Fixed-parameter tractability of satisfying beyond the number of variables

Robert Crowston† Gregory Gutin† Mark Jones‡ Venkatesh Raman‡ Saket Saurabh‡ Anders Yeo§

Abstract

We consider a CNF formula $F$ as a multiset of clauses: $F = \{c_1, \ldots, c_m\}$. The set of variables of $F$ will be denoted by $V(F)$. Let $B_F$ denote the bipartite graph with partite sets $V(F)$ and $F$ and with an edge between $v \in V(F)$ and $c \in F$ if $v \in c$ or $\bar{v} \in c$. The matching number $\nu(F)$ of $F$ is the size of a maximum matching in $B_F$. In our main result, we prove that the following parameterization of MaxSat (denoted by $(\nu(F) + k)$-SAT) is fixed-parameter tractable: Given a formula $F$, decide whether we can satisfy at least $\nu(F) + k$ clauses in $F$, where $k$ is the parameter.

A formula $F$ is called variable-matched if $\nu(F) = |V(F)|$. Let $\delta(F) = |F| - |V(F)|$ and $\delta^*(F) = \max_{F \subseteq \tau} \delta(F)$. Our main result implies fixed-parameter tractability of MaxSat parameterized by $\delta(F)$ for variable-matched formulas $F$; this complements related results of Kullmann (2000) and Szeider (2004) for MaxSat parameterized by $\delta^*(F)$.

To obtain our main result, we reduce $(\nu(F) + k)$-SAT into the following parameterization of the Hitting Set problem (denoted by $(m - k)$-Hitting Set): given a collection $C$ of $m$ subsets of a ground set $U$ of $n$ elements, decide whether there is $X \subseteq U$ such that $C \cap X \neq \emptyset$ for each $C \in C$ and $|X| \leq m - k$, where $k$ is the parameter. Gutin, Jones and Yeo (2011) proved that $(m - k)$-Hitting Set is fixed-parameter tractable by obtaining an exponential kernel for the problem. We obtain two algorithms for $(m - k)$-Hitting Set: a deterministic algorithm of runtime $O((2e)^k \log^2 k)(m + n)^{O(1)}$ and a randomized algorithm of expected runtime $O(k^{k+O(\sqrt{k})}(m + n)^{O(1)})$. Our deterministic algorithm improves an algorithm that follows from the kernelization result of Gutin, Jones and Yeo (2011).

1 Introduction

In this paper we study a parameterization of MaxSat. We consider a CNF formula $F$ as a multiset of clauses: $F = \{c_1, \ldots, c_m\}$. (We allow repetition of clauses.) We assume that no clause contains both a variable and its negation, and no clause is empty. The set of variables of $F$ will be denoted by $V(F)$, and for a clause $c$, $V(c) = V(\{c\})$. A truth assignment is a function $\tau : V(F) \rightarrow \{\text{true}, \text{false}\}$. A truth assignment $\tau$ satisfies a clause $C$ if there exists $x \in V(F)$ such that $x \in C$ and $\tau(x) = \text{true}$, or $\bar{x} \in C$ and $\tau(x) = \text{false}$. We will denote the number of clauses in $F$ satisfied by $\tau$ as $\text{sat}_\tau(F)$ and the maximum value of $\text{sat}_\tau(F)$, over all $\tau$, as $\text{sat}(F)$.

Let $B_F$ denote the bipartite graph with partite sets $V(F)$ and $F$ and with an edge between $v \in V(F)$ and $c \in F$ if $v \in V(c)$. The matching number $\nu(F)$ of $F$ is the size of a maximum matching in $B_F$. Clearly, $\text{sat}(F) \geq \nu(F)$ and this lower bound for $\text{sat}(F)$ is tight as there are formulas $F$ for which $\text{sat}(F) = \nu(F)$.

* A preliminary version of this paper appeared in SAT 2012, Lect. Notes Comput. Sci. 7317 (2012), 341–354.
† Royal Holloway, University of London, Egham, Surrey, UK
‡ The Institute of Mathematical Sciences, Chennai 600 113, India
§ University of Johannesburg, Auckland Park, 2006 South Africa
In this paper we study the following parameterized problem, where the parameterization is above a tight lower bound.

\[
(\nu(F) + k)\text{-SAT}
\]

**Instance:** A CNF formula \(F\) and a positive integer \(\alpha\).

**Parameter:** \(k = \alpha - \nu(F)\).

**Question:** Is \(\text{sat}(F) \geq \alpha\)?

A natural and well-studied parameter in most optimization problems is the size of the solution. In particular, for MAXSAT, the standard parameterized problem is whether \(\text{sat}(F) \geq k\) for a CNF formula \(F\). Using a simple observation that \(\text{sat}(F) \geq m/2\) for every CNF formula \(F\) on \(m\) clauses, Mahajan and Raman [21] showed that this problem is fixed-parameter tractable. The tight bound \(\text{sat}(F) \geq m/2\) on \(\text{sat}(F)\) means that the problem is interesting only when \(k > m/2\), i.e., when the values of \(k\) are relatively large. To remedy this situation, Mahajan and Raman introduced, and showed fixed-parameter tractable, a more natural parameterized problem: whether the given CNF formula has an assignment satisfying at least \(m/2 + k\) clauses. Since this pioneering paper [21], researchers have studied numerous problems parameterized above tight bounds including a few such parameterizations of MAXSAT [2], [6], [14], all stated in or inspired by Mahajan et al. [22]. Like the parameterizations in [2, 6, 14], \((\nu(F) + k)\text{-SAT}\) will be proved fixed-parameter tractable, but unlike them, \((\nu(F) + k)\text{-SAT}\) will be shown to have no polynomial-size kernel unless \(\text{coNP} \subseteq \text{NP/poly}\), which is highly unlikely [4].

In our main result, we show that \((\nu(F) + k)\text{-SAT}\) is fixed-parameter tractable by obtaining an algorithm with running time \(O((2e)^{2k + O(\log^2 k)}(n + m)^{O(1)})\), where \(e\) is the base of the natural logarithm. (We provide basic definitions on parameterized algorithms and complexity, including kernelization, in the next section.) We also develop a randomized algorithm for \((\nu(F) + k)\text{-SAT}\) of expected runtime \(O(8^{k + O(\sqrt{k})}(m + n)^{O(1)})\).

The **deficiency** \(\delta(F)\) of a formula \(F\) is \(|F| - |V(F)|\); the **maximum deficiency** \(\delta^*(F) = \max_{F' \subseteq F} \delta(F')\). A formula \(F\) is called **variable-matched** if \(\nu(F) = |V(F)|\). Our main result implies fixed-parameter tractability of MAXSAT parameterized by \(\delta(F)\) for variable-matched formulas \(F\).

There are two related results: Kullmann [18] obtained an \(O(n^{O(\delta^*(F))})\)-time algorithm for solving MAXSAT for formulas \(F\) with \(n\) variables and Szeider [28] gave an \(O(f(\delta^*(F))n^4)\)-time algorithm for the problem, where \(f\) is a function depending on \(\delta^*(F)\) only. Note that we cannot just drop the condition of being variable-matched from our result and expect a similar algorithm: it is not hard to see that the satisfiability problem remains NP-complete for formulas \(F\) with \(\delta(F) = 0\).

A formula \(F\) is **minimal unsatisfiable** if it is unsatisfiable but \(F \setminus c\) is satisfiable for every clause \(c \in F\). Papadimitriou and Wolfe [26] showed that recognition of minimal unsatisfiable CNF formulas is complete for the complexity class \(\text{D}^P\). Kleine B"{u}ning [10] conjectured that for a fixed integer \(k\), it can be decided in polynomial time whether a formula \(F\) with \(\delta(F) \leq k\) is minimal unsatisfiable. Independently, Kullmann [15] and Fleischner and Szeider [11] (see also [10]) resolved this conjecture by showing that minimal unsatisfiable formulas with \(n\) variables and \(n + k\) clauses can be recognized in \(n^{O(k)}\) time. Later, Szeider [23] showed that the problem is fixed-parameter tractable by obtaining an algorithm of running time \(O(2^k n^4)\). Note that Szeider’s results follow from his results mentioned in the previous paragraph and the well-known fact that \(\delta^*(F) = \delta(F)\) holds for every minimal unsatisfiable formula \(F\). Since every minimal unsatisfiable formula is variable-matched [1], our main result also implies fixed-parameter tractability of recognizing minimal unsatisfiable formula with \(n\) variables and \(n + k\) clauses, parameterized by \(k\).

