Local Enumeration: The Not-All-Equal Case

Mohit Gurumukhani
Cornell University, Ithaca, NY, USA

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

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

Navid Talebanfard
University of Sheffield, UK

Abstract

Gurumukhani et al. (CCC'24) proposed the *local enumeration* problem $\textsc{Enum}(k,\, t)$ as an approach to break the Super Strong Exponential Time Hypothesis (SSETH): for a natural number k and a parameter t, given an n-variate k-CNF with no satisfying assignment of Hamming weight less than $t(n)$, enumerate all satisfying assignments of Hamming weight exactly $t(n)$. Furthermore, they gave a randomized algorithm for $\textsc{Enum}(k,\, t)$ and employed new ideas to analyze the first non-trivial case, namely $k = 3$. In particular, they solved $\textsc{Enum}(3,\, \frac{n}{2})$ in expected 1.598^n time. A simple construction shows a lower bound of $6^{\frac{n}{4}} \approx 1.565^n$.

In this paper, we show that to break SSETH, it is sufficient to consider a *simpler* local enumeration problem NAE-$\textsc{Enum}(k,\, t)$: for a natural number k and a parameter t, given an n-variate k-CNF with no satisfying assignment of Hamming weight less than $t(n)$, enumerate all Not-All-Equal (NAE) solutions of Hamming weight exactly $t(n)$, i.e., those that satisfy and falsify some literal in every clause. We refine the algorithm of Gurumukhani et al. and show that it *optimally* solves NAE-$\textsc{Enum}(3,\, \frac{n}{2})$, namely, in expected time $\mathsf{poly}(n) \cdot 6^{\frac{n}{4}}$.

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.STACS.2025.42

Related Version *Full Version*: `https://arxiv.org/abs/2501.02886`

Funding *Mohit Gurumukhani*: Supported by NSF CAREER Award 2045576 and a Sloan Research Fellowship.
Ramamohan Paturi: Partially supported by NSF grant 2212136.

Acknowledgements We want to thank Pavel Pudlák for helpful discussions.

1 Introduction

Four decades of research on the exact complexity of k-SAT has given rise to a handful of non-trivial exponential time algorithms, i.e., algorithms running in time $2^{(1-\epsilon)n}$ with non-trivial *savings* $\epsilon > 0$ [10, 12, 15, 2, 11, 8, 6, 13]. Despite extensive effort, PPSZ [11] remains essentially the fastest known k-SAT algorithm. It is also known that its analysis cannot be substantially improved [14]. The *Super Strong Exponential Time Hypothesis (SSETH)* formalizes the lack of progress in improving the exact complexity of k-SAT, and states that it cannot be solved in time $2^{(1-\omega(1/k))n}$ [16]. Gurumukhani et al. [5] recently proposed the *local enumeration* problem as a new approach to refute SSETH. More precisely, $\textsc{Enum}(k,\, t)$ is defined as follows: for natural number k and a parameter t, given an n-variate k-CNF F with no satisfying assignment of Hamming weight less than $t(n)$, enumerate all satisfying assignments of Hamming weight exactly $t(n)$. They provided a randomized branching algorithm and presented novel ideas to analyze it for the first non-trivial case, $k = 3$.

In this paper, we argue that fast algorithms for an easier local enumeration problem would also refute SSETH. To be precise, we consider the NAE-Enum(k, t) problem which is formally defined as follows: for a natural number k and a parameter t, given an n-variate k-CNF F with no satisfying assignment of Hamming weight less than $t(n)$, enumerate all Not-All-Equal (NAE) solutions of Hamming weight exactly $t(n)$ [1]. Recall that a NAE solution to a CNF is one which satisfies and falsifies a literal in every clause. We will show that a refinement of the algorithm of [5] can *optimally* solve NAE-Enum(k, $\frac{n}{2}$) for $k = 3$. For this algorithm $t = \frac{n}{2}$ is the hardest case, and we therefore have an algorithm for all $t \leq \frac{n}{2}$. Our proof utilizes a new technique for analyzing a transversal tree (a tree of satisfying solutions with Hamming weight exactly $t(n)$ constructed using the clauses of the k-CNF).

It is easy to see that k-SAT can be reduced to $(k+1)$-NAE-SAT by the following folklore reduction: Given a k-CNF F, define a $(k+1)$-CNF F' as follows. Let z be a new variable. For each clause $C \in F$ we include the clause $C \vee z$ in F'. It is clear that F is satisfiable iff F' has a NAE solution[2]. Furthermore, if we can solve k-NAE-SAT in time $2^{(1-\mu_k)n}$, then we can solve k-SAT in time $O(2^{(1-\mu_{k-1})n})$. In the other direction, the k-NAE-SAT problem has a trivial reduction to k-SAT: Given a k-CNF F, construct the formula F' by adding, for each clause of F the clause consisting of the negations of its literals, then F has an NAE solution if and only if F' is satisfiable. Therefore, for large k, k-SAT can be solved with asymptotically the same *savings* as that of k-NAE-SAT.

It is noted in [5] that upper bounds on Enum(k, t) for all $t \leq \frac{n}{2}$ imply k-SAT upper bounds. It is easy to see that the same holds for NAE-Enum(k, t) and k-NAE-SAT. Therefore, by the discussion above, as far as k-SAT savings are concerned, we may only focus on NAE-Enum(k, t) for all $t \leq \frac{n}{2}$.

Lower bounds for local enumeration

Define k-CNF $\mathsf{Maj}_{n,k}$ as follows. Divide the n variables into blocks of size $2k-2$, and for each block include all positive clauses of size k. Every satisfying assignment of $\mathsf{Maj}_{n,k}$ sets at least $k-1$ variables in each block to 1, and thus has Hamming weight at least $\frac{n}{2}$. Notably, all minimum-weight satisfying assignments are NAE. Furthermore, the number of minimum-weight satisfying assignments of $\mathsf{Maj}_{n,k}$ is $2^{(1-O(\log(k)/k))n}$. This, in particular, is a lower bound on the complexity of both Enum(k, $\frac{n}{2}$) and NAE-Enum(k, $\frac{n}{2}$), and thus if we can show that any of these bounds is tight for all $t \leq \frac{n}{2}$, we break SSETH. We reiterate that it is wide open to obtain even a combinatorial (non-algorithmic) upper bound of this kind; while this is not good enough to break SSETH, it is necessary and will be very interesting to obtain such a bound.

1.1 Our Contributions

Gurumukhani et al. [5] showed that their algorithm solves Enum(k, t) for all $t \leq \frac{n}{2}$, and in particular, it solves Enum(3, $\frac{n}{2}$) in expected 1.598^n time. Compare this with the lower bound of $6^{\frac{n}{4}} \approx 1.565^n$ which follows from $\mathsf{Maj}_{n,3}$. With respect to this algorithm, the complexity of

[1] we emphasize that we require F to have no satisfying assignment, not only no NAE-satisfying assignment of weight less than $t(n)$

[2] A satisfying assignment of F can be extended to a NAE solution of F' by setting $z = 0$. Conversely, a NAE solution of F' is a satisfying assignment of F projected on the first n variables if $z = 0$, and if $z = 1$, the negation of the remaining variables is a satisfying assignment of F.

Enum(k, t) increases as t grows. Therefore $t = \frac{n}{2}$ is the hardest instance for this algorithm. Here we refine their algorithm, and show that it optimally solves NAE-Enum(3, $\frac{n}{2}$), which is also the hardest instance for this algorithm.

