearch is bounded by the number of leaf nodes it visits. While we know that TreeSearch outputs all minimum-size transversals without redundancy, it is much less clear how to analyze its complexity. We need two ideas to analyze TreeSearch to get a good bound. The first idea is a *canonical* clause ordering Π in which a sequence of maximally disjoint monotone clauses will precede all other clauses. The second idea is a random π, that is, a tree-edge ordering that orders the children of every node in the transversal tree uniformly and independently at random.

4 Analysis of TreeSearch for Monotone k-CNFs

Let $F = (X, \mathcal{C})$ be a monotone k-CNF where every clause has exactly k literals. Let $t = \tau(F)$ be the transversal number of F. We assume that $t \leq \frac{3n}{5}$. Let T be a transversal tree for F where T is constructed using a canonical clause ordering Π. The child edges of each of its

nodes are randomly ordered from left-right independent of other nodes. Let π denote this random tree-edge ordering. Let r be its root and Q its labeling.

In this section, we analyze TreeSearch and prove Theorem 2 for the monotone case. We start with a few ideas required to keep track of the effect of the random ordering on pruning. We then build upon them Section 5 to prove Theorem 2 for general k-CNFs.

4.1 Random Tree Edge Ordering and Pruning

▶ **Definition 13** (Cut event). *We say that an edge $e = (u, v)$ is* cut *if u has an ancestor u' with a child edge $e' = (u', v')$ where $Q_{e'} = Q_e$ and e' appears to the left of the path P_{ru} according to π.*

We use $\phi(e)$ to denote the event that the edge e is *not* cut. We also use $\phi(P_{uv})$ to denote the event no edge along P_{uv} is cut. We use $\phi(u) = \phi(P_{ru})$ to denote the event that none of the edges along the path from the root to the node u are cut.

▶ **Definition 14** (Survival probability of paths and nodes). *The survival probability $\sigma(P_{uv})$ of a path P_{uv} is $\mathbf{P}(\phi(P_{uv}))$. The* survival probability *$\sigma(u)$ of a node u is $\mathbf{P}(\phi(u))$.*

▶ **Definition 15** (Survival value of a transversal tree). *For a subtree T_u rooted at u, we define $\sigma(T_u) = \sum_{v \text{ is leaf of } T_u} \sigma(v)$ as the survival value of T_u. We write $\sigma(T) = \sigma(T_r)$ where r is the root node.*

Our main tool for upper bounding the expected running time of TreeSearch is the following basic fact.

▶ **Fact 3.** *The expected number of leaves visited by TreeSearch is exactly $\sigma(T)$. In particular $\#\Gamma(F) \leq \sigma(T)$.*

Proof. This follows by definition, since TreeSearch only visits surviving leaves of T under a random tree-edge ordering. Furthermore, let Y be a minimum-size transversal of F. We argue that $\sum_{v:Q_v=Y} \mathbf{P}(\pi(P_{rv})) = 1$ which implies $\#\Gamma(F) \leq \sigma(T)$. ◀

Our goal is to upper bound $\sigma(T)$ by bounding the survival probabilities of the leaves at depth t. A path survives if and only none of its edges are cut. For an edge to be cut, it is necessary that some ancestor of the edge has a child edge with the same label as that of the edge. We will keep track of repeated edge labels via markings to bound the cut probabilities from below and thereby bounding the survival probabilities from above.

▶ **Definition 16** (Marking set of an edge). *The* marking set *$M(e)$ of an edge $e = (u, v)$ in T is the set of nodes $w \neq u$ in the path P_{ru} which have a child edge e' such that $Q_e = Q_{e'}$. We say that the nodes in $M(e)$ mark the edge e. We also say that the nodes in $M(e)$ mark the label of e.*

▶ **Definition 17** (Marked edges). *An edge $e = (v, u)$ in T is* marked *if $M(e) \neq \emptyset$.*

Marked edges are precisely those that have a non-zero probability of being cut. In fact, we can calculate the survival probability exactly.

▶ **Fact 4.** *For an edge e in T, $\sigma(e) = 2^{-|M(e)|}$.*

▶ **Definition 18.** *Let u be a node in T. For each node v along the path P_{ru}, let $N_u(v) = \{e \in P_{ru} \mid v \in M(e)\}$. $N_u(v)$ is the set of edges along the path P_{ru} marked by v.*

▶ **Fact 5.** *For any node u in T, the survival probability $\sigma(P_{ru})$ of the path P_{ru} is given by*

$$\sigma(P_{ru}) = \Pi_{v:\text{ a node along } P_{ru}} \frac{1}{|N_u(v)|+1}$$

Proof. P_{ru} survives if and only if for every node $v \in P_{ru}$ and every edge $e \in P_{ru}$ that v marks, the child edge of v with the same label as that of e appears to the right of the path. This happens with probability $\frac{1}{|N_u(v)|+1}$ and since these events are independent, the claim follows. ◀

4.2 An Analysis of TreeSearch for Monotone 3-CNF

Although Fact 5 gives us a fairly complete picture of the survival probabilities of individual paths in T in terms of the edge markings of the path, we need a few ideas to find nontrivial upper bounds on $\sigma(T)$. The first idea is the concept of a weight which captures the number of marked edges in a path or a shoot.

▶ **Definition 19** (Weight). *Let P_{uv} be a path in T. The* weight *of P_{uv} is defined as $W(P_{uv}) \stackrel{def}{=} |\{e \in P : M(e) \neq \emptyset\}|$, i.e., the number of marked edges along P_{uv}. The* weight *of a shoot S_{uv} denoted by $W(S_{uv})$ is the number of marked edges in the shoot S_{uv}.*

The following fact provides a lower bound on the weight of each root to leaf shoot in T.

▶ **Fact 6.** *Every root to leaf shoot S_{ru} in T has a weight of at least $3t - n$.*

Proof. Since the depth of T is t, a root to leaf shoot has $3t$ edges and there are only n distinct edge labels, at least $3t - n$ labels appear at least twice. ◀

▶ **Definition 20** (Weight of a tree). *We say that a tree has weight w if every root to leaf shoot of the tree has weight at least w.*

▶ **Definition 21** ($M(w,d)$). *For non-negative integers w and d, let $M(w,d)$ be the maximum survival value over all ternary depth-d transversal trees with weight w.*

We can now upper bound $\sigma(T)$ in terms of $M(w,d)$ exploiting the canonical clause ordering. Our canonical clause ordering Π starts with a maximal collection of disjoint clauses $C_1, C_2, \cdots, C_m$ so that all clauses in the formula intersect with at least one of the clauses from these disjoint clauses. We observe that $m \geq \frac{t}{3}$ for monotone F since otherwise by setting all the variables in the monotone clauses C_i, we satisfy F contradicting that the transversal number of F is t.

▶ **Lemma 22.**

$$\sigma(T) \leq \begin{cases} 3^t & t \leq \frac{n}{3} \\ 3^{\frac{t}{3}} \times M(3t-n, \frac{2}{3}t) & \text{otherwise} \end{cases}$$

Proof. For $t \leq \frac{n}{3}$, we use the trivial upper bound of 1 on the survival probability of paths to conclude that $\sigma(T) \leq 3^t$ as desired.