\(^1\text{D}^P\) is the class of problems that can be considered as the difference of two NP-problems; clearly \(\text{D}^P\) contains all NP and all co-NP problems.
To obtain our main result, we introduce some reduction rules and branching steps and reduce the problem to a parameterized version of Hitting Set, namely, \((m-k)\)-Hitting Set defined below. Let \(H\) be a hypergraph. A set \(S \subseteq V(H)\) is called a hitting set if \(e \cap S \neq \emptyset\) for all \(e \in E(H)\).

\[
\text{(m-k)-Hitting Set} \\
\text{Instance:} \text{ A hypergraph } H \ (n = |V(H)|, \ m = |E(H)|) \text{ and a positive integer } k. \\
\text{Parameter: } k. \\
\text{Question:} \text{ Does there exist a hitting set } S \subseteq V(H) \text{ of size } m-k? \\
\]

Gutin et al. [13] showed that \((m-k)\)-Hitting Set is fixed-parameter tractable by obtaining a kernel for the problem. The kernel result immediately implies a \(2^{O(k^2)}(m+n)^{O(1)}\)-time algorithm for the problem. Here we obtain a faster algorithm for this problem that runs in \(O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})\) time using the color-coding technique. This happens to be the dominating step for solving the \((\nu(F) + k)\)-SAT problem. We also obtain a randomized algorithm for \((m-k)\)-Hitting Set of expected runtime \(O(8^{k+O(\sqrt{k})}(m+n)^{O(1)})\). To obtain the randomized algorithm, we reduce \((m-k)\)-Hitting Set into a special case of the Subgraph Isomorphism problem and use a recent randomized algorithm of Fomin et al. [9] for Subgraph Isomorphism.

It was shown in [13] that the \((m-k)\)-Hitting Set problem cannot have a kernel whose size is polynomial in \(k\) unless \(NP \subseteq coNP/poly\). In this paper, we give a parameter preserving reduction from this problem to the \((\nu(F) + k)\)-SAT problem, thereby showing that \((\nu(F) + k)\)-SAT problem has no polynomial-size kernel unless \(NP \subseteq coNP/poly\).

**Organization of the rest of the paper.** In Section 2 we provide additional terminology and notation and some preliminary results. In Section 3 we give a sequence of polynomial time preprocessing rules on the given input of \((\nu(F) + k)\)-SAT and justify their correctness. In Section 4 we give two simple branching rules and reduce the resulting input to a \((m-k)\)-Hitting Set problem instance. Section 5 gives an improved fixed-parameter algorithm for \((m-k)\)-Hitting Set using color coding. There we also obtain a faster randomized algorithm for \((m-k)\)-Hitting Set. Section 6 summarizes the entire algorithm for the \((\nu(F) + k)\)-SAT problem, shows its correctness and analyzes its running time. Section 7 proves the hardness of kernelization result. Section 8 concludes with some remarks.

### 2 Additional Terminology, Notation and Preliminaries

**Graphs and Hypergraphs.** For a subset \(X\) of vertices of a graph \(G\), \(N_G(X)\) denotes the set of all neighbors of vertices in \(X\). When \(G\) is clear from the context, we write \(N(X)\) instead of \(N_G(X)\). A matching saturates all end-vertices of its edges. For a bipartite graph \(G = (V_1, V_2; E)\), the classical Hall’s matching theorem states that \(G\) has a matching that saturates every vertex of \(V_1\) if and only if \(|N(X)| \geq |X|\) for every subset \(X\) of \(V_1\). The next lemma follows from Hall’s matching theorem: add \(d\) vertices to \(V_2\), each adjacent to every vertex in \(V_1\).

**Lemma 1.** Let \(G = (V_1, V_2; E)\) be a bipartite graph, and suppose that for all subsets \(X \subseteq V_1\), \(|N(X)| \geq |X| - d\) for some \(d \geq 0\). Then \(\nu(G) \geq |V_1| - d\).

We say that a bipartite graph \(G = (A, B; E)\) is \(q\)-expanding if for all \(A' \subseteq A\), \(|N_G(A')| \geq |A'| + q\). Given a matching \(M\), an alternating path is a path in which the edges belong alternatively to \(M\) and not to \(M\).

A hypergraph \(H = (V(H), F)\) consists of a nonempty set \(V(H)\) of vertices and a family \(F\) of nonempty subsets of \(V\) called edges of \(H\) (\(F\) is often denoted \(E(H)\)). Note that \(F\) may have parallel edges, i.e., copies of the same subset of \(V(H)\). For any vertex \(v \in V(H)\), and any \(E \subseteq F\),
\(\mathcal{E}[v]\) is the set of edges in \(\mathcal{E}\) containing \(v\), and \(N[v]\) is the set of all vertices contained in edges of \(\mathcal{F}[v]\). For a subset \(T\) of vertices, \(\mathcal{F}[T] = \bigcup_{v \in T} \mathcal{F}[v]\).

**CNF formulas.** For a subset \(X\) of the variables of CNF formula \(F\), \(F_X\) denotes the subset of \(F\) consisting of all clauses \(c\) such that \(V(c) \cap X \neq \emptyset\). A formula \(F\) is called \(q\)-expanding if \(|X| + q \leq |F_X|\) for each \(X \subseteq V(F)\). Note that, by Hall’s matching theorem, a formula is variable-expanded if and only if it is 0-expanding. Clearly, a formula \(F\) is \(q\)-expanding if and only if \(B_F\) is \(q\)-expanding.

For \(x \in V(F)\), \(n(x)\) and \(n(\bar{x})\) denote the number of clauses containing \(x\) and the number of clauses containing \(\bar{x}\), respectively.

A function \(\pi : U \to \{\text{true}, \text{false}\}\), where \(U\) is a subset of \(V(F)\), is called a partial truth assignment. A partial truth assignment \(\pi : U \to \{\text{true}, \text{false}\}\) is an autarky if \(\pi\) satisfies all clauses of \(F_U\). We have the following:

**Lemma 2** ([6]). Let \(\pi : U \to \{\text{true}, \text{false}\}\) be an autarky for a CNF formula \(F\) and let \(\gamma\) be any truth assignment on \(V(F) \setminus U\). Then for the combined assignment \(\tau := \pi \cup \gamma\), it holds that \(\text{sat}_\tau(F) = |F_U| + \text{sat}_\gamma(F \setminus F_U)\). Clearly, \(\tau\) can be constructed in polynomial time given \(\pi\) and \(\gamma\).

Autarkies were first introduced in [23]; they are the subject of much study, see, e.g., [10, 19, 28], and see [7] for an overview.

**Treewidth.** A tree decomposition of an (undirected) graph \(G\) is a pair \((U, T)\) where \(T\) is a tree whose vertices we will call nodes and \(U = \{\{U_i \mid i \in V(T)\}\}\) is a collection of subsets of \(V(G)\) such that

1. \(\bigcup_{i \in V(T)} U_i = V(G)\),
2. for each edge \(vw \in E(G)\), there is an \(i \in V(T)\) such that \(v, w \in U_i\), and
3. for each \(v \in V(G)\) the set \(\{i : v \in U_i\}\) of nodes forms a subtree of \(T\).

The \(U_i\)’s are called bags. The width of a tree decomposition \((\{U_i : i \in V(T)\}, T)\) equals \(\max_{i \in V(T)}|U_i| - 1\). The treewidth of a graph \(G\) is the minimum width over all tree decompositions of \(G\). We use notation \(\text{tw}(G)\) to denote the treewidth of a graph \(G\).

**Parameterized Complexity.** A parameterized problem is a subset \(L \subseteq \Sigma^* \times \mathbb{N}\) over a finite alphabet \(\Sigma\). The unparameterized version of a parameterized problem \(L\) is the language \(L^c = \{x \# 1^k \mid (x, k) \in L\}\). The problem \(L\) is fixed-parameter tractable if the membership of an instance \((x, k)\) in \(\Sigma^* \times \mathbb{N}\) can be decided in time \(f(k)|x|^{O(1)}\), where \(f\) is a function of the parameter \(k\) only ([8, 12, 24]). Given a parameterized problem \(L\), a kernelization of \(L\) is a polynomial-time algorithm that maps an instance \((x, k)\) to an instance \((x', k')\) (the kernel) such that (i) \((x, k) \in L\) if and only if \((x', k') \in L\), (ii) \(k' \leq g(k)\), and (iii) \(|x'| \leq g(k)\) for some function \(g\). We call \(g(k)\) the size of the kernel. It is well-known ([8, 12]) that a decidable parameterized problem \(L\) is fixed-parameter tractable if and only if it has a kernel. Polynomial-size kernels are of main interest, due to applications ([8, 12, 24]), but unfortunately not all fixed-parameter problems have such kernels unless \(\text{coNP} \subseteq \text{NP/poly}\), see, e.g., [4, 5, 7].