▶ **Theorem 1.** *For $n \geq 0, t \leq n/2$, let F be an arbitrary n-variate 3-CNF where every satisfying assignment has Hamming weight at least t. Then, the number of NAE satisfying assignments of F of Hamming weight exactly t is at most $6^{\frac{n}{4}}$. Furthermore, we can enumerate these solutions in expected poly(n) $\cdot\, 6^{\frac{n}{4}}$ time.*

Theorem 1 will follow from Theorem 21 and Theorem 38. The algorithm of [5] is a randomized variant of the seminal method of Monien and Speckenmeyer [10]: take a positive clause $C = x_1 \vee \ldots \vee x_k$ and for a random ordering π of its variables, recursively solve the problem under the restriction $x_{\pi(1)} = \ldots = x_{\pi(i-1)} = 0, x_{\pi(i)} = 1$, for every $i \in [k]$. While the analysis in [5] is essentially local (only depends on the information along root-leaf paths), we had to develop a global technique to get the optimal bound where we use information from one subtree to analyze the performance in an entirely different subtree. We believe that the elements of our analysis have the potential to generalize for $k > 3$.

Depth-3 complexity of Majority function

Majority is a natural candidate function for beating the state-of-the-art depth-3 circuit lower bounds [7, 9, 1]. The local enumeration paradigm of [5] gives a promising paradigm to establish this, and their result yields new lower bounds for Σ_3^3 circuits, i.e., OR-AND-OR circuits with bottom fan-in at most 3. Our result can also be interpreted as an optimal lower bound Majority for Σ_3^3 circuits in which every Hamming weigh $\frac{n}{2}$ input is a NAE solution of some depth-2 subcircuit.

Hypergraph Turán problems

Following [3, 5, 4], by restricting ourselves to monotone formulas, our result has a natural interpretation for hypergraphs. Let $H = (V, E)$ be a 3-uniform n-vertex hypergraph with no transversal of size less than $\frac{n}{2}$, i.e., any set of vertices that intersects every hyperedge has size at least $\frac{n}{2}$. We show that the number of 2-colorings of H with no monochromatic hyperedge is at most $6^{\frac{n}{4}}$ and give a randomized algorithm that in expected poly(n) $\cdot\, 6^{\frac{n}{4}}$ time enumerates them. This bound is tight, by considering the disjoint union of $\frac{n}{4}$ cliques of size 4 (this hypergraph corresponds to the formula $\mathsf{Maj}_{n,3}$).

Combinatorial interpretation

Our result can also be interpreted in a purely combinatorial manner. For instance, let $S(n, t, k)$ be the maximum number of weight t satisfying assignments of a k-CNF that has no solutions of weight less than t. By construction, we can show $S(n, n/2, 3) \geq 6^{n/4}$. It was conjectured in [5] that $S(n, n/2, 3) \leq 6^{n/4}$. We here prove $S(n, n/2, 3) \leq 6^{n/4}$ under an additional assumption that the 3-CNF formula F is negation closed: If α satisfies F, then negation of α also satisfies F. As pointed out in [5, 4], proving strong upper bounds for $S(n, n/2, k)$ for $k > 3$ is a major open problem and doing so for large k would lead to breakthrough circuit lower bounds for depth 3 circuits. The current best bounds were first obtained by [7] who showed $S(n, n/2, k) \leq 2^{n-O(n/k)}$. We also note that to refute SSETH, one not only needs to show strong upper bounds for $S(n, n/2, k)$ but also needs an enumeration algorithm.

1.2 Proof Strategy

As mentioned earlier, our enumeration algorithm is similar to [5] where we select an unsatisfied monotone clause C, randomly order the variables in C and for $1 \leq i \leq 3$, set the first $i-1$ variables to 0, set variable i to 1 and recurse. This gives rise to a recursion tree (henceforth called transversal tree) where each leaf may correspond to a transversal (it could be a leaf corresponds to a falsified formula). We bound the expected number of leaves that our algorithm visits by carefully ensuring we do not double count leaves that correspond to the same transversal. This is also what [5] did. The key difference here is that in our algorithm, we very carefully choose which clause to develop next: we divide the transversal tree into stages and for each stage, carefully argue certain kinds of clauses must exist and carefully pick those clauses to develop the transversal tree. This is also where we use the not-all-equals assumption to show certain kinds of favorable clauses must exist. With careful accounting, we are able to obtain the tight bound.

2 Transversal trees and the TreeSearch algorithm

In this section we review the TreeSearch algorithm from [5] which enumerates the solutions of a k-CNF solutions of minimum Hamming weight. We also make some additional observations that allow us to get a tight analysis of the algorithm for NAE-3-SAT.

The TreeSearch algorithm is for k-SAT and we want algorithms for NAE-k-SAT. Define the *negation-clause* for clause C to be the clause C' whose literals are the negations of the literals of C. The *negation-closure* of a formula F is the formula $\hat{F}$ obtained from F by adding the negation-clause of every clause of F (if it is not already in F). We say that F is negation-closed if $\hat{F} = F$. It is easy to see that the set of NAE solutions for a formula F is exactly the same as the set of SAT-solutions for its negation-closure $\hat{F}$. So any algorithm that enumerates the minimum-weight satisfying assignments for k-CNF formulas can be used to find the minimum weight NAE-assignments of a k-CNF formula F by applying the algorithm to $\hat{F}$.

2.1 Transversals and Transversal trees

2.1.1 Important definitions

▶ **Definition 2** (Transversals). *A set $S \subseteq X$ is a* transversal of 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.*

We say S is a *minimal transversal of F* if no subset of S is a transversal. We say that S is a *minimum-size transversal of F* if it has the smallest size over all transversals.

The definition of transversal of a formula can be seen as a natural generalization of the standard notion of *transversal of a hypergraph*, which is a set that has nonempty intersection with every edge. Indeed, if F is a monotone formula (where every clause consists of positive literals) then a transversal of F is exactly a transversal of the hypergraph H consisting of the set of clauses of F.

▶ **Definition 3** (Transversal number). *For a satisfiable k-CNF F, the* transversal number *$\tau(F)$ is the cardinality of the minimum-size transversal of F. The set of all minimum-size transversals of F is denoted by $\Gamma(F)$ and the cardinality of $\Gamma(F)$ is denoted by $\#\Gamma(F)$.*

We focus on two related problems: (1) Give an algorithm that enumerates $\Gamma(F)$ and analyze its complexity, and (2) Determine the maximum cardinality of $\Gamma(F)$ among all k-CNF (or NAE-k-CNF) with $\tau(F) = n/2$. Our main technical tool will be *transversal trees*, introduced in [5], and defined below. The definition applies to k-CNF with $\tau(F) = t$ for any k, t.

We need some preliminary definitions. We associate a subset Y of variables to the partial assignment that sets all variables of Y to 1. A clause C is said to be *satisfied by* Y if C contains a positive literal corresponding to a variable from Y.

▶ **Definition 4** (Simplification of clause / formula)**.** *For a clause C that is not satisfied by Y, the* simplification *of C by Y, C/Y is the clause obtained by removing occurrences of negations of variables of Y. Similarly, the* simplification *of F by Y, denoted F/Y, is the formula on $X - Y$ obtained by deleting clauses satisfied by Y, and replacing each remaining clauses C by C/Y.*

Note that the underlying set of variables of both F and F/Y is the same - X. Furthermore, the set of satisfying assignments of F with variables in Y set to 1 is in 1-1 correspondence with the set of satisfying assignments of F/Y. We say that F is *falsified by* Y if F/Y contains an empty clause.

We will be considering rooted trees with root r where each edge e is assigned a label $Q(e) \in X$.