For $t \geq \frac{n}{3}$, we use the fact that the canonical clause ordering starts with a maximal collection of disjoint clauses $C_1, C_2, \cdots, C_m$ where $m \geq \frac{t}{3}$. We observe that for $1 \leq i \leq \frac{t}{3} \leq m$ each node at level i of T is expanded by the same clause C_i. Moreover, none of the child edges of nodes at level $1 \leq i \leq\leq \frac{t}{3} \leq m$ are marked as the corresponding clauses are disjoint. The result follows since for every node u at level $\frac{t}{3}+1$, the subtree T_u has depth $\frac{2t}{3}$ and the weight of every root to leaf shoot in T_u is at least $3t-n$ minus the number of marked edges from root to $u = 3t - n - 0 = 3t - n$ (Fact 6). ◀

4.2.1 Upper Bounds on $M(w, d)$

Upper bounding $M(w, d)$ for T based on random tree-edge ordering π is challenging. Instead we introduce a different random process π' for T: Each edge e survives with probability p_e independently where $p_e = \lambda$ if e is marked and 1 otherwise, where we define $\lambda \stackrel{\text{def}}{=} \frac{1}{\sqrt{3}}$. The concept of survival probability under π' can be extended to paths and nodes of T. For example, the $\sigma'(P) = \prod_{e \in P} p_e$ is the survival probability of the path P under π'. Similarly, we define the survival value $\sigma'(T)$ of the transversal tree T according to π' as $\sum_{v \text{ is a leaf}} \sigma'(P_{rv})$. We define $M'(w, d)$ as the maximum survival value $\sigma'(T')$ of transversal trees T' of depth d where every root to leaf shoot has weight at least w. The following lemma shows that $\sigma(T) \leq \sigma'(T)$.

▶ **Lemma 23.** *For a root to leaf path P_{ru}, $\sigma(P) \leq \lambda^{W(P_{ru})} = \sigma'(P)$ which in turn implies $\sigma(T) \leq \sigma'(T)$ and $M(w, d) \leq M'(w, d)$.*

Proof. Given an edge $e \in P_{ru}$, define the contribution y_e of e to the survival probability (according to π) of P_{ru} as

$$y_e \stackrel{\text{def}}{=} \prod_{v \in M(e)} \left(\frac{1}{|N_u(v)| + 1} \right)^{1/|N_u(v)|}$$

where the empty product is considered as 1. By Fact 5, $\sigma(P_{ru}) = \prod_{e: M(e) \neq \emptyset} y_e$. It is then sufficient to show that $y_e \leq p_e$. Observe that $N_u(v)$ can be at most 2 since F is a 3-CNF. For each $v \in M(e)$ with $N_u(v) = 1$, the probability that v does not cut e is exactly $\frac{1}{2}$, independent of other nodes. If $N_u(v) = 2$ for $v \in M(e)$, v marks another edge e' along the path in addition to e. The probability that v cuts neither e nor e' is exactly $\frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$, independent of other nodes. If $N_u(v) = 2$, we regard the probability of each edge surviving as the geometric average λ of the survival probabilities of individual edges. As a consequence, y_e can be written as $(\frac{1}{2})^a (\frac{1}{\sqrt{3}})^b$ for some non-negative integers a and b where a is the number of ancestors v of e such that v marks exactly one edge along the path and b is the number of ancestors v of e such that v marks exactly two edges along the path. We now argue that y_e is at most p_e. If e is not marked, then $a + b = 0$ and hence $y_e = p_e$. If e is marked, we have $a + b > 0$ which implies $y_e \leq \lambda = p_e$. ◀

The following lemma determines $M'(w, d)$.

▶ **Lemma 24.** *For all $0 \leq d \leq n, 0 \leq w \leq 3d$, we have*

$$M'(w, d) = \begin{cases} (2 + \lambda)^w 3^{d-w} & 0 \leq w \leq d \\ (1 + 2\lambda)^{w-d} (2 + \lambda)^{2d-w} & d \leq w \leq 2d \\ (3\lambda)^{w-2d} (1 + 2\lambda)^{3d-w} & 2d \leq w \leq 3d \end{cases}$$

Lemma 24 already gives an upper bound $\sigma(T) \leq M'(3t - n, t)$. However, taking advantage of Lemma 22, we can improve this bound to establish Theorem 2 for monotone formulas.

4.2.2 Proof of Theorem 2 for monotone formulas

Proof. By Fact 3 and Lemma 23 the expected time of TreeSearch is bounded by $\sigma'(T)$ (up to polynomial factors). We divide the proof into cases based on the value of t.

Case 1. $t \leq \frac{n}{3}$. Applying Lemma 22 for the case $t \leq \frac{n}{3}$, we get $\sigma(T) \leq 3^t$.

Case 2. $\frac{n}{3} < t \leq \frac{3n}{7}$. $t \leq \frac{3n}{7}$ implies $3t - n \leq \frac{2t}{3}$. We apply Lemma 22 together with Lemma 24 for the case $0 \leq w \leq d$ to get

$$\sigma(T) \leq 3^{\frac{t}{3}} M'\left(3t - n, \frac{2t}{3}\right) = \left(\frac{3}{2+\lambda}\right)^n \left(\frac{(2+\lambda)^3}{9}\right)^t \leq 1.164^n \times 1.9023^t$$

Case 3. $\frac{3n}{7} \leq t \leq \frac{n}{2}$. We note that $t \leq \frac{n}{2}$ implies $3t - n \leq t \leq \frac{4t}{3}$ and $t \geq \frac{3n}{7}$ implies $3t - n \geq \frac{2t}{3}$. We apply Lemma 22 together with Lemma 24 for the case $d \leq w \leq 2d$ to get

$$\sigma(T) \leq 3^{\frac{t}{3}} M'\left(3t - n, \frac{2t}{3}\right) = \left(\frac{2+\lambda}{1+2\lambda}\right)^n \left(\left(\frac{3(1+2\lambda)^7}{(2+\lambda)^5}\right)^{1/3}\right)^t \leq 1.1962^n \times 1.7851^t$$

◀

4.2.3 Proof of Lemma 24

Let T' be a tree of depth d and weight $0 \leq w \leq 3d$. It is clear that the survival value $\sigma'(T')$ of T' is determined once the edges are marked consistent with the fact that every root to leaf shoot has weight at least w. We say that a transversal tree T' of depth d and weight w is *normal* if it has the following marking: Let $w = id + j$ where $0 \leq i \leq 3$ and $0 \leq j < d$. Mark $i + 1$ children of every non-leaf node in the first j levels and mark i children for each of the remaining non-leaf nodes. The survival value of normal tree is exactly $((i+1)\lambda + 2 - i)^j (i\lambda + 3 - i)^{d-j}$. One can easily see that this is given exactly as $M'(w, d)$ as in the statement of the lemma. We will show that normal trees have the largest survival values by induction on d which completes the proof of Lemma 24.