For a positive integer \(q\), let \([q] = \{1, \ldots, q\}\).

### 3 Preprocessing Rules

In this section we give preprocessing rules and their correctness.

Let \(F\) be the given CNF formula on \(n\) variables and \(m\) clauses with a maximum matching \(M\) on \(B_F\), the variable-clause bipartite graph corresponding to \(F\). Let \(\alpha\) be a given integer and recall
that our goal is to check whether \( \text{sat}(F) \geq \alpha \). For each preprocessing rule below, we let \((F', \alpha')\) be the instance resulting by the application of the rule on \((F, \alpha)\). We say that a rule is valid if \((F, \alpha)\) is a Yes instance if and only if \((F', \alpha')\) a Yes instance.

**Reduction Rule 1.** Let \( x \) be a variable such that \( n(x) = 0 \) (respectively \( n(\bar{x}) = 0 \)). Set \( x = \text{false} \) (\( x = \text{true} \)) and remove all the clauses that contain \( \bar{x} \) (\( x \)). Reduce \( \alpha \) by \( n(\bar{x}) \) (respectively \( n(x) \)).

The proof of the following lemma is immediate.

**Lemma 3.** If \( n(x) = 0 \) (respectively \( n(\bar{x}) = 0 \)) then \( \text{sat}(F) = \text{sat}(F') + n(\bar{x}) \) (respectively \( \text{sat}(F) = \text{sat}(F') + n(x) \)), and so Rule 1 is valid.

**Reduction Rule 2.** Let \( n(x) = n(\bar{x}) = 1 \) and let \( c' \) and \( c'' \) be the two clauses containing \( x \) and \( \bar{x} \), respectively. Let \( c^* = (c' - x) \cup (c'' - \bar{x}) \) and let \( F'' \) be obtained from \( F \) by deleting \( c' \) and \( c'' \) and adding the clause \( c^* \). Reduce \( \alpha \) by 1.

**Lemma 4.** For \( F \) and \( F'' \) in Reduction Rule 2 \( \text{sat}(F) = \text{sat}(F'') + 1 \), and so Rule 2 is valid.

Proof. Consider any assignment for \( F \). If it satisfies both \( c' \) and \( c'' \), then the same assignment will satisfy \( c^* \). So when restricted to variables of \( F'' \), it will satisfy at least \( \text{sat}(F) - 1 \) clauses of \( F'' \). Thus \( \text{sat}(F') \geq \text{sat}(F) - 1 \) which is equivalent to \( \text{sat}(F) \leq \text{sat}(F') + 1 \). Similarly if an assignment \( \gamma \) to \( F' \) satisfies \( c^* \) then at least one of \( c', c'' \) is satisfied by \( \gamma \). Therefore by setting \( x \) true if \( \gamma \) satisfies \( c'' \) and false otherwise, we can extend \( \gamma \) to an assignment on \( F \) that satisfies both of \( c', c'' \). On the other hand, if \( c^* \) is not satisfied by \( \gamma \) then neither \( c' \) nor \( c'' \) is satisfied by \( \gamma \), and any extension of \( \gamma \) will satisfy exactly one of \( c', c'' \). Therefore in either case \( \text{sat}(F) \geq \text{sat}(F') + 1 \). We conclude that \( \text{sat}(F') = \text{sat}(F'') + 1 \), as required.

Our next reduction rule is based on the following lemma proved in Fleischer et al. [10] Lemma 10], Kullmann [19] Lemma 7.7] and Szeider [28] Lemma 9].

**Lemma 5.** Let \( F \) be a CNF formula. Given a maximum matching in \( BF \), in time \( O(|F|) \) we can find an autarky \( \pi : U \rightarrow \{\text{true}, \text{false}\} \) such that \( F \setminus F_U \) is 1-expanding.

**Reduction Rule 3.** Find an autarky \( \pi : U \rightarrow \{\text{true}, \text{false}\} \) such that \( F \setminus F_U \) is 1-expanding. Set \( F' = F \setminus F_U \) and reduce \( \alpha \) by \( |F_U| \).

The next lemma follows from Lemma 2.

**Lemma 6.** For \( F \) and \( F'' \) in Reduction Rule 2 \( \text{sat}(F) = \text{sat}(F'') + |F_U| \) and so Rule 2 is valid.

After exhaustive application of Rule 2 we may assume that the resulting formula is 1-expanding.

For the next reduction rule, we need the following results.

**Theorem 1** (Szeider [28]). Given a variable-matched formula \( F \), with \(|F| = |V(F)| + 1 \), we can decide whether \( F \) is satisfiable in time \( O(|V(F)|^4) \).

Consider a bipartite graph \( G = (A, B; E) \). Recall that a formula \( F \) is \( q \)-expanding if and only if \( BF \) is \( q \)-expanding. From a bipartite graph \( G = (A, B; E) \), \( x \in A \) and \( q \geq 1 \), we obtain a bipartite graph \( G_{qx} \), by adding new vertices \( x_1, \ldots, x_q \) to \( A \) and adding edges such that new vertices have exactly the same neighborhood as \( x \), that is, \( G_{qx} = (A \cup \{x_1, \ldots, x_q\}, B; E \cup \{(x_i, y) : (x, y) \in E\}) \). The following result is well known.

**Lemma 7.** [20] Theorem 1.3.6] Let \( G = (A, B; E) \) be a 0-expanding bipartite graph. Then \( G \) is \( q \)-expanding if and only if \( G_{qx} \) is 0-expanding for all \( x \in A \).

**Lemma 8.** Let \( G = (A, B; E) \) be a 1-expanding bipartite graph. In polynomial time, we can check whether \( G \) is 2-expanding, and if it is not, find a set \( S \subseteq A \) such that \(|N_G(S)| = |S| + 1 \).
Proof. Let $x \in A$. By Hall’s Matching Theorem, $G_{2x}$ is 0-expanding if and only if $\nu(G_{2x}) = |A| + 2$. Since we can check the last condition in polynomial time, by Lemma 7 we can decide whether $G$ is 2-expanding in polynomial time. So, assume that $G$ is not 2-expanding and we know this because $G_{2x}$ is not 0-expanding for some $y \in A$. By Lemma 3(4) in [28], in polynomial time, we can find a set $T \subseteq A \cup \{y_1, y_2\}$ such that $|N_{G_{2x}}(T)| < |T|$. Since $G$ is 1-expanding, $y_1, y_2 \in T$ and $|N_{G_{2x}}(T)| = |T| - 1$. Hence, $|S| + 1 = |N_G(S)|$, where $S = T \setminus \{y_1, y_2\}$. \[ \]

For a formula $F$ and a set $S \subseteq V(F)$, $F|S$ denotes the formula obtained from $F_S$ by deleting all variables not in $S$.

**Reduction Rule 4.** Let $F$ be a 1-expanding formula and let $B = B_F$. Using Lemma 8, check whether $F$ is 2-expanding. If it is then do not change $F$, otherwise find a set $S \subseteq V(F)$ with $|N_B(S)| = |S| + 1$. Let $M$ be a matching that saturates $S$ in $B[S \cup N_B(S)]$ (that exists as $B[S \cup N_B(S)]$ is 1-expanding). Use Theorem 7 to decide whether $F|S$ is satisfiable, and proceed as follows.

**$F|S$ is satisfiable:** Obtain a new formula $F'$ by removing all clauses in $N_B(S)$ from $F$. Reduce $\alpha$ by $|N_B(S)|$.

**$F|S$ is not satisfiable:** Let $c'$ be the clause obtained by deleting all variables in $S$ from $\cup_{c'' \in N_B(S)} c''$. That is, a literal $l$ belongs to $c'$ if and only if it belongs to some clause in $N_B(S)$ and the variable corresponding to $l$ is not in $S$. Obtain a new formula $F''$ by removing all clauses in $N_B(S)$ from $F$ and adding $c'$. Reduce $\alpha$ by $|S|$.

**Lemma 9.** For $F$, $F'$, and $S$ introduced in Rule 4, if $F|S$ is satisfiable $\operatorname{sat}(F) = \operatorname{sat}(F') + |N_B(S)|$, otherwise $\operatorname{sat}(F) = \operatorname{sat}(F'') + |S|$ and thus Rule 4 is valid.

Proof. We consider two cases.

