Liang HY, He J. Satisfiability with index dependency. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(4): 668–677 July 2012. DOI 10.1007/s11390-012-1253-9

Satisfiability with Index Dependency

Hong-Yu Liang (梁宏宇) and Jing He (何　晶)

Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China

E-mail: {lianghy08, he-j08}@mails.tsinghua.edu.cn

Received March 7, 2011; revised March 18, 2012.

Abstract We study the Boolean satisfiability problem (SAT) restricted on input formulas for which there are linear arithmetic constraints imposed on the indices of variables occurring in the same clause. This can be seen as a structural counterpart of Schaefer's dichotomy theorem which studies the SAT problem with additional constraints on the assigned values of variables in the same clause. More precisely, let k-SAT$(m, \mathcal{A})$ denote the SAT problem restricted on instances of k-CNF formulas, in every clause of which the indices of the last $k-m$ variables are totally decided by the first m ones through some linear equations chosen from $\mathcal{A}$. For example, if $\mathcal{A}$ contains $i_3 = i_1 + 2i_2$ and $i_4 = i_2 - i_1 + 1$, then a clause of the input to 4-SAT$(2, \mathcal{A})$ has the form $y_{i_1} \vee y_{i_2} \vee y_{i_1+2i_2} \vee y_{i_2-i_1+1}$, with y_i being x_i or $\overline{x_i}$. We obtain the following results: 1) If $m \geqslant 2$, then for any set $\mathcal{A}$ of linear constraints, the restricted problem k-SAT$(m, \mathcal{A})$ is either in P or NP-complete assuming P $\neq$ NP. Moreover, the corresponding #SAT problem is always #P-complete, and the MAX-SAT problem does not allow a polynomial time approximation scheme assuming P $\neq$ NP. 2) $m = 1$, that is, in every clause only one index can be chosen freely. In this case, we develop a general framework together with some techniques for designing polynomial-time algorithms for the restricted SAT problems. Using these, we prove that for any $\mathcal{A}$, #2-SAT$(1, \mathcal{A})$ and MAX-2-SAT$(1, \mathcal{A})$ are both polynomial-time solvable, which is in sharp contrast with the hardness results of general #2-SAT and MAX-2-SAT. For fixed $k \geqslant 3$, we obtain a large class of non-trivial constraints $\mathcal{A}$, under which the problems k-SAT$(1, \mathcal{A})$, #k-SAT$(1, \mathcal{A})$ and MAX-k-SAT$(1, \mathcal{A})$ can all be solved in polynomial time or quasi-polynomial time.

Keywords Boolean satisfiability problem, index-dependency, index-width, dichotomy

1 Introduction

The Boolean satisfiability problem (SAT) has received extensive studies since it was first proposed as a natural NP-complete problem in [1]. It is also well known that 3-SAT is still NP-complete[1] while 2-SAT is linear-time tractable[2], where by k-SAT we mean the problem of deciding whether a given CNF formula with exactly k literals in each clause is satisfiable. The counting class #P is introduced by Valiant[3], and he proved in [4] that #SAT, the problem of counting the number of satisfying assignments of a given formula, is #P-complete even on instances of monotone 2-CNFs. The optimization version MAX-3SAT, and even MAX-2SAT, have been shown to be AXP-hard[5-6].

People thus became interested in examining the complexity of certain restricted versions of SAT, such as Horn-SAT[7-8] and 1-in-3-SAT[9], where the former is polynomial-time decidable while the latter is NP-complete. The most significant step in this line of research is Schaefer's dichotomy theorem[9], which we will briefly review as follows. In most cases, the restrictions associated with the SAT problem can be seen as a set of logical relations $S = \{R_1, \ldots, R_m\}$, where for all $i \in \{1, \ldots, m\}$, $R_i \subseteq \{0,1\}^k$ for some integer $k \geqslant 1$. An R-clause, written as $R(x_1, \ldots, x_k)$, is said to be satisfied iff $(b_1, \ldots, b_k) \in R$ where b_i is the assigned value to the variable x_i. Any R-clause with $R \in S$ is called an S-clause. The problem SAT(S) is then defined to be the problem of deciding whether a given set of S-clauses are simultaneously satisfiable. It is easy to see that many variants of SAT, including the canonical k-SAT, Horn SAT and 1-in-3 SAT, fall into this classification. Schaefer's dichotomy theorem states that for every such S, SAT(S) is polynomial-time decidable or NP-complete. Allender *et al.*[10] proposed a refinement of Schaefer's theorem considering AC^0 isomorphisms instead of polynomial-time reductions.

Roughly speaking, Schaefer's dichotomy theorem accounts for the variants of SAT problems for which certain constraints are imposed on the *assigned values of variables appearing in the same clause*, while it does

Regular Paper

This work was supported in part by the National Basic Research 973 Program of China under Grant Nos. 2011CBA00300, 2011CBA00301, and the National Natural Science Foundation of China under Grant Nos. 61033001, 61061130540, 61073174.

A preliminary version of this paper appeared in Proceedings of ISAAC 2010.

not directly consider the *structure of variables in the input formula* itself. For illustration, the fact that 3-SAT remains NP-complete on instances in which every variable occurs at most four times[11], cannot be derived from Schaefer's theorem. An interesting point is that 3-SAT with each variable occurring at most three times is always satisfiable[11]. This "phase transition" phenomenon is generalized to k-SAT by Kratochvíl, Savický and Tuza[12] (see [13] for a recent result for determining this threshold). Other examples include the NP-completeness of the Planar-3-SAT problem[14], which is to decide whether a given 3-CNF formula with a planar incidence graph is satisfiable.

Our Model. In this paper, we initiate the study of SAT problems restricted on formulas for which there are some pre-known constraints on the indices of variables occurring in the same clause. This type of "index-dependency constraints" can be seen as a structural counterpart of what is considered by Schaefer[9]. We focus on linear arithmetic constraints, since they are the most natural ones we may encounter in various scenes.

Besides its pure theoretical interest as being a comparison to Schaefer's results, another motivation for studying this type of constraints on the clauses comes from the generation of random CNF formulas for testing the behavior of algorithms or other purposes. In practice, we usually use $O(km \log n)$ random bits, obtained by a pseudorandom number generator, to produce a random k-CNF formula of n variables and m clauses. A dilemma is that, when the number of formulas wanted is large, we cannot hope to get "good" random formulas if the pseudorandom number generator is too simple, while it costs much more time using a complex (but close to "true randomness") generator. A natural question is that to what extent we can relax the requirement on the quality of randomness and still get good (or, hard) enough formulas. Can we still get hard instances of SAT if the pseudorandom number generator is based on simple linear arithmetic? Can we get hard formulas by first using only a few bits obtained from a complex generator, and then performing some simple deterministic steps? One of our results provides evidence to a positive answer of these questions, which essentially says that in a k-CNF formula only the indices of the first two (or arbitrarily two, since the order does not matter) variables in any clause are "important", and others can just be generated by simple linear arithmetic operations (Theorem 5). It is thus interesting to study the complexity of these types of restricted SAT problems.