Let T' be a tree of depth d and weight w with the maximum survival value $\sigma'(T')$. Let r be the root of T'. Assume that l_1 children of r are marked. Let T'_1, T'_2, and T'_3 be the subtrees of T' of depth $d-1$ and weight $w - l_1$ with r_1, r_2, and r_3 as their root nodes respectively. T'_1, T'_2, and T'_3 are normal by induction hypothesis. Assume that l_2 child edges of each of r_i are marked. The survival value of T' is $g_1 g_2 M'(w - (l_1 + l_2), d - 2)$ where $g_1 = (3 - l_1 + l_1\lambda)$ and $g_2 = (3 - l_2 + l_2\lambda)$. It is easy to see that if $l_1 + l_2$ is held constant, $g_1 g_2$ is maximized when l_1 and l_2 are as equal as possible. If $|l_1 - l_2| = 1$, the survival value of T' does not change if r has marked l_2 children and each r_i has l_1 marked children, that is, if the number of markings of the first two levels are exchanged.

If $l_1 = l_2$, then T' is normal. If $|l_1 - l_2| \geq 2$, then T' does not have the largest survival value which is a contradiction. We are left with the case that l_1 and l_2 differ by one. If $l_1 < l_2$, we swap l_1 and l_2 without changing the survival value and normality follows from induction. If $l_1 > l_2$, the tree is already normal. Otherwise, its survival value cannot be the maximum.

5 Analysis of TreeSearch for arbitrary 3-CNFs

In this section, we analyze transversal trees for arbitrary 3-CNFs and prove Theorem 2. We will introduce few more ideas in addition to those introduced in Section 4.

Throughout this section, we fix a 3-CNF $F = (X, \mathcal{C})$ and let T be its canonical transversal tree with root node r. Let the number of maximally disjoint width 3 clauses used to develop F be m. Let $X_D \subset X$ denote set of variables that appeared in this set of m disjoint clauses. We also note that unlike in Section 4, the clauses used to develop T may not have width exactly 3.

5.1 Slight modification to canonical ordering and TreeSearch

We extend the conditions on the canonical ordering Π of clauses and how TreeSearch uses Π. We order clauses in Π so that all maximally disjoint width 3 clauses appear first, followed by all width 3 monotone clauses, followed by all other clauses. For a node u at level below m, we impose that instead of choosing the first unsatisfied monotone clause from Π, u instead chooses an unsatisfied width 3 monotone clause C from Π such that C does not contain any variable $x \in X_D$ that has appeared twice in the shoot S_{ru}. If such a clause does not exist, then C can pick the first unsatisfied monotone clause from Π.

5.2 Fullness and Double marking

We reuse the notion of weight here and observe that all basic facts and basic lemmas from monotone analysis apply here as well. We introduce one more definition related to this:

▶ **Definition 25** (Uniform Weight). *Let $S = S_{uv}$ be a shoot in T of length ℓ. Let a be the number of edges in the shoot S_{uv}. The* uniform weight *of S denoted by $W^+(S) = W(S)+3\ell-a$.*

We can similarly find a lower bound to this quantity for every root to leaf path:

▶ **Fact 7.** *Every root to leaf shoot in T has a uniform weight of at least $3t - n$.*

Proof. Let S be arbitrary root to leaf path with a edges. As there are only n distinct edge labels, at least $a - n$ labels appear at least twice. So, $W(S) \geq a - n$. Since the depth of T is t, we infer that $W^+(S) = W^+(S) + 3t - a \geq 3t - n$ as desired. ◀

We extend the idea of markings and introduce double markings:

▶ **Definition 26** (Double marking). *We say an edge $e \in T$ is doubly marked if $|M(e)| \geq 2$. Let $P = P_{uv}$ be a path from u to v. We write $W_{\geq 2}(P)$ to denote number of edges e in P such that $|M(e)| \geq 2$.*

Recall that when proving Lemma 22, we took advantage of the fact that any monotone 3-CNF G with $\tau(G) = t$ will contain $\frac{t}{3}$ disjoint monotone clauses. However, there is no such guarantee for arbitrary 3-CNFs. Observe that if the number of maximally disjoint monotone clauses is small, then many clauses of width 3 will intersect with it and this will cause many edges to be doubly marked. We formalize this intution and introduce a new parameter called *fullness* that will help keep track of this:

▶ **Definition 27** (Fullness). *Let u be arbitrary node at level $\geq m$ in T. Let u_m be the node at level m along the path P_{ru}. Then,* fullness *of the shoot S_{ru} is defined as*

$$Y(S_{ru}) := |\{x \in X_D \setminus Q_{u_m} : \exists e = (a, b) \in S_{ru}, depth(a) \geq m, Q_e = x\}|,$$

i.e., the number of variables in X_D that are not along the path in the first m levels and appear as labels of some edge in the shoot after level m. For arbitrary nodes u, v where u is ancestor of v and u appears at level $\geq m$, we define $Y(S_{uv}) = Y(S_{rv}) - Y(S_{ru})$.

This parameter is useful because every node after level m will have at least one edge that will either 'add to' fullness or at least one edge that will have marking set of size at least 2. The edges that are 'doubly marked' will contribute very little to the recursion. Moreover, we will see that fullness of any root to leaf shoot is at most $2m$. As m is small, very few nodes will be such that they will not contain any doubly marked edges. This is our key insight and we fomrally prove this now.

▶ **Fact 8.** *Let M denote the set of all monotone clauses of width 3 in F. Then, every clause $C \in M$ must contain at least one variable x such that $x \in X_D$.*

▶ **Lemma 28.** *Let $u \in T$ be a node at level greater than m. Then, at least one of the following must be true:*

1. *u has at most 2 edges going out.*
2. *u has one edge e going out such that $|M(e)| \geq 2$.*
3. *u has one edge e going out such that $Q_e \in X_D$ and $|M(e)| = 1$.*

Proof. Set variables that are part of Q_u to 1 and let F' be the simplified 3-CNF. Say case 1 does not happen. Then, the monotone clause C used to develop edges out of u has width 3 and so, C is also present in F. By Fact 8, u contains an edge e such that $Q_e = x$ and $x \in X_D$. If $M(e) \geq 2$, then case 2 is satisfied and if $M(e) = 1$, then case 3 is satisfied. ◀

5.3 An Analysis of TreeSearch for arbitrary 3-CNF

We extend Lemma 23 for arbitrary 3-CNFs taking into double markings into account.

▶ **Lemma 29.** *Let T be a transversal tree and let $P = P_{ru}$ be a path starting from root r. Then $\sigma(P) \leq (\frac{1}{\sqrt{3}})^{W^+(P)+W_{\geq 2}(P)}$.*

Proof. For a marked edge $e \in P$, we define the contribution of e as

$$q_e := \prod_{v \in M(e)} \left(\frac{1}{|N_u(v)| + 1} \right)^{1/|N_u(v)|}.$$

By Fact 5, $\sigma(P) = \prod_{e:M(e)\neq\emptyset} q_e$. It is then sufficient to show that $q_e \leq \lambda$ for every marked edge e. Note that q_e can be written as $(\frac{1}{2})^a(\frac{1}{\sqrt{3}})^b$, for some non-negative integers a and b such that $a + b \geq |M(e)|$. This quantity is at most $\frac{1}{\sqrt{3}}$ if $|M(e)| = 1$ and is at most $\frac{1}{3}$ if $|M(e)| \geq 2$. Finally, we observe that e contributes 1 to $W(P)$ if $|M(e)| \geq 1$ and contributes 1 to $W_{\geq 2}(P)$ if $M(e) \geq 2$. ◀

▶ **Definition 30** ($NM(w, d, y)$)**.** *For non-negative integers w, d, y, define $NM(w, d, y)$ to be the maximum of sum of survival probabilities of leaves over depth-d transversal trees T for 3-CNFs. Moreover, for every root to leaf shoot S, $W^+(S) \geq w$ and $Y(S) \leq y$.*

5.3.1 Proving Theorem 2

We will show the following as our main lemma:

▶ **Lemma 31.** *Let F be a 3-CNF over n variables with $\tau(F) = t \leq \frac{n}{2}$. Then for any canonical transversal tree T for F, it holds that*

$$\sigma(T) \leq \begin{cases} 3^t & t \leq \frac{n}{3} \\ 3^{\frac{t}{3}} \times M'\left(3t - n, \frac{2t}{3}\right) & \text{otherwise} \end{cases}$$

where $M'(w, d)$ is the same bound we obtained in Section 4.