**Case 1:** $F|S$ is satisfiable. Observe that there is an autarky on $S$ and thus by Lemma 8 $\operatorname{sat}(F) = \operatorname{sat}(F') + |N_B(S)|$.

**Case 2:** $F|S$ is not satisfiable. Let $F'' = F' \setminus c'$. As any optimal truth assignment to $F$ will satisfy at least $\operatorname{sat}(F) - |N_B(S)|$ clauses of $F''$, it follows that $\operatorname{sat}(F) \leq \operatorname{sat}(F'') + |N_B(S)| \leq \operatorname{sat}(F'') + |N_B(S)|$.

Let $y$ denote the clause in $N_B(S)$ that is not matched to a variable in $S$ by $M$. Let $S'$ be the set of variables, and $Z$ the set of clauses, that can be reached from $y$ with an $M$-alternating path in $B[S \cup N_B(S)]$. We argue now that $Z = N_B(S)$. Since $Z$ is made up of clauses that are reachable in $B[S \cup N_B(S)]$ by an $M$-alternating path from the single unmatched clause $y$, $|Z| = |S'| + 1$. It follows that $|N_B(S)\setminus Z| = |S|\setminus|S'|$, and $M$ matches every clause in $N_B(S)\setminus Z$ with a variable in $S'\setminus S'$. Furthermore, $N_B(S\setminus S') \cap Z = \emptyset$ as otherwise the matching partners of some elements of $S\setminus S'$ would have been reachable by an $M$-alternating path from $y$, contradicting the definition of $N_B(S) \setminus S'$.

Thus $S \setminus S'$ has an autarky such that $F \setminus F_{S \setminus S'}$ is 1-expanding which would have been detected by Rule 3 hence $S \setminus S' = \emptyset$ and so $S = S'$. That is, all clauses in $N_B(S)$ are reachable from the unmatched clause $y$ by an $M$-alternating path. We have now shown that $Z = N_B(S)$, as desired.

Suppose that there exists an assignment $\gamma$ to $F'$, that satisfies $\operatorname{sat}(F')$ clauses of $F'$ that also satisfies $c'$. Then there exists a clause $c'' \in N_B(S)$ that is satisfied by $\gamma$. As $c''$ is reachable from $y$ by an $M$-alternating path, we can modify $M$ to include $y$ and exclude $c''$, by taking the symmetric difference of the matching and the $M$-alternating path from $y$ to $c''$. This will give a matching saturating $S$ and $N_B(S) \setminus c''$, and we use this matching to extend the assignment $\gamma$ to one which satisfies all of $N_B(S) \setminus c''$. We therefore have satisfied all the clauses of $N_B(S)$. Therefore since $c'$ is satisfied in $F'$ but does not appear in $F$, we have satisfied extra $|N_B(S)| - 1 = |S|$ clauses.

Suppose on the other hand that every assignment $\gamma$ for $F'$ that satisfies $\operatorname{sat}(F')$ clauses does not
satisfy \( c' \). We can use the matching on \( B[S \cup N_B(S)] \) to satisfy \( |N_B(S)| - 1 \) clauses in \( N_B(S) \), which would give us an additional \( |S| \) clauses in \( N_B(S) \). Thus \( \text{sat}(F) \geq \text{sat}(F') + |S| \).

As \( |N_B(S)| = |S| + 1 \), it suffices to show that \( \text{sat}(F) < \text{sat}(F') + |N_B(S)| \). Suppose that there exists an assignment \( \gamma \) to \( F \) that satisfies \( \text{sat}(F') + |N_B(S)| \) clauses, then it must satisfy all the clauses of \( N_B(S) \) and \( \text{sat}(F') \) clauses of \( F'' \). As \( F[S] \) is not satisfiable, variables in \( S \) alone can not satisfy all of \( N_B(S) \). Hence there exists a clause \( c'' \in N_B(S) \) such that there is a variable \( v \in V(c'') \setminus S \) that satisfies \( c'' \). But then \( v \in V(c') \) and hence \( c' \) would be satisfiable by \( \gamma \), a contradiction as \( \gamma \) satisfies \( \text{sat}(F') \) clauses of \( F'' \).

\[ \square \]

\section{Branching Rules and Reduction to \((m - k)\)-Hitting Set}

Our algorithm first applies Reduction Rules 1, 2, 3 and 4 exhaustively on \((F, \alpha)\). Then it applies two branching rules we describe below, in the following order.

Branching on a variable \( x \) means that the algorithm constructs two instances of the problem, one by substituting \( x = \text{true} \) and simplifying the instance and the other by substituting \( x = \text{false} \) and simplifying the instance. Branching on \( x \) or \( y \) being false means that the algorithm constructs two instances of the problem, one by substituting \( x = \text{false} \) and simplifying the instance and the other by substituting \( y = \text{false} \) and simplifying the instance. Simplifying an instance is done as follows. For any clause \( c \), if \( c \) contains a literal \( z \) with \( z = \text{true} \), remove \( c \) and reduce \( \alpha \) by 1. If \( c \) contains a literal \( z \) with \( z = \text{false} \) and \( c \) contains other literals, remove \( z \) from \( c \). If \( c \) consists of the single literal \( z = \text{false} \), remove \( c \).

A branching rule is correct if the instance on which it is applied is a \( \text{Yes} \)-instance if and only if the simplified instance of (at least) one of the branches is a \( \text{Yes} \)-instance.

**Branching Rule 1.** If \( n(x) \geq 2 \) and \( n(\bar{x}) \geq 2 \) then we branch on \( x \).

Before attempting to apply Branching Rule 2 we apply the following rearranging step: For all variables \( x \) such that \( n(x) = 1 \), swap literals \( x \) and \( \bar{x} \) in all clauses. Clearly, this will not change \( \text{sat}(F) \). Observe that now for every variable \( n(x) = 1 \) and \( n(\bar{x}) \geq 2 \).

**Branching Rule 2.** If there is a clause \( c \) such that positive literals \( x, y \in c \) then we branch on \( x \) being false or \( y \) being false.

Branching Rule 1 is exhaustive and thus its correctness also follows. When we reach Branching Rule 2 for every variable \( n(x) = 1 \) and \( n(\bar{x}) \geq 2 \). As \( n(x) = 1 \) and \( n(y) = 1 \) we note that \( c \) is the only clause containing these literals. Therefore there exists an optimal solution with \( x \) or \( y \) being false (if they are both true just change one of them to false). Thus, we have the following:

**Lemma 10.** Branching Rules 1 and 2 are correct.

Let \((F, \alpha)\) be the given instance on which Reduction Rules 1, 2, 3 and 4 and Branching Rules 1 and 2 do not apply. Observe that for such an instance \( F \) the following holds:

1. For every variable \( x, n(x) = 1 \) and \( n(\bar{x}) \geq 2 \).
2. Every clause contains at most one positive literal.

We call a formula \( F \) satisfying the above properties special. In what follows we describe an algorithm for our problem on special instances. Let \( c(x) \) denote the unique clause containing positive literal \( x \). We can obtain a matching saturating \( V(F) \) in \( B_F \) by taking the edge connecting the variable \( x \) and the clause \( c(x) \). We denote the resulting matching by \( M_u \).

We first describe a transformation that will be helpful in reducing our problem to \((m - k)\)-Hitting Set. Given a formula \( F \) we obtain a new formula \( F' \) by changing the clauses of \( F \) as follows. If there exists some \( c(x) \) such that \( |c(x)| \geq 2 \), do the following. Let \( c' = c(x) - x \) (that is,
Lemma 11. Let $F'$ be the formula obtained by applying the transformation described above on $F$. Then $\text{sat}(F') = \text{sat}(F)$ and $\nu(BF) = \nu(BF')$.

Proof. We note that the matching $M_u$ remains a matching in $BF'$ and thus $\nu(BF) = \nu(BF')$. Let $\gamma$ be any truth assignment to the variables in $F$ (and $F'$) and note that if $c'$ is false under $\gamma$ then $F$ and $F'$ satisfy exactly the same clauses under $\gamma$ (as we add and subtract something false to the clauses). So assume that $c'$ is true under $\gamma$.

If $\gamma$ maximizes the number of satisfied clauses in $F$ then clearly we may assume that $x$ is false (as $c(x)$ is true due to $c'$). Now let $\gamma'$ be equal to $\gamma$ except the value of $x$ has been flipped to true. Note that exactly the same clauses are satisfied in $F$ and $F'$ by $\gamma$ and $\gamma'$, respectively. Analogously, if an assignment maximizes the number of satisfied clauses in $F'$ we may assume that $x$ is true and by changing it to false we satisfy equally many clauses in $F$. Hence, $\text{sat}(F') = \text{sat}(F)$. 