We now briefly explain the model we use. More rigorous definitions can be found in Section 2. Let $f = \bigwedge_{i=1}^{t} \bigvee_{j=1}^{k} y_{a_{i,j}}$ be a k-CNF formula, where y_l denotes either x_l or $\overline{x_l}$. A simple way for demonstrating the linear constraints on indices is to introduce a matrix $\boldsymbol{\mathcal{A}}$ and a vector $\boldsymbol{b}$, and requiring that $\boldsymbol{\mathcal{A}}(a_{i,1}, \ldots, a_{i,k})^{\mathrm{T}} = \boldsymbol{b}$ for all $i \in \{1, \ldots, t\}$. But for convenience of our proof, and without loss of generality, we adopt a more strict requirement that in any clause, the first m indices can be set freely, while the last $k - m$ indices are totally decidable by the first m ones through some linear equations. (Note that it is equivalent to let the last m indices to be free, or any collection of m indices.) More precisely, there exists an integer m and a matrix $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-m)\times(m+1)}$ such that $\forall i \in \{1, 2, \ldots, t\}$, $(a_{i,m+1}, \ldots, a_{i,k})^{\mathrm{T}} = \boldsymbol{\mathcal{A}}(a_{i,1}, \ldots, a_{i,m}, 1)^{\mathrm{T}}$. For instance, when $\boldsymbol{\mathcal{A}} = \begin{pmatrix} 1 & 1 & 0 \\ 2 & -1 & -3 \end{pmatrix}$, every clause has the form $(y_i \vee y_j \vee y_{i+j} \vee y_{2i-j-3})$, with y_l denoting x_l or $\overline{x_l}$. Since subtraction is allowed to define an arithmetic constraint, we assume that the variables can be indexed by any integer, not necessarily positive.

We refer to the problem of deciding whether a given k-CNF formula of this form is satisfiable as D-k-SAT$(m, \boldsymbol{\mathcal{A}})$. When $m = k$ it degenerates to the classical k-SAT problem. Other variants like Max-k-SAT$(m, \mathcal{A})$ and #k-SAT$(m, \mathcal{A})$ can be defined similarly.

Our Contributions. We systematically study the complexity of SAT problems on instances with (linear) index-dependency between variables in each clause.

• In the first part of this article (Section 3), we prove a dichotomy theorem for the case $m \geqslant 2$. More precisely, for any $k \geqslant m \geqslant 2$ and $\mathcal{A} \in \mathbb{Z}^{(k-m)\times(m+1)}$, D-$k$-SAT$(m, \mathcal{A})$ is either in P or NP-complete, assuming P $\neq$ NP. (Recall that m is the number of free variables in each clause and k is the number of literals in each clause.) We also prove that for any $k \geqslant m \geqslant 2$ and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-m)\times(m+1)}$, #$k$-SAT$(m, \boldsymbol{\mathcal{A}})$ is #P-complete and Max-k-SAT$(m, \boldsymbol{\mathcal{A}})$ does not have a polynomial time approximation scheme, unless P = NP.

• The second part (Section 4) of this paper is devoted to the case $m = 1$. We first show that for any $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 2}$, WMax-2-SAT$(1, \boldsymbol{\mathcal{A}})$ (the weighted version of MAX-SAT) and #2-SAT$(1, \boldsymbol{\mathcal{A}})$ are both polynomial-time solvable, which is in sharp contrast with the hardness results of general #2-SAT[4] and Max-2SAT[6]. We then consider the case where $m = 1$ and $k = 3$, showing that there is a large (and surely non-trivial) class of $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{2\times 2}$ for which both WMax-k-SAT$(1, \boldsymbol{\mathcal{A}})$ and #k-SAT$(1, \boldsymbol{\mathcal{A}})$ can be solved in polynomial time. Finally we generalize the results to the case $k > 3$, showing that a large and natural class of $\boldsymbol{\mathcal{A}}$ makes the problems solvable in DTIME$(n^{\mathrm{polylog}(n)})$, which is generally believed not to contain any NP-hard problem.

Novelty of Techniques. To deal with the case $m = 1$, we developed a general framework for designing polynomial-time algorithms for the SAT problems restricted on this case. First, we define a new parameter on the formulas, which we call the index-width (see Section 2 for the definition). Roughly speaking, it captures how "far away" two variables in the same clause can be from each other, under some ordering of the variables. We compare it to the existing definition of bandwidth[15] (also referred to as the diameter in [16]). We show that small index-width implies small bandwidth if the considered formula has large "index-dependency" (or, in our notation, $m = 1$). Moreover, a "layout" of the formula achieving a good (small) bandwidth can be constructed efficiently if its index-width is small. (In fact, we can further prove that the two measures have the same magnitude up to some constant factor when $m = 1$.) This guarantees that the algorithm in [16] for solving variants of SAT, which takes the bandwidth as a parameter, runs in polynomial time. With a slight modification, we can apply it to the weighted maximization version as well.

The reason that we define a new width parameter, but not directly use bandwidth, is that this new concept is easier to handle. To get an intuitive idea, the bandwidth cares a lot about the clauses of a formula, while the index-width deals mainly with the variables themselves. Although two arbitrary formulas on n Boolean variables may have very different structures of clauses, their variables can, without loss of generality, be regarded as the same set $\{x_1, x_2, \ldots, x_n\}$. By arguing on the index-width, we are able to handle a large class of formulas at the same time. Another reason is that this definition is well related to the index-dependency, since they both consider the indices, or more generally, the integers. We are then able to use tools from number theory and combinatorics to obtain desired results. We also believe that, besides the obtained results, the concept of index-width and the techniques used are of their own interest.

2 Preliminaries

In this paper $[a, b]$ denotes the integer set $\{\lceil a \rceil, \lceil a \rceil + 1, \ldots, \lfloor b \rfloor\}$. We use $\log x$ to denote the logarithm base 2 of x. $\mathbb{Z}$ is the set of all integers, $\mathbb{Z}^+$ is the set of all positive integers and $\mathbb{Z}^{a\times b}$ is the set of all $a \times b$ integer matrices. Let $\mathcal{A}[i, j]$ denote the element of a matrix $\boldsymbol{\mathcal{A}}$ at the crossing of the i-th row and the j-th column.