Using this, Theorem 2 follows from Lemma 31 by using the exact same argument as in the Proof of monotone case of Theorem 1.

Our main lemma will make use of the following bounds on $NM(w, d, y)$:

▶ **Lemma 32.** *For all $0 \leq d \leq n, 0 \leq w \leq 3d, 0 \leq y \leq d$, it holds that:*

$$NM(w,d,y) \leq \begin{cases} (2+\lambda)^y(2+\lambda^2)^{d-y} & 0 \leq w \leq d \\ (2+\lambda)^{y-(w-d)}(1+2\lambda)^{w-d}(2+\lambda^2)^{d-y} & d \leq w \leq d+y \\ (1+2\lambda)^y(2+\lambda^2)^{2d-w}(1+\lambda+\lambda^2)^{w-d-y} & d+y \leq w \leq 2d \\ (1+2\lambda)^{y-(w-2d)}(3\lambda)^{w-2d}(1+\lambda+\lambda^2)^{d-y} & 2d \leq w \leq 2d+y \\ (3\lambda)^y(1+\lambda+\lambda^2)^{3d-w}(2\lambda+\lambda^2)^{w-2d-y} & 2d+y \leq w \leq 3d \end{cases}$$

Moreover, for $y \geq d$:

$$NM(w,d,y) \leq M'(w,d)$$

where $M'(w,d)$ is from Section 4.

Using this, we now prove our main lemma, which yields Theorem 2 as desired.

Proof of Lemma 31 assuming Lemma 32. If $t \leq \frac{n}{3}$, then observe that T has at most 3^t leaves and we trivially bound $\sigma(T) \leq 3^t$.

For $t \geq \frac{n}{3}$, we proceed by considering the maximal set of disjoint monotone width 3 clauses in F used to develop the first m levels of T. For every node $u \in T$ at level m, let the subtree rooted at u be T_u. We can bound $\sigma(T_u) \leq NM(w,d,y)$ where $d = t-m, w = 3t-n, y = 2m$. Hence, $\sigma(T) \leq 3^m NM(w,d,y)$.

If $m \geq \frac{t}{3}$, then $y = 2m \geq t - m = d$. Applying Lemma 32, we infer that

$$\begin{aligned} \sigma(T) &\leq 3^m M'(3t-n, t-m) \\ &= 3^{t/3}\left(3^{m-t/3}M'\left(3t-n, \frac{2t}{3} - \left(m - \frac{t}{3}\right)\right)\right) \\ &\leq 3^{t/3}M'\left(3t-n, \frac{2t}{3}\right) \end{aligned}$$

and we infer the claim.

So, we assume that $m \leq \frac{t}{3}$ and try to find the value of m which will maximize $\sigma(T)$. Notice that in this case, $y \leq d$ and so, we can't directly reduce to the case of $M'(w,d)$. As $t \leq \frac{n}{2}$, it must be that $w = 3t - n \leq t \leq t + m \leq d + y$. This implies $w \leq d + y$. We now take two cases based on value of w and apply Lemma 32 for the case of $y \leq d$.

Case 1. $0 \leq w \leq d$. In this case, we see that:

$$\begin{aligned} \sigma(T) &\leq 3^m NM\left(3t-n, t-m, 2m\right) \\ &\leq 3^m(2+\lambda)^{2m}(2+\lambda^2)^{t-3m} \\ &= (2+\lambda^2)^t\left(\frac{(3)(2+\lambda)^2}{(2+\lambda^2)^3}\right)^m \end{aligned}$$

Here, the fraction has value > 1 and so is maximized when m is maximized, i.e., when $m = \frac{t}{3}$ in which case:

$$\begin{aligned} \sigma(T) &\leq 3^{t/3}(2+\lambda)^{2t/3} \\ &= 3^{t/3}M'\left(3t-n, \frac{2t}{3}\right) \end{aligned}$$

Here the last equality follows by considering the case of $0 \leq w \leq d$ for $M'(w,d)$.

Case 2. $d \leq w \leq d + y$. In this case, we see that:

$$\begin{aligned}\sigma(T) &\leq 3^m NM\,(3t - n, t - m, 2m)\\ &\leq 3^m (2+\lambda)^{n-2t+m}(1+2\lambda)^{2t-n+m}(2+\lambda^2)^{t-3m}\\ &= \left(\frac{2+\lambda}{1+2\lambda}\right)^{n-2t}(2+\lambda^2)^t\left(\frac{(3)(2+\lambda)(1+2\lambda)}{(2+\lambda^2)^3}\right)^m\end{aligned}$$

Here, the rightmost fraction is > 1 and so is maximized when m is maximized, i.e., when $m = \frac{t}{3}$ in which case:

$$\begin{aligned}\sigma(T) &\leq 3^{t/3}(2+\lambda)^{n-5t/3}(1+2\lambda)^{7t/3-n}\\ &= 3^{t/3}M'\left(3t - n, \frac{2t}{3}\right)\end{aligned}$$

Here the last equality follows by considering the case of $d \leq w \leq 2d$ for $M'(w, d)$ (we can do this as $y \leq d$ and hence, $w \leq 2d$).

Thus, in either case, we exactly recover the monotone bound as desired. ◀

5.3.2 Upper bounds on $NM(w, d, y)$

As done in monotone analysis, we let $\lambda \stackrel{\text{def}}{=} \frac{1}{\sqrt{3}}$. Our goal is to prove Lemma 32. We introduce a recurrence relation $L(w, d, y)$ that we argue will upper bound $NM(w, d, y)$.

▶ **Definition 33** ($L(w, d, y)$). *We define $L(w, d, y) : \mathbb{N}^3 \to \mathbb{R}$ recursively as follows:*

$$L(w, 0, y) = \begin{cases} 1 & w \leq 0 \\ 0 & w > 0 \end{cases}$$

For $w \leq 3d, 0 \leq d \leq n$, and $y \geq 1$, define $L(w, d, y)$ as:

$$\begin{aligned}L(w, d, y) = \max\{&(2+\lambda)L(w-1, d-1, y-1),\\ &(1+2\lambda)L(w-2, d-1, y-1),\\ &3\lambda L(w-3, d-1, y-1)\}\end{aligned}$$

For $w \leq 3d, 0 \leq d \leq n$, and $y = 0$, define $L(w, d, 0)$ as:

$$\begin{aligned}L(w, d, 0) = \max\{&(2+\lambda^2)L(w-1, d-1, 0),\\ &(1+\lambda+\lambda^2)L(w-2, d-1, 0),\\ &(2\lambda+\lambda^2)L(w-3, d-1, 0)\}\end{aligned}$$

We claim that $L(w, d, y)$ gives a good bound on $M(w, d, y)$.