Given a special instance $(F, \alpha)$ we apply the above transformation repeatedly until no longer possible and obtain an instance $(F', \alpha)$ such that $\text{sat}(F') = \text{sat}(F)$, $\nu(BF) = \nu(BF')$ and $|c(x)| = 1$ for all $x \in V(F')$. We call such an instance $(F', \alpha)$ transformed special. Observe that, it takes polynomial time, to obtain the transformed special instance from a given special instance.

For simplicity of presentation we denote the transformed special instance by $(F, \alpha)$. Let $C^*$ denote all clauses that are not matched by $M_u$ (and therefore only contain negated literals). We associate a hypergraph $H^*$ with the transformed special instance. Let $H^*$ be the hypergraph with vertex set $V(F)$ and edge set $E^* = \{V(c) \mid c \in C^*\}$.

We now show the following equivalence between $(\nu(F) + k)$-SAT on transformed special instances and $(m - k)$-Hitting Set.

Lemma 12. Let $(F, \alpha)$ be the transformed special instance and $H^*$ be the hypergraph associated with it. Then $\text{sat}(F) \geq \alpha$ if and only if there is a hitting set in $H^*$ of size at most $|E(H^*)| - k$, where $k = \alpha - \nu(F)$.

Proof. We start with a simple observation about an assignment satisfying the maximum number of clauses of $F$. There exists an optimal truth assignment to $F$, such that all clauses in $C^*$ are true. Assume that this is not the case and let $\gamma$ be an optimal truth assignment satisfying as many clauses from $C^*$ as possible and assume that $c \in C^*$ is not satisfied. Let $\bar{x} \in c$ be an arbitrary literal and note that $\gamma(\bar{x}) = \text{true}$. However, changing $x$ to false does not decrease the number of satisfied clauses in $F$ and increases the number of satisfied clauses in $C^*$.

Now we show that $\text{sat}(F) \geq \alpha$ if and only if there is a hitting set in $H^*$ of size at most $|E(H^*)| - k$. Assume that $\gamma$ is an optimal truth assignment to $F$, such that all clauses in $C^*$ are true. Let $U \subseteq V(F)$ be all variables that are false in $\gamma$ and note that $U$ is a hitting set in $H^*$. Analogously if $U'$ is a hitting set in $H^*$ then by letting all variables in $U'$ be false and all other variables in $V(F)$ be true we get a truth assignment that satisfies $|F| - |U'|$ clauses in $F$. Therefore if $\tau(H^*)$ is the size of a minimum hitting set in $H^*$ we have $\text{sat}(F) = |F| - \tau(H^*)$. Hence, $\text{sat}(F) = |F| - \tau(H^*) = |V(F)| + |C^*| - \tau(H^*)$ and thus $\text{sat}(F) \geq \alpha$ if and only if $|C^*| - \tau(H^*) \geq k$, which is equivalent to $\tau(H^*) \leq |E(H^*)| - k$.

Therefore our problem is fixed-parameter tractable on transformed special instances, by the next theorem that follows from the kernelization result in [13].

Theorem 2. There exists an algorithm for $(m - k)$-Hitting Set running in time $2^{O(k^2)} + O((n + m)^{O(1)})$.

In the next section we give faster algorithms for $(\nu(F) + k)$-SAT on transformed special instances by giving faster algorithms for $(m - k)$-Hitting Set.
5 Algorithms for \((m - k)\)-Hitting Set

To obtain faster algorithms for \((m - k)\)-Hitting Set, we utilize the following concept of \(k\)-mini-hitting set introduced in [13].

**Definition 1.** Let \(H = (V, F)\) be a hypergraph and \(k\) be a nonnegative integer. A \(k\)-mini-hitting set is a set \(S_{\text{mini}} \subseteq V\) such that \(|S_{\text{mini}}| \leq k\) and \(|F[S_{\text{mini}}]| \geq |S_{\text{mini}}| + k\).

**Lemma 13 ([13]).** A hypergraph \(H\) has a hitting set of size at most \(m - k\) if and only if it has a \(k\)-mini-hitting set. Moreover, given a \(k\)-mini-hitting set \(S_{\text{mini}}\), we can construct a hitting set \(S\) with \(|S| \leq m - k\) such that \(S_{\text{mini}} \subseteq S\) in polynomial time.

### 5.1 Deterministic Algorithm

Next we give an algorithm that finds a \(k\)-mini-hitting set \(S_{\text{mini}}\) if it exists, in time \(c^k(m + n)^{O(1)}\), where \(c\) is a constant. We first describe a randomized algorithm based on color-coding [3] and then derandomize it using hash functions. Let \(\chi : E(H) \to [q]\) be a function. For a subset \(S \subseteq V(H)\), \(\chi(S)\) denotes the maximum subset \(X \subseteq [q]\) such that for all \(i \in X\) there exists an edge \(e \in E(H)\) with \(\chi(e) = i\) and \(e \cap S \neq \emptyset\). A subset \(S \subseteq V(H)\) is called a colorful hitting set if \(\chi(S) = [q]\). We now give a procedure that given a coloring function \(\chi\) finds a minimum colorful hitting set, if it exists. This algorithm will be useful in obtaining a \(k\)-mini-hitting set \(S_{\text{mini}}\).

**Lemma 14.** Given a hypergraph \(H\) and a coloring function \(\chi : E(H) \to [q]\), we can find a minimum colorful hitting set if there exists one in time \(O(2^q(m + n))\).

**Proof.** We first check whether for every \(i \in [q]\), \(\chi^{-1}(i) \neq \emptyset\). If for any \(i\) we have that \(\chi^{-1}(i) = \emptyset\), then we return that there is no colorful hitting set. So we may assume that for all \(i \in [q]\), \(\chi^{-1}(i) \neq \emptyset\). We will give an algorithm using dynamic programming over subsets of \([q]\). Let \(\gamma\) be an array of size \(2^q\) indexed by the subsets of \([q]\). For a subset \(X \subseteq [q]\), let \(\gamma[X]\) denote the size of a smallest set \(W \subseteq V(H)\) such that \(X \subseteq \chi(W)\). We obtain a recurrence for \(\gamma[X]\) as follows:

\[
\gamma[X] = \begin{cases} 
\min_{e \in V(H) \cup \{v\}, (e \cap X) \neq \emptyset} \{1 + \gamma[X \setminus \chi(\{v\})]\} & \text{if } |X| \geq 1, \\
0 & \text{if } X = \emptyset.
\end{cases}
\]

The correctness of the above recurrence is clear. The algorithm computes \(\gamma[[q]]\) by filling the \(\gamma\) in the order of increasing set sizes. Clearly, each cell can be filled in time \(O(q(m + n))\) and thus the whole array can be filled in time \(O(2^q(m + n))\). The size of a minimum colorful hitting set is given by \(\gamma[[q]]\). We can obtain a minimum colorful hitting set by the routine back-tracking.

Now we describe a randomized procedure to obtain a \(k\)-mini-hitting set \(S_{\text{mini}}\) in a hypergraph \(H\), if there exists one. We do the following for each possible value \(p\) of \(|S_{\text{mini}}|\) (that is, for \(1 \leq p \leq k\)). Color \(E(H)\) uniformly at random with colors from \([p + k]\); we denote this random coloring by \(\chi\). Assume that there is a \(k\)-mini-hitting set \(S_{\text{mini}}\) of size \(p\) and some \(p + k\) edges \(e_1, \ldots, e_{p+k}\) such that for all \(i \in [p+k], e_i \cap S_{\text{mini}} \neq \emptyset\). The probability that for all \(1 \leq i < j \leq p + k\) we have that \(\chi(e_i) \neq \chi(e_j)\) is \(\frac{1}{(p+k)^{p+k}} \geq e^{-(p+k)} \geq e^{-2k}\). Now, using Lemma [14] we can test in time \(O(2^{2p+k}(p + k)(m + n))\) whether there is a colorful hitting set of size at most \(p\). Thus with probability at least \(e^{-2k}\) we can find a \(S_{\text{mini}}\), if there exists one. To boost the probability we repeat the procedure \(e^{2k}\) times and thus in time \(O((2e)^{2k}k(m + n)^{O(1)})\) we find a \(S_{\text{mini}}\), if there exists one, with probability at least \(1 - (1 - \frac{1}{e^{2k}})^{e^{2k}} \geq \frac{1}{2}\). If we obtained \(S_{\text{mini}}\) then using Lemma [13] we can construct a hitting set of \(H\) of size at most \(m - k\).