▶ **Definition 5.** *For a node u and descendant node v we have the following definitions:*

- *$P(u, v)$ denotes the path from u to v.*
- *The* shoot *from u to v, $S(u, v)$, consists of the edges of $P(u, v)$ together with all child edges of nodes on the path other than v.*
- *$Q(u, v) = \{Q(e) : e \in P(u, v)\}$. We write $Q(v)$ for $Q(r, v)$.*
- *A clause C is* live *at v provided that C contains no variable in $Q(v)$ (but it may contain the negation of variables in $Q(v)$). For such a clause we write C/v for $C/Q(v)$ and for a formula F, we write F/v for $F/Q(v)$.*

2.1.2 Constructing the tree

We now define a process for growing a tree with node and edge labels $Q(e)$ starting from a root r. The leaves of the current tree are referred to as *frontier nodes*. Each frontier node v is examined and is either designated as a leaf (of the final tree) or a non-leaf as follows:

- If v is a node at depth at most $t - 1$ such that F/v contains no empty clause, then v is a non-leaf. It is labeled by a clause C for which C/v is a positive clause[3]. Node v is expanded to have $|C/v|$ children with the edges labeled by distinct variables in C/v. We denote the label of an edge e by $Q(e)$.
- If F/v contains an empty clause then v is a leaf of the final tree called a *falsified leaf*. For the parent u of such a leaf v, F/u has no empty clause. Since F/v is obtained from F/u by setting $Q(uv)$ to 1, the single literal clause $\neg Q(uv)$ is in F/u. We refer uv as a *falsifying edge*.
- If v is a node at depth t then v is a leaf. If it is not a falsified leaf then v is a *viable leaf*.

A tree constructed according to the above process is a *transversal tree*. The following easy fact (noted in [5]) justifies the name:

[3] F/v must contain a positive clause, otherwise $Q(v)$ is a transversal of size less than t, contradicting $\tau(F) = t$.

▶ **Proposition 6.** *Let F be a formula with $\tau(F) = t$ and let T be a transversal tree for F. For every minimum-size transversal S of F, there is a depth t leaf v of T such that $S = Q(v)$.*

The depth-t leaves can be divided into three types: falsified leaves, viable leaves ℓ for which $Q(\ell)$ is a transversal, and viable leaves ℓ for which $Q(\ell)$ is not a transversal. For a leaf of the third type, $Q(\ell)$ might be a subset of one or more transversals of size larger than t.

▶ Remark 7. Notice that if a leaf u appears before depth t, then u must be a falsified leaf. Unless arising from "effective width 2 clauses", for our analysis sake, we will continue expanding the tree below u. Since u is already falsified, we assume, without loss of generality, that all possible clauses of width at most 3 are live at u (including the empty clause). We then choose a monotone clause and recursively develop the tree underneath u until we reach depth t. Similarly, at all nodes v underneath u, we pretend this is the case, that all clauses are live at v. We do this since our algorithm will dictate for various nodes which kind of clause must be chosen and developed next. We will argue that certain kinds of clauses exist for such nodes. Hence, to simplify analysis, we will assume all clauses needed are already present.

Note that it is possible that many leaves correspond to the same transversal.

2.2 The TreeSearch algorithm for enumerating minimum-size transversals

From Proposition 6, we can enumerate all minimum-sized transversals of F by constructing a transversal tree for F, traversing the tree, and for each leaf ℓ computing $Q(\ell)$ and testing whether $Q(\ell)$ is a transversal. To make this procedure fully algorithmic we need to specify two things.

The first is a *clause selection strategy* which determines, for each node v, the clause of F that labels v. In general, the clause selection strategy may be randomized, but in this paper we consider only deterministic strategies.

The second thing to be specified is the order of traversal of the tree, which is specified a family $\pi = \{\pi(v) : \text{internal nodes } v \text{ of } T\}$ where $\pi(v)$ is a left-to-right ordering of the variables of C/v and thus of the child edges of v. We use the ordering to determine a depth first search traversal of the tree which upon arrival at each leaf ℓ, outputs $Q(\ell)$ if $Q(\ell)$ is a transversal. We will say that *u is to the left of v* if it is visited before v in the traversal. The running time of the algorithm is dominated by the size of the tree, which may be as large as $k^{\tau(F)}$.

To speed up the search, [5] described a simple criterion on edges such that in the pruned tree obtained by removing all edges meeting this criterion and the subtrees below, every minimum-size transversal corresponds to a unique leaf,

To formulate this criterion note that if u is an ancestor of v, the edges of shoot $S(u, v)$ are situated in one of three ways with respect to the path $P(u, v)$: to the left of the tree path, to the right of the tree path, or along the tree path.

▶ **Definition 8** (Superfluous edge). *An edge (uv) is* superfluous *if there is an ancestor w of v that has a child edge wz to the left of $P(r, v)$ such that $Q(wz) = Q(uv)$.*

We define the algorithm TreeSearch on the ordered tree (T, π) to be the depth first search procedure with the following modification: before traversing any edge check whether it is superfluous; if it is then skip that edge (thereby pruning the edge and the subtree below it). This indeed corresponds to setting the corresponding variable to 0.

▶ **Proposition 9** (Implicit in [5]). *For any transversal tree T of F and any ordering of T, the tree obtained by removing all superfluous edges and the subtrees below them satisfies: every minimum-size transversal S of F has a unique leaf ℓ of the pruned tree for which $Q(\ell) = S$, and this leaf is the leftmost leaf of the original tree for which $Q(\ell) = S$. Therefore* TreeSearch *outputs each minimum-size transversals exactly once.*

The time complexity of TreeSearch is bounded (up to $\mathsf{poly}(n)$ factors) by the number of leaf nodes in the pruned tree. The number of leaves depends on the clause selection strategy and the order π. Our goal is to choose the strategy and the order so that we can prove a good upper bound on the number of leaves.

In [5], the algorithm for 3-SAT was analyzed under a specific clause selection rule by considering a randomly chosen ordering π, in which the children of each node in the transversal tree are ordered independently and uniformly at random. This enabled them to get an upper bound that is non-trivial, though probably not optimal.

In this paper, we use a similar approach to analyze NAE-3-SAT. As in [5] we choose π at random. We combine this with a carefully chosen clause selection strategy, described in Section 3, which leads to an optimal upper bound for enumeration of minimum-sized transversals.

2.3 Pruning under random edge ordering

Let $F = (X, \mathcal{C})$ be a k-CNF. Let $t = \tau(F)$ be the transversal number of F. Let T be a transversal tree for F. As indicated above we consider a random ordering π in which the child edges of each node are randomly ordered from left-right independent of other nodes. Under this distribution, whether or not a particular edge is superfluous is a random event.

▶ **Definition 10** (Survival and survival probability).

- *An edge uv is said to* survive *provided that v is not a falsified leaf v and uv is not superfluous. We say that* path $P(u, v)$ survives *if every edge of the path survives, and* node v survives *if path $P(r, v)$ survives.*
- *The* conditional survival probability *of an edge uv, $\sigma(uv)$ is defined to be $\mathbf{P}[v \text{ survives } | u \text{ survives }]$. More generally the* conditional survival probability $\sigma(u, v)$ of path $P(u, v)$ *is defined to be $\mathbf{P}[P(u, v) \text{ survives } | u \text{ survives }]$. It follows that $\sigma(u, v) = \prod_{e \in P(u,v)} \sigma(e)$.*
- *For a node u, let $L(u)$ be the number of leaves of the subtree $T(u)$ that survive, and define the* survival value of u, *$\psi(u)$, to be the conditional expectation $\mathbb{E}[L(u) | u \text{ survives }]$. It follows from linearity of expectation that $\psi(u) = \sum_{v \text{ is leaf of } T(u)} \sigma(u, v)$.*

▶ **Proposition 11.** *Let F be a 3-CNF with n variables and $\tau(F) = n/2$. Fix a clause selection rule for building transversal trees. Then the expected running time of* TreeSearch *on a randomly selected order is $\psi(r)\mathsf{poly}(n)$ and the number of transversals of F satisfies $\#\Gamma(F) \leq \psi(r)$.*

Proof. For the first statement, the running time of TreeSearch for a given π is at most $L(r)\text{poly}(n)$ and so the expected running time on a random order is at most $\psi(r)\text{poly}(n)$. For the second statement, for any order π, Proposition 9 implies that $\#\Gamma(F) \leq L(r)$, and so $\#\Gamma(F) \leq \psi(r)$. ◀

In the sections that follow, we will describe a clause specification strategy and analyze $\psi(r)$ for the case of negation-closed 3-CNF formula (which as noted earlier, corresponds to the case of NAE-3-SAT). For the analysis we will need some additional observations.