Let $X = \{x_i \mid i \in \mathbb{Z}\}$ be an infinite set of Boolean variables indexed by all integers. Without loss of generality, all Boolean variables considered in this paper are chosen from X. For all $i \in \mathbb{Z}$, we write y_i to represent a literal chosen from $\{x_i, \overline{x_i}\}$. Define an index function $\mathcal{I}$ as $\mathcal{I}(y_i) = i$ for all $i \in \mathbb{Z}$.

A clause is the disjunction of literals, which can be seen as an ordered tuple. A CNF formula (or simply CNF) is the conjunction of clauses, and a k-CNF is a CNF in which all clauses have exactly k literals. For a clause c, let $\mathcal{I}(c) = \{\mathcal{I}(x) \mid x \in c\}$ be the set of indices occurring in c, and $\mathcal{I}(c, i)$ be the index of the i-th literal of c. For a CNF f, let $\mathcal{I}(f) = \bigcup_{c\in f} \mathcal{I}(c)$. Define the index-width of f as:

$$\mathcal{W}(f) = \max_{c\in f} \max_{i,j\in\mathcal{I}(c)} |i - j|.$$

Observe that the index-width of f is just the maximum difference between any two indices in the same clause of f. This concept is a dual analog of the bandwidth of a formula[15], whose definition will be restated below in a more helpful form. Given a CNF formula $f = \bigwedge_{i=1}^{m} c_i$, a layout of f is an injective function $l : \{c_1, \ldots, c_m\} \to \mathbb{Z}$ (or, a relabeling of the clauses). The bandwidth of f under layout l is defined as the minimum integer k s.t. whenever a variable or its negation appears in two clauses c_i and c_j, the inequality $|l(c_i) - l(c_j)| \leqslant k$ holds. The bandwidth of f is the minimum bandwidth of f under all possible layouts.

Definition 1. *Let $k \geqslant m \geqslant 1$ be two positive integers and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-m)\times(m+1)}$. Define k-CNF$(m, \boldsymbol{\mathcal{A}})$ to be the collection of all k-CNF f such that for any clause $c \in f$, it holds that $(\mathcal{I}(c, m + 1), \ldots, \mathcal{I}(c, k))^{\mathrm{T}} = \boldsymbol{\mathcal{A}}(\mathcal{I}(c, 1), \ldots, \mathcal{I}(c, m), 1)^{\mathrm{T}}$. With a little abuse of notation, we also use k-CNF$(m, \boldsymbol{\mathcal{A}})$ to denote a formula chosen from this set. (For example, any clause of a formula $f \in 4\text{-CNF}\left(2, \begin{pmatrix} 1 & 1 & 0 \\ 2 & -1 & -3 \end{pmatrix}\right)$ has the form $y_i \vee y_j \vee y_{i+j} \vee y_{2i-j-3}$.)*

Definition 2. *Let $k \geqslant 2, m \in [1, k]$ and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-m)\times(m+1)}$. An instance of k-SAT$(m, \boldsymbol{\mathcal{A}})$ consists of a string 1^n as well as a formula $f \in k$-CNF$(m, \boldsymbol{\mathcal{A}})$ with $\mathcal{I}(f) \subseteq [-n, n]$. The goal is to find an assignment to $\{x_i \mid i \in [-n, n]\}$ that satisfies f, or correctly report that no such assignment exists. Define* D-k-CNF$(m, \boldsymbol{\mathcal{A}})$, MAX-$k$-CNF$(m, \boldsymbol{\mathcal{A}})$, WMAX-$k$-CNF$(m, \boldsymbol{\mathcal{A}})$ *and* #k-CNF$(m, \boldsymbol{\mathcal{A}})$ *to be the decision version, the optimization version, the weighted optimization version and the counting version of k-SAT$(m, \boldsymbol{\mathcal{A}})$, respectively. (Thus in our notation, D-k-SAT$(k, \varnothing)$, MAX-k-SAT$(k, \varnothing)$ and #k-SAT$(k, \varnothing)$ correspond to the canonical problems k-SAT, MAX-k-SAT and #k-SAT, respectively.)*

3 Dichotomy Results for k-SAT$(m, \mathcal{A})$ when $m \geqslant 2$

3.1 Dichotomy Theorem for 3-SAT(2, $\mathcal{A}$)

We examine the complexity of variants of 3-SAT(2,

$\boldsymbol{\mathcal{A}}$). In this case $\boldsymbol{\mathcal{A}}$ is just a 3-D integer vector (a, b, c), and every clause of an instance has the form $y_i \vee y_j \vee y_{ai+bj+c}$. It is easy to find some special vectors $\boldsymbol{\mathcal{A}}$ for which 3-SAT(2, $\boldsymbol{\mathcal{A}}$) degenerates to 2-SAT: When $\boldsymbol{\mathcal{A}} = (0, 1, 0)$ or $(1, 0, 0)$ it is obvious, and when $\boldsymbol{\mathcal{A}} = (0, 0, c)$ it suffices to examine the two 2-SAT instances where x_c is set to 0 and 1 respectively. Therefore 3-SAT(2, $\boldsymbol{\mathcal{A}}$) can be solved in polynomial time under these choices of $\boldsymbol{\mathcal{A}}$.

It is somehow surprising that they are the only possibilities that 3-SAT(2, $\boldsymbol{\mathcal{A}}$) is polynomial-time solvable, assuming $P \neq$ NP. For convenience we define $\mathit{Easy}\mathcal{A} = \{(0, 1, 0), (1, 0, 0)\} \cup \{(0, 0, c) \mid c \in \mathbb{Z}\}$ which consists of all easy cases.

Theorem 1. *Let $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 3}$. Then D-3-SAT(2, $\boldsymbol{\mathcal{A}}$) is in P if $\boldsymbol{\mathcal{A}} \in \mathit{Easy}\mathcal{A}$, and is NP-complete otherwise.*

We also obtain the following inapproximability results of the corresponding optimization problems, whose proof is shown below. The NP-completeness part of Theorem 1 is proved in a totally similar manner.

Theorem 2. *Let $\epsilon > 0$ be any positive constant. Then, for $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 3}$, it is NP-hard to approximate* Max-3-*SAT*(2, $\boldsymbol{\mathcal{A}}$) *better than*

- $21/22 + \epsilon$ *if* $\boldsymbol{\mathcal{A}} \in \{(0, 1, 0), (1, 0, 0)\}$;
- $43/44 + \epsilon$ *if* $\boldsymbol{\mathcal{A}} \in \{(0, 0, c) \mid c \in \mathbb{Z}\}$;
- $23/24 + \epsilon$ *if* $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 3} \setminus \mathit{Easy}\boldsymbol{\mathcal{A}}$.