To derandomize the procedure, we need to replace the first step of the procedure where we color the edges of \(E(H)\) uniformly at random from the set \([p + k]\) to a deterministic one. This is done by making use of an \((m, p + k, p + k)\)-perfect hash family. An \((m, p + k, p + k)\)-perfect hash family,
perfect hash family $H$ is a set of functions from $[m]$ to $[p+k]$ such that for every subset $S \subseteq [m]$ of size $p+k$ there exists a function $f \in H$ such that $f$ is injective on $S$. That is, for all $i, j \in S$, $f(i) \neq f(j)$. There exists a construction of an $(m, p+k, p+k)$-perfect hash family of size $O(e^{p+k} \cdot k^{O(\log k)} \cdot \log m)$ and one can produce this family in time linear in the output size [27]. Using an $(m, p+k, p+k)$-perfect hash family $H$ of size at most $O(e^{2k} \cdot k^{O(\log k)} \cdot \log m)$ rather than a random coloring we get the desired deterministic algorithm. To see this, it is enough to observe that if there is a subset $S_{\text{mini}} \subseteq V(H)$ such that $|F[S_{\text{mini}}]| \geq |S_{\text{mini}}| + k$ then there exists a coloring $f \in H$ such that the $p+k$ edges $e_1, \ldots, e_{p+k}$ that intersect $S_{\text{mini}}$ are distinctly colored. So if we generate all colorings from $H$ we will encounter the desired $f$. Hence for the given $f$, when we apply Lemma [4] we get the desired result. This concludes the description. The total time of the derandomized algorithm is $O(k^{2k}(m+n)e^{2k} \cdot k^{O(\log k)} \cdot \log m) = O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$.

Theorem 3. There exists an algorithm solving $(m-k)$-Hitting Set in time $O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$.

By Theorem 3 and the transformation discussed in Section 3 we have the following theorem.

Theorem 4. There exists an algorithm solving a transformed special instance of $(\nu(F)+k)$-SAT in time $O((2e)^{2k+O(\log^2 k)}(m+n)^{O(1)})$.

5.2 Randomized Algorithm

In this subsection we give a randomized algorithm for $(m-k)$-Hitting Set running in time $O(8k^{O(\sqrt{T})}(m+n)^{O(1)})$. However, unlike the algorithm presented in the previous subsection we do not know how to derandomize this algorithm. Essentially, we give a randomized algorithm to find a $k$-mini-hitting set $S_{\text{mini}}$ in the hypergraph $H$, if it exists.

Towards this we introduce notions of a star-forest and a bush. We call $K_{1,\ell}$ a star of size $\ell$; a vertex of degree $\ell$ in $K_{1,\ell}$ is a central vertex (thus, both vertices in $K_{1,1}$ are central). A star-forest is a forest consisting of stars. A star-forest $F$ is said to have dimension $(a_1, a_2, \ldots, a_p)$ if $F$ has $p$ stars with sizes $a_1, a_2, \ldots, a_p$ respectively. Given a star-forest $F$ of dimension $(a_1, a_2, \ldots, a_p)$, we construct a graph, which we call a bush of dimension $(a_1, a_2, \ldots, a_p)$, by adding a triangle $(x, y, z)$ and making $y$ adjacent to every vertex in the central vertex of every star of $F$.

For a hypergraph $H = (V, F)$, the incidence bipartite graph $B_H$ of $H$ has partite sets $V$ and $F$, and there is an edge between $v \in V$ and $e \in F$ in $H$ if $v \in e$. Given $B_H$, we construct $B'_H$ by adding a triangle $(x, y, z)$ and making $y$ adjacent to every vertex in the $V$. The following lemma relates $k$-mini-hitting sets to bushes.

Lemma 15. A hypergraph $H = (V, F)$ has a $k$-mini-hitting set $S_{\text{mini}}$ if and only if there exists a tuple $(a_1, \ldots, a_p)$ such that

(a) $p \leq k$, $a_i \geq 1$ for all $i \in [p]$, and $\sum_{i=1}^{p} a_i = p+k$; and

(b) there exists a subgraph of $B'_H$ isomorphic to a bush of dimension $(a_1, \ldots, a_p)$.

Proof. We first prove that the existence of a $k$-mini-hitting set in $H$ implies the existence of a bush in $B'_H$ of dimension satisfying (a) and (b). Let $S_{\text{mini}} = \{w_1, \ldots, w_q\}$ be a $k$-mini-hitting set and let $S_i = \{w_{i1}, \ldots, w_{iq}\}$. We know that $q \leq k$ and $|F[S_{\text{mini}}]| \geq |S_{\text{mini}}| + k$. We define $E_i := F[S_i]\setminus F[S_{i-1}]$ for every $i \geq 2$, and $E_1 := F[S_1]$. Let $E_{s_1}, \ldots, E_{s_r}$ be the subsequence of the sequence $E_1, \ldots, E_q$ consisting only of non-empty sets $E_i$, and let $b_j := |E_{s_j}|$ for each $j \in [r]$. Let $p$ be the least integer from $[r]$ such that $\sum_{j=1}^{p} b_j \geq k+p$.

Observe that for every $j \in [p]$, the vertex $w_{s_j}$ belongs to every hyperedge of $E_{s_j}$. Thus, the bipartite graph $B_H$ contains a star-forest $F$ of dimension $(b_1, \ldots, b_p)$, such that $p \leq k$, $b_j \geq 1$ for all $j \in [p]$, and $c := \sum_{j=1}^{p} b_j \geq p+k$. Moreover, each star in $F$ has a central vertex in $V$. By the
minimality of \( p \), we have \( \sum_{j=1}^{p-1} b_j < p - 1 + k \) and so \( b_p \geq c + 1 - (p + k) \). Thus, the integers \( a_j \) defined as follows are positive: \( a_j := b_j \) for every \( j \in [p-1] \) and \( a_p := b_p - c + (p + k) \). Hence, \( B_H \) contains a star-forest \( F' \) of dimension \((a_1, \ldots, a_p)\), such that each star in \( F' \) has a central vertex in \( V \).

Thus, all central vertices are in \( V \), \( p \leq k \), \( a_i \geq 1 \) for all \( i \in [p] \), and \( \sum_{i=1}^{p} a_i = p + k \), which implies that \( B_H' \) contains, as a subgraph, a bush with dimension \((a_1, \ldots, a_p)\) satisfying the conditions above.

The construction above relating a \( k \)-mini-hitting set of \( H \) with the required bush of \( B_H' \) can be easily reversed in the following sense: the existence of a bush of dimension satisfying (a) and (b) in \( B_H' \) implies the existence of a \( k \)-mini-hitting set in \( H \). Here the triangle ensures that the central vertices are in \( V \). This completes the proof.

Next we describe a fast randomized algorithm for deciding the existence of a \( k \)-mini-hitting set using the characterization obtained in Lemma 15. Towards this we will use a fast randomized algorithm for the \textsc{Subgraph Isomorphism} problem. In the \textsc{Subgraph Isomorphism} problem we are given two graphs \( F \) and \( G \) on \( k \) and \( n \) vertices, respectively, as an input, and the question is whether there exists a subgraph of \( G \) isomorphic to \( F \). Recall that \( \text{tw}(G) \) denotes the treewidth of a graph \( G \). We will use the following result.

**Theorem 5** (Fomin et al.[9]). Let \( F \) and \( G \) be two graphs on \( q \) and \( n \) vertices respectively and \( \text{tw}(F) \leq t \). Then, there is a randomized algorithm for the \textsc{Subgraph Isomorphism} problem that runs in expected time \( O(2^t(n t)^{t+O(1)}) \).

Let \( \mathcal{P}(s) \) be the set of all unordered partitions of an integer \( s \) into \( \ell \) parts. Nijenhuis and Wilf [25] designed a polynomial delay generation algorithm for partitions of \( \mathcal{P}(s) \). Let \( p(s) \) be the partition function, i.e., the overall number of partitions of \( s \). The asymptotic behavior of \( p(s) \) was first evaluated by Hardy and Ramanujan in the paper in which they develop the famous “circle method.”

**Theorem 6** (Hardy and Ramanujan [15]). We have \( p(s) \sim e^{\sqrt{s}/4}/(4 \sqrt{3}) \), as \( s \to \infty \).

This theorem and the algorithm of Nijenhuis and Wilf [25] imply the following:

**Proposition 1.** There is an algorithm of runtime \( 2^{O(\sqrt{s})} \) for generating all partitions in \( \mathcal{P}(s) \).