First, we determine a condition under which we can exactly determine $\sigma(uv)$ for an edge uv. It follows from the definition of survival probability if uv is a falsifying edge then $\sigma(uv) = 0$. So we consider the case that uv is not falsifying. None of the edges on the path $P(r, u)$ are falsified (since the child node of a falsifying edge is always a leaf).

If uv is not falsifying then it survives if and only if is not superfluous. The condition for being superfluous depends on the label of uv appearing as the label of a child edge of an ancestor of u. To account for this we use a system of marking of edges and vertices:

▶ **Definition 12** (Marking of edges and vertices).
- *The* marking set *$M(e)$ of a tree edge $e = (u, v)$ is the set of nodes $w \neq u$ in the path $P(r, u)$ which have a child edge e' such that $Q(e) = Q(e')$. For $w \in M(e)$ we say that w* marks the edge *e and w* marks the label *$Q(e)$.*
- *An edge e is* marked *provided that $M(e) \neq \emptyset$.*
- *A node u is i-*marked *(for $i \in \{0, 1, 2, 3\}$) if exactly i child edges of u are marked.*

Marked edges are edges that have a non-zero probability of being superfluous. The analysis in [5] uses this to obtain upper bounds on the survival probability of nodes. We now show that under favorable conditions we can calculate the survival probability of nodes exactly.

The event that edge uv survives is exactly the event that for every node $w \in M(uv)$, the child edge of w with the same label as uv is to the right of the path edge that comes from w, which happens with probability 1/2. Thus whether uv survives is determined by the orderings $\pi(w)$ of the child edges of w for all $w \in M(uv)$, and $\mathbf{P}[uv \text{ survives }] = 2^{-|M(e)|}$.

The above computation gives the *unconditioned* probability that uv survives, but $\sigma(uv) = \mathbf{P}[uv \text{ survives } | u \text{ survives }]$ is a conditional probabilty. We now identify a condition under which the conditioning event is independent of the event that uv survives. Say that an edge uv is *disjointly marked* if its marking set $M(uv)$ is disjoint from $M(e)$ for every edge $e \in P(r, u)$. For a disjointly marked edge, the event that uv survives is independent of the event that u survives, i.e., that all edges on $P(r, u)$ survive, because the latter event is determined by the order $\pi(w)$ for nodes that mark some edge of $P(r, u)$ and since uv is disjointly marked, this set of nodes is disjoint from $M(uv)$. Since the order of child edges of nodes are chosen independently, the event that uv survives is independent of the event that all edges on $P(r, u)$ survive. From this we conclude:

▶ **Proposition 13.** *Let T be a transversal tree for the k-CNF F.*
1. *Let uv be an edge that is not a falsifying edge and is disjointly marked.Then $\sigma(uv) = 2^{-|M(e)|}$.*
2. *If v is a leaf that is not falsifying and each edge on $P(r, v)$ is disjointly marked then $\sigma(v) = 2^{-\sum_{e \in P(r,v)} |M(e)|}$.*

For the case of negation-closed 3-CNF (which corresponds to NAE 3-SAT) we note the following:

▶ **Proposition 14.** *If F is a negation-closed 3-CNF formula then in any transversal tree T, every edge that is not a falsifying edge is disjointly marked. Therefore Proposition 13 applies to every edge and every leaf of T.*

Proof. Let uv be an edge with ancestor edge wz and y is a node that marks both uv and wz. We claim that uv is a falsifying edge. y is necessarily an ancestor of w. Let y' be the child of y on the path to w. Then the clause labeling y is $Q(yy') \vee Q(wz) \vee Q(uv)$. Since F is negation closed, F also contains the clause $(\neg Q(yy')) \vee (\neg Q(wz)) \vee (\neg Q(uv))$. This clause is falsified at the node v since $Q(yy'), Q(wz), Q(uv) \in Q(v)$, and therefore v is a falsifying leaf. ◀

We need a few additional definitions.

▶ **Definition 15** (Mass of a node). *For a non-leaf node u in a transversal tree, the* mass of u *is the conditional expectation of the number of surviving children of u given that u survives. By linearity of expectation this is the same as $\sum_{v\text{: child node of } u} \sigma(u, v)$. We also refer to this as the* mass of the clause at u.

▶ **Definition 16** (Defect). *For node u, with e child edges, let* defect *of u to be $3 - e$. We similarly define* defect *of path / shoot as the sum of the defects of all nodes on the path / shoot.*

▶ **Definition 17** (Weight). *Let $S(u, v)$ be a shoot in T. The* weight *of $S(u, v)$ is defined as $W(u, v) \overset{def}{=} |\{e \in S : M(e) \neq \emptyset\}| +$ defect of $S(u, v)$, i.e., the number of marked edges along $S(u, v)$ plus the defect of $S(u, v)$.*

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

▶ **Fact 18.** Every root to leaf shoot $S(r, u)$ in T has a weight of at least $3t - n$.

Proof. Let f be the defect of $S(r, u)$. Since the depth of T is t, a root to leaf shoot has $3t - f$ edges. Partition the edges of the shoot into classes according to the label of the edge, and note that all edges in a class are at different depths. For each class, the edge of minimum depth is unmarked and all other edges in the class are marked, so the number of unmarked edges is at most n and the number of marked edges is at least $3t - f - n$. Hence, the weight of $S(r, u)$ is at least $(3t - f - n) + f = 3t - n$. ◀

3 Clause Selection Criterion

We here provide a clause selection criterion for transversal trees for 3-CNFs that will provide us with a tight analysis for the number of NAE-solutions and for the complexity of enumerating them. The clause selection is divided into three stages. Each node will be associated with a stage and a corresponding clause will be selected from that stage. The three stages are:

1. *Disjoint stage*
2. *Controlled stage*
3. *Arbitrary stage*

We begin the disjoint stage by selecting a pseudomaximum collection $\mathcal{C}^0$ of disjoint monotone clauses of width 3 from F(See Remark 20 for what we mean by pseudomaximum collection; on first reading, the readers should pretend as if we selected maximum such collection). Let $t_0 = |\mathcal{C}^0|$. Our further clause selection criterion depends on the value of t_0:

- If $t_0 \geq \frac{n}{4}$, then we skip the controlled stage and directly enter the arbitrary stage where we select arbitrary monotone clauses for the rest of the TreeSearch process.
- If $t_0 \leq \frac{n}{4}$, then we enter the controlled stage where we carefully control which clauses to select, argue about existence of certain clauses and our analysis leverages this control. This controlled stage also helps us get a handle on the kinds of clauses we can encounter in the arbitrary stage.

We here describe the disjoint stage further. The arbitrary stage needs no further explanation. We analyze the $t_0 \geq \frac{n}{4}$ case in Section 4. We will explain the controlled stage further in Section 5 and will analyze the $t_0 \leq \frac{n}{4}$ case depending upon it in Section 6.

3.1 The disjoint stage

Let the clauses in $\mathcal{C}^0$ be arbitrarily ordered as $C^0_0, C^0_1, \ldots, C^0_{t_0-1}$. We will develop the transversal tree so that for all $0 \leq i \leq t_0 - 1$, all nodes at level i will be expanded using clause C^0_i. We can do this because the disjointness of the clauses ensures that C^0_i is a live clause at every node at level i. We record the useful observation that any clause used to develop a node appearing at level $\ell \geq t_0$ will have at least one marking:

▶ **Fact 19.** Let u be a node at level $\ell \geq t_0$ and let C be a width 3 clause used to expand at u. Then, u will not be 0-marked, i.e., u will be j-marked for $1 \leq j \leq 3$.