Proof. Since Max-2SAT is $(21/22+\epsilon)$-inapproximable[6], the result is straightforward when $\boldsymbol{\mathcal{A}} \in \{(0, 1, 0), (1, 0, 0)\}$. Now suppose $\boldsymbol{\mathcal{A}} = (0, 0, c)$ for some $c \in \mathbb{Z}$. We reduce Max-2SAT to Max-3-SAT(2, $\boldsymbol{\mathcal{A}}$). Let $f = \bigwedge_{i=1}^{m} c_i$ be an instance of Max-2SAT. For $i \in [1, m]$, let $c'_{i,0} := c_i \vee x_c$ and $c'_{i,1} := c_i \vee \overline{x_c}$. Define a new formula $f' := f'_0 \wedge f'_1$, where $f'_j = \bigwedge_{i=1}^{m} c'_{i,j}$ for $j \in \{0, 1\}$. Noting that f is an instance of Max-3-SAT(2, $\boldsymbol{\mathcal{A}}$), our reduction is finished. Let OPT and OPT' be the maximum number of clauses of f and f' that can be satisfied simultaneously respectively. We show that $OPT' = m + OPT$. Assume $\mathcal{X}$ is an assignment of the variables of f that satisfies OPT clauses of f. Extend $\mathcal{X}$ to an assignment $\mathcal{X}'$ of the variables of f' in the following way: let $x_c = 0$ if $c \notin \mathcal{I}(f)$, and do nothing otherwise. It is easy to see that this assignment satisfies $m + OPT$ clauses of f'. Conversely, let $\mathcal{X}'$ be an assignment that satisfies OPT' clauses of f'. Then, by our construction of f', the partial assignment of $\mathcal{X}'$ restricted on the variables of f satisfies $OPT' - m$ clauses of f. Therefore, we have $OPT' = OPT + m$. By [6] we know that it is NP-hard to distinguish between satisfiable 2-SAT formulas, and 2-SAT formulas in which at most a $21/22 + \epsilon$ fraction of the clauses can be satisfied simultaneously. Thus, it is NP-hard to distinguish between satisfiable 3-SAT(2, $\boldsymbol{\mathcal{A}}$) formulas, and 3-SAT(2, $\boldsymbol{\mathcal{A}}$) formulas in which at most a $(1 + 21/22 + \epsilon)/2 = 43/44 + \epsilon/2$ fraction of the clauses are simultaneously satisfiable. This gives us the desired inapproximability result.

In the following we focus on the rest case $\boldsymbol{\mathcal{A}} = (a, b, c) \notin \mathit{Easy}\mathcal{A}$. We may assume that $b \neq 0$, since otherwise we can exchange the values of a and b to get an equivalent problem. We need the following lemma.

Lemma 1. *Suppose $(a, b, c) \in \mathbb{Z}^{1\times 3} \setminus \mathit{Easy}\mathcal{A}$ where $b \neq 0$. Given m integers $\{i_1, \ldots, i_m\} \subseteq [-n, n]$ with the restriction that $(b = 1 \Rightarrow \forall l \in [1, m] : ai_l + c \neq 0)$, we can find $2m$ pairwise-different integers $j'_1, \ldots, j'_m, k'_1, \ldots, k'_m$ such that for all $l \in [1, m]$:*

1) $n < |j'_l|, |k'_l| \leqslant qmn$ where q only depends on a, b and c;

2) $k'_l = ai_l + bj'_l + c$.

Moreover, these numbers can be found in time polynomial in n and m.

Proof. We provide a case analysis as follows guided by the value of b.

1) $b \geqslant 2$. Let $t = \max\{1, \lceil \frac{a(i_l - i_{l+1})}{b} \rceil + 1 \mid l \in [1, m-1]\}$ and $s = \max\{1, n+1, \lceil \frac{(m-1)t - ai_1 - c}{b-1} \rceil + 1\}$. Define $j'_l = s + (l-1)t$ and $k'_l = ai_l + bj'_l + c$ for all $l \in [1, m]$. It is easy to verify that both conditions required by the lemma hold, and obviously $n < j'_1 < j'_2 < \cdots < j'_m$. In addition, we have $k'_1 - j'_m = (ai_1 + bs + c) - (s + (m-1)t) = (b-1)s + ai_1 + c - (m-1)t > 0$ due to the choice of s, and for all $1 \leqslant l \leqslant m-1$, $k'_{l+1} - k'_l = (ai_{l+1} + b(s + lt) + c) - (ai_l + b(s + (l-1)t) + c) = bt + a(i_{l+1} - i_l) > 0$ due to the choice of t. Thus all $2m$ integers are pairwise different.

2) $b = 1$. Let $j'_l = n + 2l(n|a| + |c| + 1)$ and $k'_l = j'_l + ai_l + c$ for all $l \in [1, m]$. Since $ai_l + c \neq 0$ we have $k'_l \neq j'_l$. Furthermore, for $l \in [1, m-1]$ we have $\min\{j_{l+1}, k_{l+1}\} - \max\{j_l, k_l\} \geqslant 2(n|a| + |c| + 1) - |ai_l + c| - |ai_{l+1} + c| > 0$, due to the fact that $|i_l|, |i_{l+1}| \leqslant n$. Hence these numbers are eligible.

3) $b < 0$. Let $t = \max\{1, \lceil \frac{a(i_l - i_{l+1})}{b} \rceil + 1 \mid l \in [1, m-1]\}$ and $s = \max\{1, n+1, \lceil \frac{-n - ai_1 - c}{b} \rceil + 1\}$. Define $j'_l = s + (l-1)t$ and $k'_l = ai_l + bj'_l + c$ for all $l \in [1, m]$. Similar to Case 1), an elementary calculation shows that $n < j_1 < j_2 < \cdots < j_m$ and $k'_m < k'_{m-1} < \cdots < k'_1 < -n$, and other conditions also hold. This completes the proof of Lemma 1. □

We now continue to prove Theorem 2. We present a polynomial-time reduction from Max-E3LIN2, which is NP-hard to be approximated within a factor of $1/2+\epsilon$ due to Håstad's PCP verifier[6], to Max-3-SAT(2, $\boldsymbol{\mathcal{A}}$). An instance of Max-E3LIN2 is a system of linear equations modulo 2 with 3 distinct variables per equation (e.g., $x_2 \oplus x_4 \oplus x_7 = 1$). The goal is to find an assignment which satisfies the maximum number of equations. Note that the decision version of this problem can be solved easily by Gaussian elimination.