Now we are ready to describe and analyze a fast randomized algorithm for deciding the existence of a \( k \)-mini-hitting set in a hypergraph \( H \). By Lemma 15, it suffices to design and analyze a fast randomized algorithm for deciding the existence of a bush in \( B_H' \) of dimension \((a_1, \ldots, a_p)\) satisfying conditions (a) and (b) of Lemma 15. Our algorithm starts by building \( B_H' \). Then it considers all possible values of \( p \) one by one \( (p \in [k]) \) and generates all partitions in \( \mathcal{P}_p(p+k) \) using the algorithm of Proposition 1. For each such partition \((a_1, \ldots, a_p)\) that satisfies conditions (a) and (b) of Lemma 15, the algorithm of Fomin et al.[9] mentioned in Theorem 5 decides whether \( B_H' \) contains a bush of dimension \((a_1, \ldots, a_p)\). If such a bush exists, we output \text{Yes} and we output \text{No}, otherwise.

To evaluate the runtime of our algorithm, observe that the treewidth of any bush is 2 and any bush in Lemma 15 has at most \(3k+3\) vertices. This observation, the algorithm above, Theorem 5 and Proposition 1 imply the following:

**Theorem 7.** There exists a randomized algorithm solving \((m-k)\)-Hitting Set in expected time \( O(8^{k+O(\sqrt{k})}(m+n)^{O(1)}) \).

This theorem, in turn, implies the following:

**Theorem 8.** There exists a randomized algorithm solving a transformed special instance of \((\nu(F)+k)\)-SAT in expected time \( O(8^{k+O(\sqrt{k})}(m+n)^{O(1)}) \).
6 Complete Algorithm, Correctness and Analysis

The complete algorithm for an instance \((F, \alpha)\) of \((\nu(F) + k)\)-SAT is as follows.

Find a maximum matching \(M\) on \(B_F\) and let \(k = \alpha - |M|\). If \(k \leq 0\), return Yes. Otherwise, apply Reduction Rules 1 to 4 whichever is applicable, in that order and then run the algorithm on the reduced instance and return the answer. If none of the Reduction Rules apply, then apply Branching Rule 1 if possible, to get two instances \((F', \alpha')\) and \((F'', \alpha'')\). Run the algorithm on both instances; if one of them returns Yes, return Yes, otherwise return No. If Branching Rule 1 does not apply then we rearrange the formula and attempt to apply Branching Rule 2 in the same way. Finally if \(k > 0\) and none of the reduction or branching rules apply, then we have for all variables \(x, n(x) = 1\) and every clause contains at most one positive literal, i.e. \((F, \alpha)\) is a special instance. Then solve the problem by first obtaining the transformed special instance, then the corresponding instance \(H^*\) of \((m - k)\)-Hitting Set and solving \(H^*\) in time \(O((2e)^{2k + O(\log^2 k)}(m + n)^{O(1)})\) as described in Sections 4 and 5.

Correctness of all the preprocessing rules and the branching rules follows from Lemmata 2, 3, 4, 6, 9 and 10.

Analysis of the algorithm. Let \((F, \alpha)\) be the input instance. Let \(\mu(F) = \mu = \alpha - \nu(F)\) be the measure. We will first show that our preprocessing rules do not increase this measure. Following this, we will prove a lower bound on the decrease in the measure occurring as a result of the branching, thus allowing us to bound the running time of the algorithm in terms of the measure \(\mu\). For each case, we let \((F', \alpha')\) be the instance resulting by the application of the rule or branch. Also let \(M'\) be a maximum matching of \(B_{F'}\).

Reduction Rule 1. We consider the case when \(n(x) = 0\); the other case when \(n(\bar{x}) = 0\) is analogous. We know that \(\alpha' = \alpha - n(\bar{x})\) and \(\nu(F') \geq \nu(F) - n(\bar{x})\) as removing \(n(\bar{x})\) clauses can only decrease the matching size by \(n(\bar{x})\). This implies that \(\mu(F) - \mu(F') = \alpha - \nu(F) - \alpha' + \nu(F') = (\alpha - \alpha') + (\nu(F') - \nu(F)) \geq n(\bar{x}) - n(\bar{x})\). Thus, \(\mu(F') \leq \mu(F)\).

Reduction Rule 2. We know that \(\alpha' = \alpha - 1\). We show that \(\nu(F') \geq \nu(F) - 1\). In this case we remove the clauses \(c'\) and \(c''\) and add \(c^* = (c' - x) \cup (c'' - \bar{x})\). We can obtain a matching of size \(\nu(F) - 1\) in \(B_{F'}\) as follows. If at most one of the \(c'\) and \(c''\) is the end-point of some matching edge in \(M\) then removing that edge gives a matching of size \(\nu(F) - 1\) for \(B_{F'}\). So let us assume that some edges \((a, c')\) and \((b, c'')\) are in \(M\). Clearly, either \(a \neq x\) or \(b \neq \bar{x}\). Assume \(a \neq x\). Then \(M \setminus \{(a, c'), (b, c'')\} \cup \{(a, c^*)\}\) is a matching of size \(\nu(F) - 1\) in \(B_{F'}\). Thus, we conclude that \(\mu(F') \leq \mu(F)\).

Reduction Rule 3. The proof is the same as in the case of Reduction Rule 1.

Reduction Rule 4. The proof that \(\mu(F') \leq \mu(F)\) in the case when \(F[S]\) is satisfiable is the same as in the case of Reduction Rule 4 in the case when \(F[S]\) is not satisfiable is the same as in the case of Reduction Rule 2.

Branching Rule 1. Consider the case when we set \(x = \text{true}\). In this case, \(\alpha' = \alpha - n(x)\). Also, since no reduction rules are applicable we have that \(F\) is 2-expanding. Hence, \(\nu(F) = |V(F)|\). We will show that in \((F', \alpha')\) the matching size will remain at least \(\nu(F'') - n(x) + 1\) \((\leq |V(F')| - n(x) + 1 = |V(F')| - n(x) + 2)\). This will imply that \(\mu(F') \leq \mu(F) - 1\). By Lemma 1 and the fact that \(n(x) - 2 \geq 0\), it suffices to show that in \(B' = B_{F'}\), every subset \(S \subseteq V(F'), |N_{B'}(S)| \geq |S| - (n(x) - 2)\). The only clauses that have been removed by the simplification process after setting \(x = \text{true}\) are those where \(x\) appears positively and the singleton clauses \((\bar{x})\). Hence, the only edges of \(G[S \cup N_{B'}(S)]\) that are missing in \(N_B(S)\) from \(N_B(S)\) are 12
those corresponding to clauses that contain \( x \) as a pure literal and some variable in \( S \). Thus, \(|N^{B'}(S)| \geq |S| + 2 - n(x) = |S| - (n(x) - 2)\) (as \( F \) is 2-expanding).

The case when we set \( x = \text{false} \) is similar to the case when we set \( x = \text{true} \). Here, also we can show that \( \mu(F') \leq \mu(F) - 1 \). Thus, we get two instances, with each instance \((F', \alpha')\) having \( \mu(F') \leq \mu(F) - 1 \).

**Branching Rule 2.** The analysis here is the same as for Branching Rule 1 and again we get two instances with \( \mu(F') \leq \mu(F) - 1 \).

We therefore have a depth-bounded search tree of size of depth at most \( \mu = \alpha - \nu(F) = k \), in which any branching splits an instance into two instances. Thus, the search tree has at most \( 2^k \) instances. As each reduction and branching rule takes polynomial time, every rule decreases the number of variables, the number of clauses, or the value of \( \mu \), and an instance to which none of the rules apply can be solved in time \( O((2e)^2\mu\mu^O(\log \mu)(m + n)^O(1)) \) (by Theorem 4), we have by induction that any instance can be solved in time

\[
O(2 \cdot (2e)^O((\mu-1)^O(\log(\mu-1)))(m + n)^O(1)) = O((2e)^2\mu\mu^O(\log \mu)(m + n)^O(1)).
\]

Thus the total running time of the algorithm is at most \( O((2e)^{2k+O(\log^2 k)}(n + m)^O(1)) \). Applying Theorem 5 instead of Theorem 4 we conclude that \((\nu(F) + k)\text{-SAT}\) can be solved in expected time \( O(8^{k+O(\sqrt{k})}(n + m)^O(1)) \). Summarizing, we have the following:

**Theorem 9.** There are algorithms solving \((\nu(F) + k)\text{-SAT}\) in time

\[
O((2e)^{2k+O(\log^2 k)}(n + m)^O(1)) \text{ or expected time } O(8^{k+O(\sqrt{k})}(n + m)^O(1)).
\]

### 7 Hardness of Kernelization

In this section, we show that \((\nu(F) + k)\text{-SAT}\) does not have a polynomial-size kernel, unless \( \text{coNP} \subseteq \text{NP/poly} \). To do this, we use the concept of a polynomial parameter transformation. Let \( L \) and \( Q \) be parameterized problems. We say a polynomial time computable function \( f : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N} \) is a polynomial parameter transformation from \( L \) to \( Q \) if there exists a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) such that for any \((x,k) \in \Sigma^* \times \mathbb{N}, (x,k) \in L \) if and only if \( f(x,k) = (x',k') \in Q \), and \( k' \leq p(k) \).