Proof. For u to be 0-marked, C must be a width 3 monotone clause and C must be disjoint from all clauses from $\mathcal{C}^0$. However, that contradicts the fact that $\mathcal{C}^0$ is a maximal collection of disjoint width 3 monotone clauses. ◀

▶ Remark 20 (Algorithmic aspect of maximum matching - pseudomaximum matching). Finding a collection $\mathcal{C}^0$ of maximum size is NP-hard, and may require large exponential overhead. To mitigate this, we implement an auxiliary data structure that maintains a pseudomaximum collection of disjoint clauses. Initially, the data structure will greedily pick a maximal such collection of clauses. Later in the algorithm, we might encounter scenarios where we can find a larger collection of disjoint clauses. In these cases, we will "reset" the algorithm with that larger collection of disjoint clauses. Such a reset can take place at most n times and hence, the overhead is $\mathsf{poly}(n)$. In our analysis, we will make claims that take this for granted and allow them to use the fact that $\mathcal{C}^0$ has maximum size ; for any claim where we use this, if the claim does not hold because $\mathcal{C}^0$ is not necessarily of maximum size, the proof of the claim actually will supply us with clauses that will help us form a larger disjoint collection of clauses and we will "reset" our algorithm.

4 Bounding ψ when $t_0 \geq n/4$

In this section, we will show that if in a transversal tree, the number of maximally disjoint clauses chosen in the disjoint stage is at least $n/4$, then $\psi(r) \leq 6^{n/4}$ where r is the root of the tree. Formally:

▶ **Theorem 21.** *Let T be a transversal tree developed using the clause selection criterion as laid out in Section 3 with $t_0(T) \geq \frac{n}{4}$. Then, $\psi(r) \leq 6^{\frac{n}{4}}$ where r is the root of T.*

We will use the following lemma to prove the theorem:

▶ **Lemma 22.** *Let $0 \leq d \leq \frac{n}{2} - t_0, w \geq 0$ and let u be a node at level $\frac{n}{2} - d$ such that every u to leaf shoot in $T(u)$ has weight at least w. Then, $\psi(u) \leq F(w,d)$ where $F(w,d)$ is defined as follows:*

$$F(w,d) = \begin{cases} (\frac{5}{2})^{2d-w} 2^{w-d} & \text{if } w \leq 2d \\ 2^{3d-w}(\frac{3}{2})^{w-2d} & \text{if } 2d \leq w. \end{cases}$$

Assuming this lemma, our main theorem easily follows:

Proof of Theorem 21. Let $t_0 = \frac{n}{4} + \Delta$ for some $\Delta \geq 0$. Let u be an arbitrary node at depth t_0. Let $T(u)$ be the sub-tree rooted at u. The tree $T(u)$ has remaining depth $\frac{n}{4} - \Delta$, and minimum weight of every root to leaf shoot is at least $\frac{n}{2}$ since all edges in the disjoint stage are unmarked. Thus, $T(u)$ satisfies the requirements of Lemma 22, and we infer that $\psi(u) \leq F(n/2, n/4 - \Delta)$. Summing over all $3^{n/4+\Delta}$ nodes at the end of the disjoint stage, we infer that

$$\begin{aligned}\psi(r) &\leq 3^{n/4+\Delta} F(n/2, n/4 - \Delta) \\ &= 3^{n/4+\Delta} 2^{n/4-3\Delta} \left(\frac{3}{2}\right)^{2\Delta} \\ &= 6^{n/4} \left(\frac{27}{32}\right)^{\Delta} \\ &\leq 6^{n/4}\end{aligned}$$

where the last inequality follows because $\Delta \geq 0$. ◀

5 Clause Selection Criterion: the Controlled Stage

Recall that this stage appears after the disjoint stage and applies to nodes u at level t_0 or larger provided $t_0 \leq \frac{n}{4}$. To analyze this stage we will need to construct the transversal tree more carefully, using an involved clause selection criterion which is described in this section.

We use the structural properties of the tree to obtain our lower bound in Section 6.

In the controlled stage, we will carefully select clauses in a way that imposes useful restrictions on the structure of the tree. After that, we enter the arbitrary stage during which we expand the tree using monotone clauses in an arbitrary order. We will show that the structure of the clauses in the controlled stage imposes useful restrictions on the structure of the clauses that arises in the arbitrary stage which will help us bound the survival probability. Nodes with same number of markings will appear consecutively in the controlled stage.

We will need the following definition:

▶ **Definition 23.** *A node u with clause C of width w has* effective width *w' if it has $w - w'$ falsifying edges (as defined in Section 2.1.2) arising out of it.*

Recall from Section 2.1.2 that a falsifying edge leads to a falsifying leaf, i.e. a node v for which F/v contains an empty clause. Therefore the effective width of a node corresponds to the number of children whose subtrees need to be explored.

Our controlled stage is divided into two substages:

κ_1. 1-marked nodes.

κ_2. 2-marked nodes corresponding to clauses of effective width 2.

▶ Remark 24. In stage κ_2, an effective width 2 (monotone) clause C corresponds to a width 3 (monotone) clause where one of the variables, say x, in C will cause a falsifying edge. We will show that there exists a monotone clause C' s.t. $x \in C'$ and remaining two literals in C' are set to 1. Since F is negation-closed, the negation of C' will have both its literals set to 0, simplifying it to the clause $\neg x$. So, any node expanded using C will have the edge labelled with x as a falsifying edge.

We first introduce useful notation regarding the disjoint stage. We then provide a detailed construction of each of the substages in the controlled stage followed by the impact of the controlled stage on the arbitrary stage.

5.1 Additional notation for the disjoint stage

As described in Section 3.1, we select a pseudomaximum size disjoint collection of clauses $C^0_0, C^0_1, \ldots, C^0_{t_0-1}$ in this stage. Our analysis in Section 6 will focus on obtaining an upper bound on $\psi(u_0)$ where u_0 is an arbitrary node at level t_0. We will fix such a node u_0 for the rest of the section. Write $C^0_i = \{p_i, x_i, x'_i\}$ where the variable p_i is the label of the edge along the path $P(r, u_0)$. Let $V_0 = \{0, 1, \ldots, t_0 - 1\}$. For $i \in [t_0]$, let $X_i = \{x_i, x'_i\}$ and let $X = \cup_{i\in[t_0]} X_i$. We will also need the following useful fact:

▶ **Fact 25.** Let u be a node at level $\ell \geq t_0$ and let C be a width 3 monotone clause used to expand at u. Then, there must exist $i \in V_0$ such that $X_i \cap C \neq \emptyset$.

Proof. Indeed, assume such C existed. Then, C must be disjoint from all clauses in $\mathcal{C}^0$. However, this contradicts the fact $\mathcal{C}^0$ is a pseudomaximum collection of clauses. ◀

5.2 Stage κ_1: 1-marked nodes

Let F_1 be the set of width 3 monotone clauses that are live at u_0 and have exactly one marked variable at u_0. We select a pseudomaximum size disjoint collection $\mathcal{C}^1$ of clauses from F_1 and expand u_0 to a sub-tree of uniform depth $t_1 = |\mathcal{C}^1|$.

▶ Remark 26. Similar to Remark 20, in the algorithmic implementation we will maintain an auxiliary data structure that will maintain the pseudomaximum collection $\mathcal{C}^1$. Initially, the collection will be a maximal set of disjoint clauses from F_1 and whenever we come across a claim that uses maximum size property of $\mathcal{C}^1$ and is violated, we will reset $\mathcal{C}^1$ to the disjoint clauses supplied by the proof and "reset" this phase of the algorithm. Again, since a reset can happen at most n times per u_0, this will increase the total runtime of the algorithm by factor of n.