Suppose an instance $\mathcal{E}$ of MAX-E3LIN2 consists of m equations $\{x_{i_l} \oplus x_{j_l} \oplus x_{k_l} = b_l \mid l \in [1,m]\}$, where $i_l, j_l, k_l \in [1,n], i_l \neq j_l \neq k_l$ and $b_l \in \{0,1\}$ (we call the equations $E_1, \ldots, E_m$). If $b = 1$ we can assume that $ai_l + c \neq 0$, since otherwise we can use $x_{j_l} \oplus x_{i_l} \oplus x_{k_l} = b_l$ to replace the original equation (recall that a and c cannot be both 0 in this case). We apply Lemma 1 to get $2m$ pairwise-different integers $\{j'_l, k'_l \mid 1 \leqslant l \leqslant m\}$ such that $k'_l = ai_l + bj'_l + c$. Let $j''_l = aj_l + bj'_l + c$ and $k''_l = ak_l + bk'_l + c$ for $l \in [1,m]$.

For any $l \in [1,m]$, the equation $E_l : x_{i_l} \oplus x_{j_l} \oplus x_{k_l} = b_l$ is implied by a set of three equations: $x_{i_l} \oplus x_{j'_l} \oplus x_{k'_l} = b_l, x_{j_l} \oplus x_{j'_l} = 0$ and $x_{k_l} \oplus x_{k'_l} = 0$, which we will refer to as $E'_{l,1}, E'_{l,2}$ and $E'_{l,3}$ respectively. $E'_{l,1}$ can be expressed as a 4-clause 3-CNF(2, $\boldsymbol{\mathcal{A}}$), since for any x, y and z we have $x \oplus y \oplus z = 0 \Leftrightarrow (\overline{x} \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y \vee z) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z}) = 1$, and $x \oplus y \oplus z = 1 \Leftrightarrow (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z}) = 1$. An interpretation of $E'_{l,2}$ is $(x_{j_l} \vee \overline{x_{j'_l}}) \wedge (\overline{x_{j_l}} \vee x_{j'_l}) = 1$, which is equivalent to: $(x_{j_l} \vee \overline{x_{j'_l}} \vee x_{j''_l}) \wedge (x_{j_l} \vee \overline{x_{j'_l}} \vee \overline{x_{j''_l}}) \wedge (\overline{x_{j_l}} \vee x_{j'_l} \vee x_{j''_l}) \wedge (\overline{x_{j_l}} \vee x_{j'_l} \vee \overline{x_{j''_l}}) = 1$. Similarly, $E'_{l,3}$ is equivalent to $(x_{k_l} \vee \overline{x_{k'_l}} \vee x_{k''_l}) \wedge (x_{k_l} \vee \overline{x_{k'_l}} \vee \overline{x_{k''_l}}) \wedge (\overline{x_{k_l}} \vee x_{k'_l} \vee x_{k''_l}) \wedge (\overline{x_{k_l}} \vee x_{k'_l} \vee \overline{x_{k''_l}}) = 1$. Let $E'_l = E'_{l,1} \wedge E'_{l,2} \wedge E'_{l,3} \in$ 3-CNF(2, $\boldsymbol{\mathcal{A}}$) for any $l \in [1,m]$. Let $f = \bigwedge_{i=1}^{m} E'_l$ and n' be the smallest value such that $V(f) \subseteq [-n', n']$. Our instance of MAX-3-SAT(2, $\boldsymbol{\mathcal{A}}$) contains both f and $1^{n'}$, noting that n' is polynomially bounded by m and n. This finishes the description of the reduction.

In [6] it is proved that for any $\epsilon > 0$, given a set of input equations of MAX-E3LIN2, it is NP-hard to distinguish between the following two cases: 1) at least a $(1-\epsilon)$ fraction of the input equations are simultaneously satisfiable, and 2) any assignment can satisfy at most a $(1/2+\epsilon)$ fraction of the input. We translate these two scenarios in terms of f to which the above reduction maps the MAX-E3LIN2 instance $\mathcal{E} = \{E_l \mid l \in [1,m]\}$. If a $(1-\epsilon)$ fraction of $\mathcal{E}$ are satisfiable by an assignment $\mathcal{X} = \{\hat{x}_i \mid i \in [1,n]\}$, we construct an assignment $\mathcal{X}' = \{x_i \mid i \in [-n', n']\}$ for f as follows: for all $i \in [1,n]$, let $x_i = \hat{x}_i$, and for all $l \in [1,m]$, let $x_{j'_l} = \hat{x}_{j_l}$ and $x_{k'_l} = \hat{x}_{k_l}$. For all unassigned variables x_t we let $x_t = 1$. This is a valid assignment, since all j'_ls and k'_ls are pairwise-distinct integers beyond the range $[1,n]$, and therefore no variable is assigned twice. Moreover, from the construction of f we know that for any $l \in [1,m]$, E_l is satisfied by $\mathcal{X}$ if and only if E'_l, which is a sub-formula of f with 12 clauses, is satisfied by $\mathcal{X}'$. Hence, $\mathcal{X}$' satisfies at least $12(1-\epsilon)m$ clauses of f.

Conversely, if an assignment $\mathcal{X}' = \{x_i \mid i \in [-n', n']\}$ satisfies E'_l, the partial assignment $\mathcal{X} = \{x_i \mid i \in [1,n]\}$ must satisfy E_l. Therefore, in the second case any assignment for f can satisfy at most $12(1/2+\epsilon)m + 11(1/2-\epsilon)m = (23/2+\epsilon)m$ clauses of f. As a consequent, it is NP-hard to approximate MAX-3-SAT(2, $\boldsymbol{\mathcal{A}}$) with a factor of $\frac{(23/2+\epsilon)m}{12(1-\epsilon)m} < 23/24 + 3\epsilon$ for $0 < \epsilon < 1/2$, completing the proof of Theorem 2.

To prove the NP-completeness part of Theorem 1, it suffices to apply a similar reduction from the canonical 3-SAT problem to D-3-SAT(2, $\mathcal{A}$). □

Finally, we have the following hardness result for the counting version.

Theorem 3. *#3-SAT(2, $\boldsymbol{\mathcal{A}}$) is #P-complete for any $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 3}$.*

Proof. Let $sol(f)$ denote the number of satisfying assignments for formula f. When $\boldsymbol{\mathcal{A}} \in Easy\mathcal{A}$ the result follows easily from the #P-completeness of #2SAT. If $\boldsymbol{\mathcal{A}} \notin Easy\mathcal{A}$, for any 3-CNF f we apply a reduction similar to that used in Theorem 2 to obtain $f' \in$ 3-CNF(2, $\boldsymbol{\mathcal{A}}$) such that f and f' are equivalent w.r.t. their satisfiability. Suppose f contains n variables and m clauses, while f' has n' variables. From the proof of Theorem 2 we know that each satisfying assignment for f corresponds to exactly $2^{n'-n-2m}$ satisfying assignments for f' (that is, all variables not in $\{x_i \mid i \in [1,n]\} \cup \{x_{j'_l}, x_{k'_l} \mid l \in [1,m]\}$ can be set freely), hence $sol(f') = 2^{n'-n-2m} sol(f)$. This finishes the reduction from #3SAT to #3-SAT(2, $\boldsymbol{\mathcal{A}}$) and concludes that the latter is #P-complete. □

3.2 Results for $k > 3$ and $m \geqslant 2$

In this subsection we consider the cases where $k \geqslant 4$ and $m \geqslant 2$. The following two theorems are not hard to prove given Theorem 1.