**Lemma 16.** [3, Theorem 3] Let \( L \) and \( Q \) be parameterized problems, and suppose that \( L^e \) and \( Q^e \) are the derived classical problems.\footnote{The parameters of \( L \) and \( Q \) are no longer parameters in \( L^e \) and \( Q^e \); they are part of input.} Suppose that \( L^e \) is \text{NP}-complete, and \( Q^e \in \text{NP} \). Suppose that \( f \) is a polynomial parameter transformation from \( L \) to \( Q \). Then, if \( Q \) has a polynomial-size kernel, then \( L \) has a polynomial-size kernel.

The proof of the next theorem is similar to the proof of Lemma 12.

**Theorem 10.** \((\nu(F) + k)\text{-SAT}\) has no polynomial-size kernel, unless \( \text{coNP} \subseteq \text{NP/poly} \).

**Proof.** By [13, Theorem 3], there is no polynomial-size kernel for the problem of deciding whether a hypergraph \( H \) has a hitting set of size \(|E(H)| - k\), where \( k \) is the parameter unless \( \text{coNP} \subseteq \text{NP/poly} \). We prove the theorem by a polynomial parameter reduction from this problem. Then the theorem follows from Lemma 13 as \((\nu(F) + k)\text{-SAT}\) is \text{NP}-complete.

Given a hypergraph \( H \) on \( n \) vertices, construct a CNF formula \( F \) as follows. Let the variables of \( F \) be the vertices of \( H \). For each variable \( x \), let the unit clause \( (x) \) be a clause in \( F \). For every edge \( \epsilon \) in \( H \), let \( c_{\epsilon} \) be the clause containing the literal \( \bar{x} \) for every \( x \in E \). Observe that \( F \) is matched, and that \( H \) has a hitting set of size \(|E(H)| - k\) if and only if sat\((F) \geq n + k \).
8 Conclusion

We have shown that for any CNF formula $F$, it is fixed-parameter tractable to decide if $F$ has a satisfiable subformula containing $\alpha$ clauses, where $\alpha = \nu(F)$ is the parameter. Our result implies fixed-parameter tractability for the problem of deciding satisfiability of $F$ when $F$ is variable-matched and $\delta(F) \leq k$, where $k$ is the parameter. In addition, we show that the problem does not have a polynomial-size kernel unless coNP $\subseteq$ NP/poly.

Clearly, parameterizations of MaxSAT above $m/2$ and $\nu(F)$ are “stronger” than the standard parameterization (i.e., when the parameter is the size of the solution). Whilst the two non-standard parameterizations have smaller parameter than the standard one, they are incomparable to each other as for some formulas $F$, $m/2 < \nu(F)$ (e.g., for variable-matched formulas with $m < 2n$) and for some formulas $F$, $m/2 > \nu(F)$ (e.g., when $m > 2n$). Recall that Mahajan and Raman [21] proved that MaxSAT parameterized above $m/2$ is fixed-parameter tractable. This result and our main result imply that MaxSAT parameterized above max{$m/2, \nu(F)$} is fixed-parameter tractable: if $m/2 > \nu(F)$ then apply the algorithm of [21], otherwise apply our algorithm.

If every clause of a formula with $m$ clauses contains exactly two literals then it is well known that we can satisfy at least $3m/4$ clauses. From this, and by applying Reduction Rules 1 and 2, we can get a linear kernel for this version of the $(\nu(F) + k)$-SAT problem. It would be nice to see whether a linear or a polynomial-size kernel exists for the $(\nu(F) + k)$-SAT problem if every clause has exactly $r$ literals.

Acknowledgment This research was partially supported by an International Joint grant of the Royal Society.

References

[1] R. Aharoni and N. Linial, Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. J. Combin. Th. Ser. A, 43: 196–204, 1986.

[2] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, Solving MAX-$r$-SAT above a tight lower bound. Algorithmica 61 (2011), 638–655.

[3] N. Alon, R. Yuster, and U. Zwick. Color-coding. J. ACM, 42(4):844–856, 1995.

[4] H.L. Bodlaender, R.G. Downey, M.R Fellows, and D. Hermelin, On problems without polynomial kernels, J. Comput. System Sci., 75(8): 423–434, 2009.

[5] H.L. Bodlaender, S. Thomasse, and A. Yeo, Kernel bounds for disjoint cycles and disjoint paths, Theor. Comput. Sci. 412(35): 4570–4578, 2011.

[6] R. Crowston, G. Gutin, M. Jones, and A. Yeo, A new lower bound on the maximum number of satisfied clauses in Max-SAT and its algorithmic applications. Algorithmica 64 (2012), 56–68.

[7] M. Dom, D. Lokshhtanov and S. Saurabh, Incompressibility though Colors and IDs, Proc. 36th ICALP, Part I, Lect. Notes Comput. Sci. 5555: 378–389, 2009.

[8] R. G. Downey and M. R. Fellows. Parameterized Complexity, Springer, 1999.

[9] F. V. Fomin, D. Lokshhtanov, V. Raman, S. Saurabh and B. V. R. Rao. Faster algorithms for finding and counting subgraphs. J. Comput. Syst. Sci., 78(3):698–706, 2012.
[10] H. Fleischner, O. Kullmann, and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theor. Comput. Sci.*, 289(1):503–516, 2002.

[11] H. Fleischner and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(49), 2000.

[12] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer Verlag, 2006.

[13] G. Gutin, M. Jones, and A. Yeo. Kernels for below-upper-bound parameterizations of the hitting set and directed dominating set problems. *Theor. Comput. Sci.*, 412(41):5744–5751, 2011.

[14] G. Gutin, M. Jones and A. Yeo, A New Bound for 3-Satisfiable MaxSat and its Algorithmic Application. *Proc. FCT 2011*, Lect. Notes Comput. Sci. 6914 (2011), 138–147.

[15] G. H. Hardy and S. Ramanujan. Asymptotic formulae in combinatorial analysis. *Proc. London Math. Soc.*, 17:75–115, 1918.

[16] H. Kleine Büning. On subclasses of minimal unsatisfiable formulas. *Discr. Appl. Math.*, 107(1-3):83–98, 2000.

[17] H. Kleine Bünning and O. Kullmann, Minimal Unsatisfiability and Autarkies, Handbook of Satisfiability, chapter 11, 339–401.

[18] O. Kullmann. An application of matroid theory to the sat problem. In *IEEE Conference on Computational Complexity*, pages 116–124, 2000.

[19] O. Kullmann, Lean clause-sets: Generalizations of minimally unsatisfiable clause-sets, *Discr. Appl. Math.*,130:209-249, 2003.

[20] L. Lovász and M. D. Plummer. *Matching theory*. AMS Chelsea Publ., 2009.

[21] M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. J. Algorithms 31(2) (1999), 335–354.

[22] M. Mahajan, V. Raman, and S. Sikdar. Parameterizing above or below guaranteed values. J. Comput. Sys. Sci. 75(2):137–153, 2009. Preliminary version in the 2nd IWPEC, *Lect. Notes Comput. Sci.* 4169 (2006), 38–49.

[23] B. Monien and E. Speckenmeyer. Solving satisfiability in less than $2^n$ steps. *Discr. Appl. Math*. 10:287–295, 1985.

[24] R. Niedermeier. Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.

[25] A. Nijenhuis and H. S. Wilf. *Combinatorial Algorithms*. Academic Press, Inc. 1978.

[26] C. H. Papadimitriou and D. Wolfe. The complexity of facets resolved. *J. Comput. Syst. Sci.*, 37(1):2–13, 1988.

[27] A. Srinivasan. Improved approximations of packing and covering problems. In *STOC’95*, pages 268–276, 1995.

[28] S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. *J. Comput. Syst. Sci.*, 69(4):656–674, 2004.