▶ **Proposition 1.** *For all $0 \leq d \leq n, 0 \leq w \leq 3d, 0 \leq y$, it holds that: $NM(w, d, y) \leq L(w, d, y)$*

We will show the following bound on $L(w, d, y)$.

▶ **Lemma 34.** *For all* $0 \leq d \leq n, 0 \leq w \leq 3d, 0 \leq y \leq d$*, it holds that:*

$$L(w,d,y) \leq \begin{cases} (2+\lambda)^y(2+\lambda^2)^{d-y} & 0 \leq w \leq d \\ (2+\lambda)^{y-(w-d)}(1+2\lambda)^{w-d}(2+\lambda^2)^{d-y} & d \leq w \leq d+y \\ (1+2\lambda)^y(2+\lambda^2)^{2d-w}(1+\lambda+\lambda^2)^{w-d-y} & d+y \leq w \leq 2d \\ (1+2\lambda)^{y-(w-2d)}(3\lambda)^{w-2d}(1+\lambda+\lambda^2)^{d-y} & 2d \leq w \leq 2d+y \\ (3\lambda)^y(1+\lambda+\lambda^2)^{3d-w}(2\lambda+\lambda^2)^{w-2d-y} & 2d+y \leq w \leq 3d \end{cases}$$

Moreover, for $y \geq d$*:*

$$L(w,d,y) \leq M'(w,d)$$

where $M'(w,d)$ *is from Section* 4*.*

Combining Proposition 1 and Lemma 34, Lemma 32 trivially follows.

5.3.3 Proving $NM(w,d,y) \leq L(w,d,y)$

Using Lemma 28 and Lemma 29, we come up with a recurrence for $NM(w,d,y)$ and show it's bounded by $L(w,d,y)$, proving Proposition 1.

Proof of Proposition 1. Recall that in the canonical ordering, all width 3 monotone clauses appear first and remaining clauses appear later. After exhausting the width 3 monotone clauses, the remaining clauses that we develop in the transversal tree have width at most 2. Towards this, for non-negative integers w, d: let $M_2(w,d)$ be the maximum sum of survival probabilities over all transversal trees for 2-CNFs where every root to leaf path has uniform weight at least w. Recall that uniform weight is defined with respect to 3-CNFs and we continue using that definition. We get various recurrences by considering cases on number of marked edges out of the root node (0 or 1 or 2) and by observing that some cases are dominated by others (such as various cases of width 1 clauses). The remaining recurrences that are not dominated by any other recurrence are the following:

$$\begin{aligned} M_2(w,d) \leq \max\{&(2)M_2(w-1,d-1), \\ &(1+\lambda)M_2(w-2,d-1), \\ &2\lambda M_2(w-3,d-1)\} \end{aligned}$$

By induction, we infer that $M_2(w,d) \leq L(w,d,0)$.

We now develop a recurrence for $NM(w,d,0)$. Recall that in canonical ordering, either we exhaust all width 3 monotone clause and reach $M_2(w,d)$, or we develop width 3 monotone clause. Observe that Lemma 28 guarantees that every node in such a tree must have an edge e coming out of it such that $|M(e)| \geq 2$. Taking cases on the number of marked edges coming out of the root node and whether root node has 3 or at most 2 edges coming out, we get many recurrences. However certain recurrences are dominated by others and the remaining recurrences that are not dominated by any othe recurrence are as follows:

$$\begin{aligned} NM(w,d,0) \leq \max\{&(2+\lambda^2)NM(w-1,d-1,0), \\ &(1+\lambda+\lambda^2)NM(w-2,d-1,0), \\ &(2\lambda+\lambda^2)NM(w-3,d-1,0), \\ &M_2(w,d)\} \end{aligned}$$

By induction, we again infer that $NM(w,d,0) \leq L(w,d,0)$.

We now develop a recurrence for $NM(w,d,y)$. We again take advantage of the fact that in canonical ordering, either we exhaust all width 3 monotone clause and reach $M_2(w,d)$, or we develop width 3 monotone clause. Moreover, amongst width 3 clauses, canonical ordering causes either y to decrease by at least 1 or we exhaust such clauses and all remaining clauses have the property that a node developed using such a clause will have an outgoing edge e such that $|M(e)| \geq 2$.

We get many recurrences for $NM(w,d,y)$ by considering cases on number of marked edges (1 or 2 or 3) out of the root node, number of marked edges that cause Y to decrease (1 or 2 or 3), various combinations of number of double marked edges (1 or 2 or 3), whether the root node has at most 2 edges coming out, and whether the root node has no edges that cause Y to decrease by at least 1. Notice that if the root node has no edges that cause Y to decrease by at least 1, then by clause ordering, no other width 3 clause can cause Y to decrease and hence, we are in case $NM(w,d,0)$. Lastly, we observe that certain recurrences are dominated by others. The remaining recurrences that are not dominated by any other recurrences are the following:

$$\begin{aligned} NM(w,d,y) \leq \max\{&(2+\lambda)NM(w-1,d-1,y-1),\\ &(1+2\lambda)NM(w-2,d-1,y-1),\\ &3\lambda NM(w-3,d-1,y-1),\\ &NM(w,d,0),\\ &M_2(w,d)\} \end{aligned}$$

By induction, utilizing the fact that $L(w,d,y) \geq L(w,d,0)$, we again infer that $NM(w,d,y) \leq L(w,d,y)$ as desired. ◀

5.3.4 Upper bound on $L(w,d,0)$

We first show Lemma 34 for the special case of $y = 0$:

▶ **Lemma 35.** *For all $0 \leq d \leq n, 0 \leq w \leq 3d$, it holds that:*

$$L(w,d,0) \leq \begin{cases} (2+\lambda^2)^d & 0 \leq w \leq d \\ (2+\lambda^2)^{2d-w}(1+\lambda+\lambda^2)^{w-d} & d \leq w \leq 2d \\ (1+\lambda+\lambda^2)^{3d-w}(2\lambda+\lambda^2)^{w-2d} & 2d \leq w \leq 3d \end{cases}$$

Proof. Let $G_1, G_2, G_3, G : \mathbb{N}^2 \to \mathbb{R}$ be defined as:

$$\begin{aligned} G_1(w,d) &= (2+\lambda^2)^d \\ G_2(w,d) &= (2+\lambda^2)^{2d-w}(1+\lambda+\lambda^2)^{w-d} \\ G_3(w,d) &= (1+\lambda+\lambda^2)^{3d-w}(2\lambda+\lambda^2)^{w-2d} \\ G(w,d) &= \min\{G_1(w,d), G_2(w,d), G_3(w,d)\} \end{aligned}$$

For $1 \leq i \leq 3$, define P_i to be the set of pairs (w,d) such that $d \geq 0$ and $w \in [(i-1)d, id]$. We will show the following two propositions:

▶ **Proposition 36.** *For all* $1 \leq i \leq 3$*, and all* $(w, d) \in P_i : G(w, d) = G_i(w, d)$.