Theorem 4. *Let $k \geqslant m \geqslant 3$ be two integers and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-m)\times(m+1)}$. Then D-k-SAT($m$, $\boldsymbol{\mathcal{A}}$) is NP-complete.*

Proof. We make a reduction from the conventional problem m-SAT to it. Let f be any m-CNF formula. For any clause $c \in f$, say $c = y_{r_1} \vee y_{r_2} \vee \cdots \vee y_{r_m}$, we introduce 2^{k-m} clauses $(c \vee y_{r_{m+1}} \vee \cdots \vee y_{r_k})$ where $(r_{m+1}, \ldots, r_k)^{\mathrm{T}} = \boldsymbol{\mathcal{A}}(r_1, \ldots, r_m, 1)^{\mathrm{T}}$ and y_i denotes x_i or $\overline{x_i}$. It is easy to see that all the new clauses are in k-CNF(m, $\boldsymbol{\mathcal{A}}$) and c is equivalent with the conjunction of them. In this way we obtain an instance of D-k-SAT(m, $\boldsymbol{\mathcal{A}}$) which is equivalent with f, and hence conclude that D-k-SAT(m, $\boldsymbol{\mathcal{A}}$) is NP-complete. □

Theorem 5. *Let $k \geqslant 3$ and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-2)\times 3}$. Assuming $P \neq$ NP, D-k-SAT(2, $\boldsymbol{\mathcal{A}}$) is in P if all row vectors of $\boldsymbol{\mathcal{A}}$ are in Easy$\mathcal{A}$, and is NP-complete otherwise.*

Proof. If all row vectors of $\boldsymbol{\mathcal{A}}$ are in $Easy\mathcal{A}$, the problem degenerates to at most 2^{k-2} 2-SAT cases (the worst case happens when all vectors have the form $(0,0,c)$ and we go over all possible assignments of the fixed $k-2$ variables) and is in P since k is a fixed integer.

Otherwise, without loss of generality we assume that the first row of $\boldsymbol{\mathcal{A}}$ (denoted by $\boldsymbol{\mathcal{A}}_1$) is not in *Easy*$\mathcal{A}$. We provide a reduction from 3-SAT(2, $\boldsymbol{\mathcal{A}}_1$) to k-SAT(2, $\boldsymbol{\mathcal{A}}$). Let $f \in$ 3-CNF(2, $\boldsymbol{\mathcal{A}}_1$). For any clause $c \in f$, say $c = y_{r_1} \vee y_{r_2} \vee y_{r_3}$ where $r_3 = \boldsymbol{\mathcal{A}}_1(r_1, r_2, 1)^{\mathrm{T}}$, we introduce 2^{k-3} clauses $(c \vee y_{r_4} \vee \cdots \vee y_{r_k})$ where $(r_3, r_4, \ldots, r_k)^{\mathrm{T}} = \boldsymbol{\mathcal{A}}(r_1, r_2, 1)^{\mathrm{T}}$. The rest of the proof is similar to that of Theorem 4. □

Combining the above statements and noting that the preceding hardness results also carry over to the case $k > 3$, we obtain:

Corollary 1. *Let $k \geqslant m \geqslant 2$ and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-m)\times(m+1)}$. Then,*

- *D-k-SAT$(m, \boldsymbol{\mathcal{A}})$ is either in P or NP-complete, assuming $P \neq$ NP;*
- *#k-SAT$(m, \boldsymbol{\mathcal{A}})$ is always #P-complete;*
- MAX-*SAT$(m, \boldsymbol{\mathcal{A}})$ does not admit a polynomial time approximation scheme (PTAS) unless $P = NP$.*

4 Tractability Results for $m = 1$

We now turn to the case $m = 1$. A little surprisingly, we will show that for any $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 2}$, all the variants of 2-SAT(1, $\boldsymbol{\mathcal{A}}$) defined previously are polynomial-time solvable. For $k \geqslant 3$, there is a large class of non-trivial choices of $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-1)\times 2}$ for which the variants of k-SAT(1, $\boldsymbol{\mathcal{A}}$) can all be solved in polynomial time or quasi-polynomial time (DTIME($n^{\mathrm{polylog}(n)}$)). We assume without loss of generality that every input formula has pairwise different clauses, since otherwise we can easily detect and remove redundant clauses (and add the weight to another clause when dealing with MAX-SAT). We first define the following subsets of $\mathbb{Z}^{2\times 2}$ (all variables below are assumed to be integers):

$$\begin{aligned}
\mathscr{A}_1 := &\left\{\begin{pmatrix} 0 & b_1 \\ a_2 & b_2 \end{pmatrix}\right\} \bigcup \left\{\begin{pmatrix} a_1 & b_1 \\ 0 & b_2 \end{pmatrix}\right\} \bigcup \\
&\left\{\begin{pmatrix} 1 & 0 \\ a_2 & b_2 \end{pmatrix}\right\} \bigcup \left\{\begin{pmatrix} a_1 & b_1 \\ 1 & 0 \end{pmatrix}\right\} \bigcup \\
&\left\{\begin{pmatrix} a_1 & b_1 \\ a_1 & b_1 \end{pmatrix}\right\}, \\
\mathscr{A}_2 := &\left\{\begin{pmatrix} 1 & b_1 \\ 1 & b_2 \end{pmatrix}\right\}, \quad \mathscr{A}_3 := \left\{\begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix}\right\}, \\
\mathscr{A}_4 := &\left\{\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \;\middle|\; b_1 b_2 \neq 0, \frac{a_1 - 1}{b_1} = \frac{a_2 - 1}{b_2}\right\}, \\
\mathscr{A} := &\mathscr{A}_1 \cup \mathscr{A}_2 \cup \mathscr{A}_3 \cup \mathscr{A}_4.
\end{aligned}$$

Theorem 6. *For any $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 2}$, the problems 2-SAT$(1, \boldsymbol{\mathcal{A}})$, WMAX-2-SAT$(1, \boldsymbol{\mathcal{A}})$ and #2-SAT$(1, \boldsymbol{\mathcal{A}})$ are all polynomial-time solvable.*

Theorem 7. *For any $\boldsymbol{\mathcal{A}} \in \mathscr{A}$, the problems 3-SAT$(1, \boldsymbol{\mathcal{A}})$, WMAX-3-SAT$(1, \boldsymbol{\mathcal{A}})$ and #3-SAT$(1, \boldsymbol{\mathcal{A}})$ are all polynomial-time solvable.*

We will make use of the index-width of f to prove the two theorems. The following lemma is needed.