Clauses in F_1 satisfy the following property:

▶ **Fact 27.** If $C = \{x_i, a, b\}$ and $C' = \{x'_i, c, d\}$ are in F_1 where $i \in V_0$, then $\{a, b\} \cap \{c, d\} \neq \emptyset$.

Proof. If C and C' are disjoint, we can replace the clause C^0_i in $\mathcal{C}^0$ with C and C' to obtain a larger size collection violating the pseudomaximum size condition on $\mathcal{C}^0$. ◀

Let $V_1 = \{i \mid \text{a variable from } X_i \text{ appears in a clause from } \mathcal{C}^1\}$. Note that $|V_1| = |\mathcal{C}^1| = t_1$. By Fact 25, for every $C \in F_1$, there is a unique $i \in V_0$ such that C contains exactly one of x_i or x'_i with a single marking, and the remaining two variables in C do not appear in any clause from $\mathcal{C}^0$. Furthermore, by Fact 27, for each $i \in V_0$, at most one variable from X_i can appear in some clause of $\mathcal{C}^1$; that is, if x_i (x'_i) appears in some clause of $\mathcal{C}^1$, then x'_i (x_i) does not appear in any other clause of $\mathcal{C}^1$. These two indeed imply that $|V_1| = |\mathcal{C}^1| = t_1$.

We index the clauses in $\mathcal{C}^1$ using V_1. So, for $i \in V_1$, we write clause C^1_i as $\{\tilde{x}_i, y_i, y'_i\}$ where $\tilde{x}_i \in X_i$. For $i \in V_1$, let $Y_i = \{y_i, y'_i\}$. Let $V_B = V_0 - V_1$ and $m_B = |V_B| = t_0 - t_1$.

Before we describe stage κ_2, we make some careful observations to prepare for it.

5.3 Preparation for Stage κ_2

For a node u at the end of the stage κ_1, define $F_2(u)$ to be the set of monotone width 3 clauses C that are live at u and have exactly two singly marked variables (so that C has mass 2).

▶ **Lemma 28.** *Let u be an arbitrary node at the end of stage κ_1. Let $C \in F_2(u)$ be arbitrary. Then, C must be of one of the following forms:*

1. *$C = \{\tilde{x}_i, \tilde{y}_i, z\}$ for some $i \in V_1, \tilde{x}_i \in X_i \setminus C_i^0, \tilde{y}_i \in Y_i$ and where z does not appear in the shoot $S(r, u)$.*
2. *$C = \{\tilde{x}_i, \tilde{x}_j, z\}$ for some $i \in V_1, j \in V_B, \tilde{x}_i \in X_i \setminus C_i^0, \tilde{x}_j \in X_j$ and z does not appear in the shoot $S(r, u)$.*
3. *$\{\tilde{x}_i, \tilde{x}_j, z\}$ where $i, j \in V_B, \tilde{x}_i \in X_i, \tilde{x}_j \in X_j$ and z does not appear in the shoot $S(r, u)$.*
4. *$\{\tilde{x}_i, \tilde{y}_j, z\}$ where $i \in V_B, j \in V_1, \tilde{x}_i \in X_i, \tilde{y}_j \in Y_j$, and z does not appear in the shoot $S(r, u)$.*

Proof. Since C has mass 2, there must be $z \in C$ that does not appear in the shoot $S(r, u)$. By Fact 25, there exists $i \in V_0$ such that $X_i \cap C \neq \emptyset$. We take cases on whether $i \in V_1$ or not and whether there exist two variables in C from X.

Case 1. $i \in V_1$. Since $C \in F_2(u)$ and $\tilde{x}_i$ is singly marked, $\tilde{x}_i \notin C_i^1 \cap X_i$.

Assume that the second marked variable in C is from X. By Fact 27, the second marked variable must be from X_j where $j \in V_0$ and $j \neq i$. We claim that $j \in V_B$. Assume for contradiction that $j \in V_1$. Since $\tilde{x}_j$ is singly marked, $\tilde{x}_j \notin C_j^1$. Then C_i^1, C_j^1 and C are disjoint clauses that we can use to replace clauses C_i^0, C_j^0 from $\mathcal{C}^0$ and obtain a larger set of disjoint clauses, contradicting the fact that $\mathcal{C}^0$ is a pseudomaximum collection of clauses. Hence, the claim follows.

If the second marked variable in C is not from X, then it must be from Y. Let $\tilde{y}_j \in C$ where $j \in V_1, \tilde{y}_j \in Y_j$. We claim that $i = j$ in this case. Indeed, if not then the clauses C_i^1 and C will be disjoint clauses containing distinct variables from X_i, violating Fact 27. Hence, the claim follows.

Case 2. $i \notin V_1$. As $V_B = V_0 \setminus V_1$, we infer that $i \in V_B$. In this case, if the second marked variable in C is from X, it must be from X_j where $j \in V_B$ (if $j \in V_1$, then we are in the previous case) and the claim follows. Otherwise, the second marked variable in C is from Y and the claim follows as well. ◀

Let u^* denote the unique node at the end of stage κ_1 where the path $P(u_0, u^*)$ consists of only marked edges, each of which is labeled by a variable from X_i for $i \in V_1$. We note a useful relationship between the live clauses at an arbitrary node u at the end of κ_1 and the live clauses at u^*:

▶ **Fact 29.** For every node u at the end of stage κ_1, $F_2(u) \subseteq F_2(u^*)$.

Proof. Let $C \in F_2(u)$ be arbitrary. If C is live at u^*, then C must have exactly 2 markings since $S(r, u) = S(r, u^*)$ and the markings only depend on the shoot. Hence, we show that all such C are live at u^*. Since C is live at u, it is also live at u_0. Hence, if C is not live at u^*, then it must be that C contains one of the variables from $P(u_0, u^*)$. Equivalently, C must have to contain $\tilde{x}_i$ where $i \in V_1$ and $\tilde{x}_i \in C_i^0$. This cannot happen since by Lemma 28, it must be that $\tilde{x}_i \in X_i \setminus C_i^0$. ◀

For simplicity we write $F_2 := F_2(u^*)$. Let $F_{2R} = \{C \in F_2 : \tilde{x}_i \in C \text{ where } i \in V_1\}$. Let $F_{2B} = F_2 \setminus F_{2R}$.

▶ **Fact 30.** Any maximally disjoint set of clauses from F_{2B} has size at most m_B.

Proof. By choice of F_{2B}, it only contains clauses of type 3 or type 4 as laid out in Lemma 28. Assume that there exists a collection S of more than m_B disjoint clauses from F_{2B}. Let $T = \{C_i^0 : i \in V_1\}$. We see that clauses in S are disjoint from T, and S and T have no clauses in common. However then, $S \cup T$ is a disjoint collection of clauses of size

$$|S| + |T| \geq (m_B + 1) + |V_1| = |V_0| + 1 = t_0 + 1$$

violating the fact that $\mathcal{C}^0$ is a pseudomaximum collection of clauses. ◀

▶ Remark 31. We use this fact later in Lemma 37. So even though this fact does not algorithmically provide us with clauses to replace $\mathcal{C}^0$ with, for algorithm's sake, we only care about the maximal collection we encounter from Lemma 37 and there indeed, we can constructively find such a collection if pseudomaximum property is violated.

▶ **Fact 32.** Let $C \in F_{2R}$ be arbitrary. Let $i \in V_1$ be such that $\tilde{x}_i \in C$. Let u be arbitrary node at the end of stage κ_1 where a variable from X_i appears along the path $P(u_0, u)$. Then C is live at u.