▶ **Proposition 37.** *For all* $1 \leq i \leq 3$ *and all* $(w, d) \in P_i : L(w, d, 0) \leq G(w, d)$.

We observe that Proposition 36 and Proposition 37 together imply our claim.

Proof of Proposition 36. The result follows immediately from the following claims:

▷ Claim 38. $G_1(w, d) \leq G_2(w, d)$ if and only if $w \leq d$, with equality when $w = d$.

▷ Claim 39. $G_2(w, d) \leq G_3(w, d)$ if and only if $w \leq 2d$, with equality when $w = 2d$.

Claim 38 holds because:

$$\frac{G_1(w,d)}{G_2(w,d)} = \left(\frac{2+\lambda^2}{1+\lambda+\lambda^2}\right)^{w-d}$$

which is greater than 1 if and only if $w > d$. Claim 39 holds because:

$$\frac{G_2(w,d)}{G_3(w,d)} = \left(\frac{(1+\lambda+\lambda^2)^2}{(2\lambda+\lambda^2)(2+\lambda^2)}\right)^{w-2d}$$

which is greater than 1 if and only if $w > 2d$. ◀

Proof of Proposition 37. We consider cases on value of w and in every case, induct on d and apply Definition 33 to infer the claim.

Case 1. Assume $(w, d) \in P_1$.

$$\begin{aligned}
L(w,d,0) &\leq \max\{(2+\lambda^2)G(w-1,d-1,0),\\
&\quad (1+\lambda+\lambda^2)G(w-2,d-1,0),\\
&\quad (2\lambda+\lambda^2)G(w-3,d-1,0)\}\\
&\leq \max\{(2+\lambda^2)G_1(w-1,d-1),(1+\lambda+\lambda^2)G_1(w-2,d-1),(2\lambda+\lambda^2)G_1(w-3,d-1)\}\\
&= G_1(w,d)\max\{1,(1+\lambda+\lambda^2)/(2+\lambda^2),(2\lambda+\lambda^2)/(2+\lambda^2)\}\\
&= G_1(w,d)\\
&= G(w,d)
\end{aligned}$$

The last equality follows by applying Proposition 36 for the case $(w, d) \in P_1$.

Case 2. Assume $(w, d) \in P_2$.

$$\begin{aligned}
L(w,d,0) &\leq \max\{(2+\lambda^2)G(w-1,d-1,0),\\
&\quad (1+\lambda+\lambda^2)G(w-2,d-1,0),\\
&\quad (2\lambda+\lambda^2)G(w-3,d-1,0)\}\\
&\leq \max\{(2+\lambda^2)G_2(w-1,d-1),(1+\lambda+\lambda^2)G_2(w-2,d-1),(2\lambda+\lambda^2)G_2(w-3,d-1)\}\\
&= G_2(w,d)\max\{1,1,(2\lambda+\lambda^2)(2+\lambda^2)/(1+\lambda+\lambda^2)^2\}\\
&= G_2(w,d)\\
&= G(w,d)
\end{aligned}$$

The last equality follows by applying Proposition 36 for the case $(w, d) \in P_2$.

Case 3. Assume $(w,d) \in P_3$.

$$\begin{aligned} L(w,d,0) &\leq \max\{(2+\lambda^2)G(w-1,d-1,0), \\ &\quad (1+\lambda+\lambda^2)G(w-2,d-1,0), \\ &\quad (2\lambda+\lambda^2)G(w-3,d-1,0)\} \\ &\leq \max\{(2+\lambda^2)G_3(w-1,d-1),(1+\lambda+\lambda^2)G_3(w-2,d-1),(2\lambda+\lambda^2)G_3(w-3,d-1)\} \\ &= G_3(w,d)\max\{(2+\lambda^2)(2\lambda+\lambda^2)/(1+\lambda+\lambda^2)^2,1,1\} \\ &= G_3(w,d) \\ &= G(w,d) \end{aligned}$$

The last equality follows by applying Proposition 36 for the case $(w,d) \in P_3$.

◀

◀

5.3.5 Upper bound on $L(w,d,y)$

We are finally ready to give general bounds on $L(w,d,y)$:

Proof of Lemma 34. Notice that if $y \geq d$, then if we try and unravel the recurrence, no path can lead to the case $y = 0, d > 0$. Hence, y plays no role in restricting the recurrence and $L(w,d,y)$ follows the same recurrence as $M'(w,d)$, yielding the claim.

For $y \leq d$, we proceed by first defining $H_1, H_2, H_3, H_4, H_5, H : \mathbb{N}^3 \to \mathbb{R}$ as follows:

$$\begin{aligned} H_1(w,d,y) &= (2+\lambda)^y(2+\lambda^2)^{d-y} \\ H_2(w,d,y) &= (2+\lambda)^{y-(w-d)}(1+2\lambda)^{w-d}(2+\lambda^2)^{d-y} \\ H_3(w,d,y) &= (1+2\lambda)^y(2+\lambda^2)^{2d-w}(1+\lambda+\lambda^2)^{w-d-y} \\ H_4(w,d,y) &= (1+2\lambda)^{y-(w-2d)}(3\lambda)^{w-2d}(1+\lambda+\lambda^2)^{d-y} \\ H_5(w,d,y) &= (3\lambda)^y(1+\lambda+\lambda^2)^{3d-w}(2\lambda+\lambda^2)^{w-2d-y} \\ H(w,d,y) &= \min\{H_2(w,d,y),H_3(w,d,y),H_4(w,d,y),H_5(w,d,y)\} \end{aligned}$$

For $1 \leq i \leq 5$, define $Q_i \subset \mathbb{N}^3$ as follows:

$$\begin{aligned} Q_1 &= \{(w,d,y) \in \mathbb{N}^3 : 0 \leq w \leq d+y\} \\ Q_2 &= \{(w,d,y) \in \mathbb{N}^3 : d \leq w \leq d+y\} \\ Q_3 &= \{(w,d,y) \in \mathbb{N}^3 : d+y \leq w \leq 2d\} \\ Q_4 &= \{(w,d,y) \in \mathbb{N}^3 : 2d \leq w \leq 2d+y\} \\ Q_5 &= \{(w,d,y) \in \mathbb{N}^3 : 2d+y \leq w \leq 3d\} \end{aligned}$$

We will show the following propositions that together imply our claim:

▶ **Proposition 40.** *For all $1 \leq i \leq 5$, and all $(w,d,y) \in Q_i : H(w,d,y) = H_i(w,d,y)$.*

▶ **Proposition 41.** *For all $1 \leq i \leq 5$ and all $(w,d,y) \in Q_i : L(w,d,y) \leq H(w,d,y)$.*

We will in fact use Proposition 40 in the proof of Proposition 41. Hence, we prove the former first:

Proof of Proposition 40. The result follows immediately from the following claims:

▷ Claim 42. $H_1(w,d,y) \leq H_2(w,d,y)$ if and only if $w \leq d$, with equality when $w = d$.

▷ Claim 43. $H_2(w,d,y) \leq H_3(w,d,y)$ if and only if $w \leq d+y$, with equality when $w = d+y$.