Lemma 2. *Fix $k \geqslant 3$ and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-1)\times 2}$. Given any formula $f \in k$-CNF$(1, \boldsymbol{\mathcal{A}})$, we can construct a layout for f in polynomial time under which f has bandwidth $O(\mathcal{W}(f))$.*

Proof. Let $f \in k$-CNF$(1, \boldsymbol{\mathcal{A}})$. We construct a layout l for f as follows. For any ordered clause $c = (y_{i_1} \vee \cdots \vee y_{i_k}) \in f$, let $l(c) = 2^k i_1 + r$ where $r \in [0, 2^k - 1]$ depends on the polarity of literals of c in the obvious way (2^k possible cases in total; recall that $i_2, \ldots, i_k$ are fixed after i_1 is chosen). This is a valid layout since f contains no duplicated clauses. Furthermore, suppose x_i or its negation appear in both c_1 and c_2. Let $j = \mathcal{I}(c_1, 1)$ and $k = \mathcal{I}(c_2, 1)$. By definition, we have $|j - i| \leqslant \mathcal{W}(f)$ and $|k - i| \leqslant \mathcal{W}(f)$, and therefore

$$|l(c_1) - l(c_2)| \leqslant 2^k |j - k| + 2^k < 2^{k+1}(\mathcal{W}(f) + 1).$$

Thus, f has bandwidth $O(\mathcal{W}(f))$ under layout l. □

Given any k-CNF formula f, we can first apply Lemma 2 to get an equivalent formula f' with bandwidth $O(\mathcal{W}(f))$, and then use the algorithms in [16] (see Theorem 3 in [16]) to solve SAT, MAX-SAT and #SAT on f. Furthermore, their algorithm can be modified to solve the weighted version WMAX-SAT. Thus we obtain the following lemma. We give a self-contained proof of it in Appendix.

Lemma 3. *SAT,* WMAX-*SAT and #SAT can be solved in time $n^{O(1)}2^{O(\mathcal{W}(f))}$ on any input formula f with $\mathcal{I}(f) \subseteq [-n, n]$.*

Lemma 3 implies that we can solve variants of SAT efficiently on instances with $\mathcal{W}(f) = O(\log n)$. Since for any $\boldsymbol{\mathcal{A}} = \begin{pmatrix} 1 & b_1 \\ 1 & b_2 \end{pmatrix}$ and $f \in$ 3-CNF(1, $\boldsymbol{\mathcal{A}}$) it holds that $\mathcal{W}(f) \leqslant \max\{|b_1|, |b_2|, |b_1 - b_2|\} = O(1)$, we have:

Corollary 2. *The result of Theorem 7 holds for all $\boldsymbol{\mathcal{A}} \in \mathscr{A}_2$.*

Given Lemma 3, a natural way for efficiently solving SAT on f is to find a family of injective mappings $\{\mathcal{F}_n\}$ where $\mathcal{F}_n : [-n, n] \to [-n', n']$ for some $n' = n^{O(1)}$, such that the new formula f', obtained by replacing all indices in f with their images under $\mathcal{F}_n$, obeys $\mathcal{W}(f') = O(\log n)$. Notice that the injectivity of $\mathcal{F}_n$ is important since it ensures the SAT-equivalence of f and f'. Now we prove Theorem 6 using this idea.

Proof of Theorem 6. Let $\boldsymbol{\mathcal{A}} = (a, b)$ and $f \in$ 2-CNF(1, $\boldsymbol{\mathcal{A}}$) with $\mathcal{I}(f) \subseteq [-n, n]$. Construct a directed graph $G = (V, E)$ with $V = \{v_i \mid i \in [-n, n]\}$ and $E = \{(v_i, v_j) \mid j = ai + b, j \neq i\}$. Every vertex in V has in-degree at most 1 and out-degree at most 1. Now we prove that every connected component is a chain (a single vertex is regarded as a chain of length 0). If this is not the case, then there exists a connected component

that is a cycle, which indicates the existence of t distinct integers $i_1, i_2, \ldots, i_t$, $t \geqslant 2$, such that $i_{l+1} = ai_l + b$ for all $l \in [1, t-1]$ and $i_1 = ai_t + b$. Note that in this case $a \neq 1$. Simple calculations show that $i_1 = b/(1-a)$. We then reach a contradiction, since $i_2 = ai_1 + b = i_1$.

Thus, E is a collection of disjoint chains. We give an arbitrary ordering on the chains. For every $i \in [-n, n]$, let $q(i), r(i)$ be the integers such that i is the $r(i)$-th node on the $q(i)$-th chain. For every edge $(v_i, v_j) \in E$, we have $q(i) = q(j)$ and $|r(i) - r(j)| = 1$. Now define a function $\mathcal{F}_n$ as $\mathcal{F}_n(i) = (2n+2)q(i) + r(i)$ for all $i \in [-n, n]$. This is an injective function since $r(i) \leqslant 2n+1$. We construct f' by replacing all literals y_i in f with $y_{\mathcal{F}_n(i)}$. It is easy to see that $\mathcal{W}(f') \leqslant 1$, and by Lemma 3 we are done. □

Lemma 4. *The result of Theorem 7 holds for all* $\boldsymbol{\mathcal{A}} \in \mathscr{A}_1$.

Proof. It is obvious that for any $\boldsymbol{\mathcal{A}} \in \mathscr{A}_1$, 3-SAT(1, $\boldsymbol{\mathcal{A}}$) can be reduced to 2-SAT(1, $\boldsymbol{\mathcal{A}}'$) for some $\boldsymbol{\mathcal{A}}' \in \mathbb{Z}^{1\times 2}$. Other variants can be treated analogously. □

Next we deal with $\boldsymbol{\mathcal{A}} = \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix} \in \mathscr{A}_3$, trying to find a family of injective mappings $\{\mathcal{F}_n\}$ as suggested above. There is a simple construction of $\{\mathcal{F}_n\}$ when $|a_1|, |a_2| \geqslant 2$ and a_1 and a_2 are co-prime, in which case every integer m can be uniquely represented by an integer-tuple (m_1, m_2, m_3) such that $m = {a_1}^{m_1} {a_2}^{m_2} m_3$, $a_1 \nmid m_3$ and $a_2 \nmid m_3$. For all $m \in [-n, n]$, let $\mathcal{F}_n(m) = m_3 n + m_1(\lceil \log n \rceil + 1) + m_2$. Since $|m_1|, |m_2| \leqslant \lceil \log n \rceil$, this is an injective mapping for large enough n. The corresponding tuple associated with $a_1 m$ is $(m_1 + 1, m_2, m_3)$, so we have $|\mathcal{F}_n(a_1 m) - \mathcal{F}(m)| = O(\log n)$ and similarly $|\mathcal{F}_n(a_2 m) - \mathcal{F}(m)| = O(1)$, giving us the desired result. However, this method fails if a_1 and a_2 are not co-prime. In the following, we will attack the general case using a different approach.