Proof. By Lemma 28, C can only take one of two forms: If C is of the form $(\tilde{x}_i, \tilde{x}_j, z)$ where $j \in V_B$ and z does not appear along the shoot $S(u_0, u)$, then this follows. Otherwise, C must be of the form $(\tilde{x}_i, \tilde{y}_i, z)$ where z is not along the shoot $S(u_0, u)$ and $\tilde{y}_i \in Y_i$. By assumption, a variable from X_i is in the path $P(u_0, u)$. This can only happen at the node corresponding to the clause C_i^1. As $\tilde{y}_i$ appears exactly once along the shoot $S(u_0, u)$ - at the node corresponding to the clause C_i^1 - we infer that $\tilde{y}_i$ is not along the path $P(u_0, u)$. Hence, the clause C is still live at u. ◀

Let $V_R = \{i \in V_1 \mid \exists C \in F_{2R}$ such that $X_i \cap C \neq \emptyset\}$. Let $m_R = |V_R|$. Let $m_I = |V_1 \setminus V_R|$. We have $m_B + m_R + m_I = t_0 = \frac{n}{4} - \Delta$. Fix a pseudomaximum size collection $\mathcal{C}'_R$ of disjoint clauses from F_{2R}. Let $V'_R = \{i \in V_R \mid x'_i$ appears in a clause in $\mathcal{C}'_R\}$ and $m'_R = |V'_R| = |\mathcal{C}'_R|$. Observe that $m'_R \leq m_R$.

▶ Remark 33. Similar to Remark 20 and Remark 26, in the algorithmic implementation, we will maintain an auxiliary data structure that will maintain $\mathcal{C}'_R$ to be a pseudomaximum set of disjoint clauses from F_{2R} and whenever we come across a claim that uses this property but is violated, we will reset $\mathcal{C}'_R$ to be the disjoint clauses supplied by the proof and will reset stage κ_1. Such reset can happen at most n times per u_0 and hence, the runtime of the algorithm can be increased by at most n. We will allow remaining claims in this section to use that $\mathcal{C}'_R$ is a maximum sized collection.

5.4 Stage κ_2: 2-marked nodes with effective width 2 clauses

Consider a node u at the end of the stage κ_1. Let $V^{\text{marked}}(u)$ denote the set of marked variables along the root to u path $P(r, u)$. Let $\mathcal{C}'_R(u) = \{C \in \mathcal{C}'_R \mid \exists i \in V^{\text{marked}}(u)$ such that $C \cap X_i \neq \emptyset\}$. Let $\ell(u) = |\mathcal{C}'_R(u)| = |V^{\text{marked}}(u) \cap V'_R|$.

In stage κ_2, we expand using the clauses in $\mathcal{C}'_R(u)$. By Fact 32, these clauses are live at u and disjoint from each other, the expansion will be carried out for exactly $\ell(u)$ levels where each clause in $\mathcal{C}'_R(u)$ will be used to expand one level. We record this fact here:

▶ **Fact 34.** Let u be a node at the end of stage κ_1. Then, stage κ_2 underneath u has length $\ell(u) = |\mathcal{C}'_R(u)| = |V^{\text{marked}}(u) \cap V'_R|$

After this expansion, we finish the controlled stage and enter the arbitrary stage. We record the following useful property of nodes in this stage:

▶ **Lemma 35.** *All nodes in stage κ_2 will have effective width 2 and mass at most $\frac{3}{2}$.*

Proof. Let v be arbitrary node in stage κ_2 developed using clause C where $C \in \mathcal{C}'_R(u)$. Let $i \in V^{\text{marked}}(u)$ be such that $\tilde{x}_i \in C$. As $C_i^0 = (p_i \vee x_i \vee x'_i)$ is a clause in $\mathcal{C}^0$, and F is negation-closed, the negation-clause of C_i^0 is also in F. At v, the negation-clause of C_i^0 simplifies to the unit clause $\neg\tilde{x}_i$. So, at v, the edge with label $\tilde{x}_i$ will be a falsifying edge, making its effective width equal to 2. Moreover, since v is 2-marked, some other edge from C is also marked. This implies the mass of v will indeed be at most $\frac{3}{2}$ as desired. ◀

5.5 Arbitrary Stage

Let u be arbitrary node at the end of the controlled stage. In the controlled stage, we expand using any monotone clause that is available and put no restrictions. However, we will still be able to argue regarding the kinds of clauses that one could encounter in this stage.

We begin by showing that every 1-marked vertex in this stage will have mass at most $\frac{9}{4}$:

▶ **Lemma 36.** *Every 1-marked node in arbitrary stage has mass at most $\frac{9}{4}$.*

Proof. Indeed, if a 1-marked vertex v with clause C in arbitrary stage has a larger mass, then it must have mass $\frac{5}{2}$. In that case, there exists a unique $i \in V_0$ such that C contains exactly one element from X_i. Since mass of v is $\frac{5}{2}$ and $i \notin V_1$, the remaining two variables do not appear in the shoot $S(r, u)$. However, this implies C is disjoint from $\mathcal{C}^1$, contradicting the fact that $\mathcal{C}^1$ is a pseudomaximum disjoint set of clauses with single marking. ◀

▶ **Lemma 37.** *For any root to leaf shoot S in $T(u)$, the number of nodes with 2 marked edges and mass 2 is at most $m'_R + m_B - \ell(u)$.*

Proof. Call such clauses as *heavy clauses*. By Fact 29, heavy clauses along any shoot during arbitrary stage are all in F_2. Observe that these heavy clauses must be disjoint from $\mathcal{C}'_R(u)$. We first claim that there can be at most $m'_R - \ell(u)$ heavy clauses from F_{2R} during arbitrary stage along S. Indeed, if there were more, then these heavy clauses combined with $\mathcal{C}'_R(u)$ would form a collection of disjoint clauses of size at least $(m'_R - \ell(u) + 1) + \ell(u) = m'_R + 1$, violating the fact that $\mathcal{C}'_R$ is a pseudomaximum collection of disjoint clauses.

The only other heavy clauses that can occur in arbitrary stage are from the set F_{2B}. By Fact 30, there are at most m_B disjoint clauses in F_{2B}. Since heavy clauses along a shoot must be disjoint from each other, there can be at most $m'_R + m_B - \ell(u)$ heavy clauses in S. ◀

6 Bounding ψ when $t_0 \leq \frac{n}{4}$

In this section, we show that if in a transversal tree T, the number of maximally disjoint clauses chosen in the disjoint stage is at most $n/4$, then $\psi(r) \leq 6^{n/4}$ where r is the root of T. Formally, our main theorem is:

▶ **Theorem 38.** *Let T be a transversal tree developed using the clause selection criteria as laid out in Section 3 and Section 5 with $t_0 \leq \frac{n}{4}$. Then, $\psi(r) \leq 6^{\frac{n}{4}}$ where r is the root of T.*

To bound $\psi(r)$, we will carefully count and sum up the survival values of all nodes that appear at the end of the disjoint stage. To facilitate that, we need a handle on the survival value of a subtree that is developed in arbitrary stage. We introduce the following quantity to help us with that:

Let $M(w,d,h) = \max_u \psi(u)$ where u is a node at depth $\frac{n}{2} - d$ in arbitrary stage, every root to leaf shoot in $T(u)$ has weight at least w, and for every root to leaf shoot, the number of 2-marked nodes with mass 2 is bounded by h.

As u is a node at level $\frac{n}{2} - d$, the depth of $T(u)$ is d. Moreover, as u is in arbitrary stage, every node in T has at least one marked edge coming out of it, and by Lemma 36, every 1-marked node has that marked edge with survival probability at most $1/4$.