▷ Claim 44. $H_3(w,d,y) \leq H_4(w,d,y)$ if and only if $w \leq 2d$, with equality when $w = 2d$.

▷ Claim 45. $H_4(w,d,y) \leq H_5(w,d,y)$ if and only if $w \leq 2d+y$, with equality when $w = 2d+y$.

Claim 42 holds because:

$$\frac{H_1(w,d,y)}{H_2(w,d,y)} = \left(\frac{2+\lambda}{1+2\lambda}\right)^{w-d}$$

which is greater than 1 if and only if $w > d$. Claim 43 holds because:

$$\frac{H_2(w,d,y)}{H_3(w,d,y)} = \left(\frac{(1+2\lambda)(2+\lambda^2)}{(2+\lambda)(1+\lambda+\lambda^2)}\right)^{w-d-y}$$

which is greater than 1 if and only if $w > d+y$. Claim 44 holds because:

$$\frac{H_3(w,d,y)}{H_4(w,d,y)} = \left(\frac{(1+2\lambda)(1+\lambda+\lambda^2)}{(2+\lambda^2)(3\lambda)}\right)^{w-2d}$$

which is greater than 1 if and only if $w > 2d$. Claim 45 holds because:

$$\frac{H_4(w,d,y)}{H_5(w,d,y)} = \left(\frac{(3\lambda)(1+\lambda+\lambda^2)}{(1+2\lambda)(2\lambda+\lambda^2)}\right)^{w-2d-y}$$

which is greater than 1 if and only if $w > 2d+y$. ◀

We prove our final proposition:

Proof of Proposition 41. We observe that for $y = 0$, our claim follows from Lemma 35. We use this fact in the inductive argument below and only consider cases where $y \geq 1$. We consider cases on value of w and in every case, induct on $d+y$ and apply Definition 33 to infer the claim.

Case 1. Assume $(w,d,y) \in Q_1$ and $y \geq 1$.

$$\begin{aligned}
L(w,d,y) &= \max\{(2+\lambda)H(w-1,d-1,y-1),(1+2\lambda)H(w-2,d-1,y-1),\\
&\quad (3\lambda)H(w-3,d-1,y-1)\}\\
&\leq \max\{(2+\lambda)H_1(w-1,d-1,y-1),(1+2\lambda)H_1(w-2,d-1,y-1),\\
&\quad (3\lambda)H_1(w-3,d-1,y-1)\}\\
&= H_1(w,d,y)\max\left\{1,\frac{1+2\lambda}{2+\lambda},\frac{3\lambda}{2+\lambda}\right\}\\
&= H_1(w,d,y)\\
&= H(w,d,y)
\end{aligned}$$

The last equality follows by applying Proposition 40 for the case $(w,d,y) \in Q_1$.

Case 2. Assume $(w,d,y) \in Q_2$ and $y \geq 1$.

$$\begin{aligned} L(w,d,y) &\leq \max\{(2+\lambda)L(w-1,d-1,y-1),(1+2\lambda)L(w-2,d-1,y-1),\\ &\quad (3\lambda)L(w-3,d-1,y-1)\}\\ &\leq \max\{(2+\lambda)H_2(w-1,d-1,y-1),(1+2\lambda)H_2(w-2,d-1,y-1),\\ &\quad (3\lambda)H_2(w-3,d-1,y-1)\}\\ &= H_2(w,d,y)\max\left\{1,1,\frac{(3\lambda)(2+\lambda)}{(1+2\lambda)^2}\right\}\\ &= H_2(w,d,y)\\ &= H(w,d,y) \end{aligned}$$

The last equality follows by applying Proposition 40 for the case $(w,d,y) \in Q_2$.

Case 3. Assume $(w,d,y) \in Q_3$ and $y \geq 1$.

$$\begin{aligned} L(w,d,y) &\leq \max\{(2+\lambda)L(w-1,d-1,y-1),(1+2\lambda)L(w-2,d-1,y-1),\\ &\quad (3\lambda)L(w-3,d-1,y-1)\}\\ &\leq \max\{(2+\lambda)H_3(w-1,d-1,y-1),(1+2\lambda)H_3(w-2,d-1,y-1),\\ &\quad (3\lambda)H_3(w-3,d-1,y-1)\}\\ &= H_3(w,d,y)\max\left\{\frac{(2+\lambda)(1+\lambda+\lambda^2)}{(1+2\lambda)(2+\lambda^2)},1,\frac{(2+\lambda^2)(3\lambda)}{(1+2\lambda)(1+\lambda+\lambda^2)}\right\}\\ &= H_3(w,d,y)\\ &= H(w,d,y) \end{aligned}$$

The last equality follows by applying Proposition 40 for the case $(w,d,y) \in Q_3$.

Case 4. Assume $(w,d,y) \in Q_4$ and $y \geq 1$.

$$\begin{aligned} L(w,d,y) &\leq \max\{(2+\lambda)L(w-1,d-1,y-1),(1+2\lambda)L(w-2,d-1,y-1),\\ &\quad (3\lambda)L(w-3,d-1,y-1)\}\\ &\leq \max\{(2+\lambda)H_4(w-1,d-1,y-1),(1+2\lambda)H_4(w-2,d-1,y-1),\\ &\quad (3\lambda)H_4(w-3,d-1,y-1)\}\\ &= H_4(w,d,y)\max\left\{\frac{(3\lambda)(2+\lambda)}{(1+2\lambda)^2},1,1\right\}\\ &= H_4(w,d,y)\\ &= H(w,d,y) \end{aligned}$$

The last equality follows by applying Proposition 40 for the case $(w,d,y) \in Q_4$.

Case 5. Assume $(w,d,y) \in Q_5$ and $y \geq 1$.

$$\begin{aligned} L(w,d,y) &\leq \max\{(2+\lambda)L(w-1,d-1,y-1),(1+2\lambda)L(w-2,d-1,y-1),\\ &\quad (3\lambda)L(w-3,d-1,y-1)\}\\ &\leq \max\{(2+\lambda)H_5(w-1,d-1,y-1),(1+2\lambda)H_5(w-2,d-1,y-1),\\ &\quad (3\lambda)H_5(w-3,d-1,y-1)\}\\ &= H_5(w,d,y)\max\left\{\frac{(2+\lambda)(2\lambda+\lambda^2)^2}{(3\lambda)(1+\lambda+\lambda^2)^2},\frac{(1+2\lambda)(2\lambda+\lambda^2)}{(3\lambda)(1+\lambda+\lambda^2)},1\right\}\\ &= H_5(w,d,y)\\ &= H(w,d,y) \end{aligned}$$

The last equality follows by applying Proposition 40 for the case $(w,d,y) \in Q_5$.

◀

◀

6 Satisfiability for CNFs with bounded negations

We now use TreeSearch to give an enumeration algorithm for class of CNFs with arbitrary width and bounded negations in each clause.

We will use the following well known estimate of binomial coefficients:

▶ **Proposition 46.** *Let $H_2 : (0,1) \to (0,1)$ be the binary entropy function defined as $H_2(x) = -x\log_2(x) + (1-x)\log_2(1-x)$. Then, for $k \leq n/2$, it holds that: $\sum_{i=0}^{k}\binom{n}{k} \leq \text{poly}(n)2^{nH_2(k/n)}$.*

Proof of Theorem 4. Without loss of generality we assume each clause F contains at most 3 postive literals. Indeed, if every clause in F contains at most 3 negative literals, then we can negate every literal in every clause and consider the resultant CNF. This CNF is satisfiable if and only if the original CNF was satisfiable. Moreover, the new CNF has the property that each clause contains at most 3 positive literals.