We will define the following useful function:

$$F(w,d,h) = \begin{cases} \left(\frac{9}{4}\right)^{d} & \text{if } w \le d \\ \left(\frac{9}{4}\right)^{2d-w} 2^{w-d} & \text{if } d \le w \le d+h \\ \left(\frac{9}{4}\right)^{2d-w} 2^{h} \left(\frac{27}{8}\right)^{(w-d-h)/2} = 2^{h} \left(\frac{27}{8}\right)^{(3d-w-h)/2} \left(\frac{3}{2}\right)^{w-2d} & \text{if } d+h \le w \le 3d-h \\ 2^{3d-w} \left(\frac{3}{2}\right)^{w-2d} & \text{if } 3d-h \le w \end{cases}$$

We will use the following bound on $M(w,d,h)$ that we prove in the appendix of the full version.

▶ **Lemma 39.** *For all w,d,h: $M(w,d,h) \le F(w,d,h)$.*

With this, we are ready to prove Theorem 38:

Proof of Theorem 38. Let $u_0 \in T$ be an arbitrary node at depth t_0. Then, we can associate quantities $m_B(u_0), m'_R(u_0), t_1(u_0)$ with the subtree $T(u_0)$. We bound $\psi(u_0)$ in terms of these quantities and function F from Lemma 39. We will sum over $\psi(u)$ where u is a node at the end of controlled stage and then use F to bound $\psi(u)$ as u will be at the beginning of arbitrary stage. We will also need to precisely compute $\sigma(u_0,u)$, for such u and for that, we keep track of how many marked edges (say i) from stage κ_1 corresponding to $m'_R(u_0)$ are on the path from u_0 to u. This i will also be the length of stage κ_2, which will further help us in bounding $\sigma(u_0,u)$.

To do that, we first introduce the following quantities that we will use in the upper bound: Let $w(u_0,i) = \frac{n}{2} - 2i - t_1(u_0), d(u_0,i) = \frac{n}{2} - t_0 - t_1(u_0) - i, h(u_0,i) = m'_R(u_0) + m_B(u_0) - i$. Using these quantities, we will upper bound $\psi(u_0)$ using the following expression: Define

$$N(u_0) = \left(\frac{5}{2}\right)^{t_1(u_0)} \left(\frac{4}{5}\right)^{m'_R(u_0)} \sum_{i=0}^{m'_R(u_0)} \binom{m'_R(u_0)}{i} \left(\frac{3}{8}\right)^{i} \cdot F(w(u_0,i), d(u_0,i), h(u_0,i))$$

Using this function, we will obtain the following as our main lemma:

▶ **Lemma 40.** *$\psi(u_0) \le N(u_0)$.*

We assume this bound holds and continue our proof of Theorem 38. We will prove the lemma in the appendix of the full version of the paper.

As $N(u_0)$ depends on F, we want to figure out what case for the function of F applies. Recall that this depends on the relationship between $w(u_0,i), d(u_0,i), h(u_0,i)$. To help with this, we introduce the following function:

$$I(u_0) = 3t_0 + 2t_1(u_0) + m'_R(u_0) + m_B(u_0)$$

We show that the values of I and n govern which of the 4 functions will F equal. This is surprising since I is a function of u_0 and not a function of i. We first show that only 2 function choices of F can arise. We do this by showing:

▷ Claim 41. $d(u_0,i) + h(u_0,i) \le w(u_0,i)$.

Proof of Claim 41. Indeed we compute:

$$\begin{aligned} d(u_0,i)+h(u_0,i) &= \frac{n}{2} - t_0 - t_1(u_0) + m'_R(u_0) + m_B(u_0) - 2i \\ &= w(u_0,i) + (m'_R(u_0) + m_B(u_0) - t_0) \\ &\leq w(u_0,i) \end{aligned}$$

where the last inequality follows because $m'_R(u_0) + m_B(u_0) \leq t_1(u_0) + m_B(u_0) = t_0$. ◁

We next show that the value of I decides the choice function for F:

▷ Claim 42. $w(u_0,i) \leq 3d(u_0,i) - h(u_0,i)$ if and only if $I(u_0) \leq n$.

Proof of Claim 42. We compute the following:

$$\begin{aligned} 3d(u_0,i) - h(u_0,i) &= \frac{3n}{2} - 3t_0 - 3t_1(u_0) - m'_R(u_0) - m_B(u_0) - 2i \\ &= w(u_0,i) + n - 3t_0 - 2t_1(u_0) - m'_R(u_0) - m_B(u_0) \end{aligned}$$

and hence,

$$w(u_0,i) \leq 3d(u_0,i) - h(u_0,i) \iff 3t_0 + 2t_1(u_0) + m'_R(u_0) + m_B(u_0) = I(u_0) \leq n \quad ◁$$

With this, we bound $\psi(r)$ as follows:

$$\begin{aligned} \psi(r) &= \sum_{u_0 : u_0 \text{ at depth } t_0} \psi(u_0) \\ &\leq \sum_{u_0 : u_0 \text{ at depth } t_0} N(u_0) \\ &= \sum_{u_0 : u_0 \text{ at depth } t_0, I(u_0) \leq n} N(u_0) + \sum_{u_0 : u_0 \text{ at depth } t_0, I(u_0) > n} N(u_0) \\ &\leq 3^{t_0} \max\left(\max_{u_0 : I(u_0) \leq n} N(u_0), \max_{u_0 : I(u_0) > n} N(u_0) \right) \\ &= \max\left(\max_{u_0 : I(u_0) \leq n} 3^{t_0} \cdot N(u_0), \max_{u_0 : I(u_0) \geq n} 3^{t_0} \cdot N(u_0) \right) \end{aligned}$$

We show that the inner quantities are maximized when $I(u_0) = n$ by the following two claims. These claims just rely on the inequalities listed in the claim, proving that some expression is bounded. We see these claims as solving an optimization problem and do not rely on any properties of the transversal tree.

▷ Claim 43. Let u_0 be such that $m_B(u_0) + t_1(u_0) \leq t_0, m'_R(u_0) \leq t_1(u_0), I(u_0) \leq n$. Then, $3^{t_0} \cdot N(u_0)$ is maximised when $I(u_0) = n$.

▷ Claim 44. Let u_0 be such that $m_B(u_0) + t_1(u_0) \leq t_0, m'_R(u_0) \leq t_1(u_0), I(u_0) \geq n$. Then, $3^{t_0} \cdot N(u_0)$ is maximised when $I(u_0) = n$.

Lastly, we show that when $I(u_0) = n$, then the inner quantity is bounded by $6^{n/4}$:

▷ Claim 45. Let u_0 be such that $m_B(u_0) + t_1(u_0) \leq t_0, m'_R(u_0) \leq t_1(u_0), I(u_0) = n$. Then, $3^{t_0} \cdot N(u_0) \leq 6^{n/4}$.

These 3 claims together indeed show that $\psi(r) \leq 6^{n/4}$ as desired. We defer the proofs of all these claims to the appendix of the full version of the paper. ◀

7 Conclusion

We gave an optimal algorithm for the Not-All-Equal variant of $\text{Enum}(k, \frac{n}{2})$ for $k = 3$. Extending the analysis of our algorithm to large k would break SSETH. However, extending this to even $k = 4$ poses a great challenge.

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. `doi:10.4230/LIPICS.ISAAC.2023.7`.

2 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`.

3 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`.

4 Mohit Gurumukhani, Marvin Künnemann, and Ramamohan Paturi. On extremal properties of k-cnf: Capturing threshold functions. *CoRR*, abs/2412.20493, 2024. `arXiv:2412.20493`.

5 Mohit Gurumukhani, Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Navid Talebanfard. Local enumeration and majority lower bounds. In Rahul Santhanam, editor, *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPIcs*, pages 17:1–17:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPICS.CCC.2024.17`.

6 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`.

7 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`.

8 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`.

9 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`.

10 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`.

11 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`.

12 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`.

13 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.

14 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.

15 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.

16 Nikhil Vyas and R. Ryan Williams. On super strong ETH. *J. Artif. Intell. Res.*, 70:473–495, 2021. doi:10.1613/JAIR.1.11859.