Let $c = 0.71347$. Then, we use TreeSearch to to enumerate all minimal satisfiable assignments of weight at most cn. We then exhaustively go over all assignments α with weight at least cn and check whether α satisfies F and output such minimal α.

The runtime of the exhaustive procedure is

$$\text{poly}(n)\sum_{i=cn}^{n}\binom{n}{k} \leq \text{poly}(n)2^{nH_2(c)} \leq O(1.8204^n)$$

Notice that when we develop the transversal tree, we only develop positive monotone clauses. Any positive monotone clauses that we encounter during the TreeSearch procedure for F must have width at most 3 as each clause contains at most 3 positive literals. Hence, the resultant transversal tree T is still a ternary tree. So, every root to leaf shoot S must have weight at least $3t - n$ where $t = cn$. We do not put any lower bound on Y for any such shoot and so, we set $y = \infty$. Then, the runtime of TreeSearch upto polynomial factors is bounded by $NM(3(cn) - n, cn, \infty) \leq M'((3c-1), cn)$. We observe that $c \leq 3c - 1 \leq 2c$ and so, we are in the regime where $w \leq d \leq 2d$. Thus,

$$\begin{aligned} M'((3c-1)n, cn) &\leq \left(\left(1+\frac{2}{\sqrt{3}}\right)^{2c-1}\left(2+\frac{1}{\sqrt{3}}\right)^{1-c}\right)^n \\ &\leq 1.8204^n \end{aligned}$$

Hence, the runtime of our algorithm is indeed $O(1.8204^n)$ as desired.

◀

7 Conclusion

We gave a new non-trivial algorithm for Enum$(3, \frac{n}{2})$: given an n-variable 3-CNF with no satsifying assignment of Hamming weight less than $\frac{n}{2}$, we can enumerate all satisfying assignments of Hamming weight exactly $\frac{n}{2}$ in expected time 1.598^n. Several fascinating questions with major consequences remain open. Here we list the most pressing.

1. We already mentioned that Enum$(3, \frac{n}{2})$ cannot be solved in less than 1.565^n steps. Close this gap.
2. Can our approach produce significant improvements for k-CNFs with $k > 3$?

It seems that to make progress towards resolving these problems, deeper analysis of the structure of k-CNFs will be required.

References

1 Kazuyuki Amano. Depth-three circuits for inner product and majority functions. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation, ISAAC 2023, December 3-6, 2023, Kyoto, Japan*, volume 283 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: `https://doi.org/10.4230/LIPIcs.ISAAC.2023.7`, `doi:10.4230/LIPICS.ISAAC.2023.7`.

2 Tobias Brüggemann and Walter Kern. An improved deterministic local search algorithm for 3-SAT. *Theor. Comput. Sci.*, 329(1-3):303–313, 2004. `doi:10.1016/j.tcs.2004.08.002`.

3 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Comput. Complex.*, 24(2):333–392, 2015. `doi:10.1007/s00037-015-0100-0`.

4 Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon M. Kleinberg, Christos H. Papadimitriou, Prabhakar Raghavan, and Uwe Schöning. A deterministic $(2 - 2/(k+1))^n$ algorithm for k-SAT based on local search. *Theor. Comput. Sci.*, 289(1):69–83, 2002. `doi:10.1016/S0304-3975(01)00174-8`.

5 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2):8:1–8:23, 2019. `doi:10.1145/3284176`.

6 Peter Frankl, Svyatoslav Gryaznov, and Navid Talebanfard. A variant of the VC-dimension with applications to depth-3 circuits. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 72:1–72:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.ITCS.2022.72`.

7 Alexander Golovnev, Alexander S. Kulikov, and R. Ryan Williams. Circuit depth reductions. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 24:1–24:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. URL: `https://doi.org/10.4230/LIPIcs.ITCS.2021.24`, `doi:10.4230/LIPICS.ITCS.2021.24`.

8 Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k-SAT algorithms using biased-PPSZ. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 578–589. ACM, 2019. `doi:10.1145/3313276.3316359`.

9 Johan Håstad, Stasys Jukna, and Pavel Pudlák. Top-down lower bounds for depth-three circuits. *Comput. Complex.*, 5(2):99–112, 1995. `doi:10.1007/BF01268140`.

10 Timon Hertli. Breaking the PPSZ barrier for unique 3-SAT. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 600–611. Springer, 2014. `doi:10.1007/978-3-662-43948-7\_50`.

11 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972. SIAM, 2012. `doi:10.1137/1.9781611973099.77`.

12 Peter Keevash. Hypergraph Turán problems. In *Surveys in combinatorics 2011*, volume 392 of *London Math. Soc. Lecture Note Ser.*, pages 83–139. Cambridge Univ. Press, Cambridge, 2011.

13 Konstantin Kutzkov and Dominik Scheder. Using CSP to improve deterministic 3-sat. *CoRR*, abs/1007.1166, 2010. URL: `http://arxiv.org/abs/1007.1166`, `arXiv:1007.1166`.

14 Victor Lecomte, Prasanna Ramakrishnan, and Li-Yang Tan. The composition complexity of majority. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 19:1–19:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.CCC.2022.19`.

15 Andrea Lincoln and Adam Yedidia. Faster random k-CNF satisfiability. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata,*

Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference), volume 168 of *LIPIcs*, pages 78:1–78:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.78.

16 Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discret. Appl. Math.*, 10(3):287–295, 1985. doi:10.1016/0166-218X(85)90050-2.

17 Robin A. Moser and Dominik Scheder. A full derandomization of schöning's k-SAT algorithm. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 245–252. ACM, 2011. doi:10.1145/1993636.1993670.

18 Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 163–169. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185365.

19 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.

20 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chic. J. Theor. Comput. Sci.*, 1999, 1999. URL: http://cjtcs.cs.uchicago.edu/articles/1999/11/contents.html.

21 Ramamohan Paturi, Michael E. Saks, and Francis Zane. Exponential lower bounds for depth three boolean circuits. *Comput. Complex.*, 9(1):1–15, 2000. doi:10.1007/PL00001598.

22 Dominik Scheder. PPSZ is better than you think. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 205–216. IEEE, 2021. doi:10.1109/FOCS52979.2021.00028.

23 Dominik Scheder and Navid Talebanfard. Super strong ETH is true for PPSZ with small resolution width. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 3:1–3:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.3.

24 Uwe Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. *Algorithmica*, 32(4):615–623, 2002. doi:10.1007/s00453-001-0094-7.

25 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical foundations of computer science (Proc. Sixth Sympos., Tatranská Lomnica, 1977)*, pages 162–176. Lecture Notes in Comput. Sci., Vol. 53, 1977.

26 Nikhil Vyas and R. Ryan Williams. On super strong ETH. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 406–423. Springer, 2019. doi:10.1007/978-3-030-24258-9_28.

27 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. doi:10.1137/10080703X.