Lemma 5. *The result of Theorem 7 holds for all* $\boldsymbol{\mathcal{A}} \in \mathscr{A}_3$.

Proof. Fix $a_1, a_2 \in \mathbb{Z} \setminus \{0\}$. Let

$$f \in \text{3-SAT}\left(1, \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix}\right)$$

with

$$\mathcal{I}(f) \subseteq [-n, n], n \geqslant 2.$$

We construct an undirected graph $G = (V, E)$ (note that in the proof of Theorem 6 we use a directed graph) as follows: Let $V = \{v_i \mid i \in [-n, n]\}$, and E be the set of all (v_i, v_j) such that i and j can appear in the same clause of a 3-CNF$\left(1, \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix}\right)$ formula. By our construction, $(v_i, v_j) \in E$ indicates $i = \lambda j$ for $\lambda \in \{a_1, a_2, 1/a_1, 1/a_2, a_1/a_2, a_2/a_1\}$. Notice that the graph G actually only depends on a_1, a_2 and n.

Suppose G has connected components $\{G_i = (V_i, E_i) \mid i \in [1, c^*]\}$, $c^* \geqslant 1$. Inside each component G_i, we pick an arbitrary vertex v (which we call the *root* of G_i) and define $l(v')$ for all $v' \in V_i$ to be the the length of the shortest path between v' and v (note $l(v) = 0$). Then for all possible values t and all $v' \in V_i$ with $l(v') = t$, say, $l(v_{i_1}) = \cdots = l(v_{i_m}) = t$, we define $r(v_{i_q}) = q$ for all $q \in [1, m]$ (notice that the order of vertices in this list can be arbitrary). Finally let $c(v') = i$ for all $v' \in V_i$. In this way we have defined three functions $c(\cdot), l(\cdot)$ and $r(\cdot)$ on V, which can be computed in polynomial time using simple breadth-first-search algorithms. Intuitively, v is the $r(v)$-th node with distance $l(v)$ to the root of $G_{c(v)}$, and $(c(\cdot), l(\cdot), r(\cdot))$ can be seen as a new indexing on the vertices in that $(c(v), l(v), r(v)) = (c(v'), l(v'), r(v'))$ implies $v = v'$. We next prove two lemmas bounding $l(v)$ and $r(v)$ from above.

Lemma 6. *There exists* $c_1 > 0$*, which is independent of* n*, such that* $l(v) \leqslant c_1 \log n$ *for all* $v \in V$.

Proof. For any $v_i \in V$, let v_j be the root of $G_{c(v_i)}$ and P is (one of) the shortest path(s) between v_j and v_i. For any edge $(v_{t_1}, v_{t_2}) \in P$, we have $t_2 = \lambda t_1$ where $\lambda \in \{a_1, a_2, 1/a_1, 1/a_2, a_1/a_2, a_2/a_1\}$, implying that $i = j a_1^{u_1} a_2^{u_2}$ for some $u_1, u_2 \in \mathbb{Z}$ and obviously $l(v_i) \leqslant |u_1| + |u_2|$. Let $\{p_1, \ldots, p_t\}$ be the set of all prime divisors of a_1 or a_2. Due to the fundamental theorem of arithmetic (see e.g., Chapter 3.5 in [17]), we can assume $a_1 = (-1)^{r_0} \prod_{l=1}^{t} p_l^{r_l}$ and $a_2 = (-1)^{r'_0} \prod_{l=1}^{t} p_l^{r'_l}$ where $r_0, r'_0 \in \{0, 1\}$ and $r_1, \ldots, r_t, r'_1, \ldots, r'_t \in \mathbb{Z}^+ \cup \{0\}$. Note that $p_1, \ldots, p_t, r_0, \ldots, r_t, r'_0, \ldots, r'_t$ are all constants that only depend on a_1 and a_2. Suppose $i/j = (-1)^{s_0} \prod_{l=1}^{t} p_l^{s_l}$ where $s_0 \in \{0, 1\}$ and $s_l \in \mathbb{Z}$ for any $l \in [1, t]$. We have $|s_l| \leqslant c \log n$ for some constant c because $p_l \geqslant 2$. Substituting them into $i = j a_1^{u_1} a_2^{u_2}$ gives

$$r_0 u_1 \oplus r'_0 u_2 = s_0 \text{ and } \forall l \in [1, t] : \; r_l u_1 + r'_l u_2 = s_l,$$

where "$\oplus$" denotes the modulo-2 addition. We know that there exists a solution $\boldsymbol{u} = (u_1, u_2)$ to these equations, and $l(v_i) \leqslant |u_1| + |u_2|$ for any solution (u_1, u_2). If there exist $l_1, l_2 \in [1, t]$ s.t. $\begin{vmatrix} r_{l_1} & r'_{l_1} \\ r_{l_2} & r'_{l_2} \end{vmatrix} \neq 0$, we will get a unique solution of $(u_1, u_2) = \left(\begin{vmatrix} s_{l_1} & r'_{l_1} \\ s_{l_2} & r'_{l_2} \end{vmatrix} \Big/ \begin{vmatrix} r_{l_1} & r'_{l_1} \\ r_{l_2} & r'_{l_2} \end{vmatrix}, \begin{vmatrix} r_{l_1} & s_{l_1} \\ r_{l_2} & s_{l_2} \end{vmatrix} \Big/ \begin{vmatrix} r_{l_1} & r'_{l_1} \\ r_{l_2} & r'_{l_2} \end{vmatrix} \right)$ and hence $|u_1|, |u_2| \leqslant O(|s_{l_1}| + |s_{l_2}|) \leqslant c' \log n$ for some constant c'. Otherwise all equations but the first one (which only reflects whether the number is positive or negative) become equivalent. From the property and general formulas of linear diophantine equations over two variables (see,

e.g., Chapter 3.7 in [17]), we know that there exists one solution for which $|u_1|, |u_2| \leqslant O(s_1) \leqslant c'' \log n$ for some constant c''. Therefore $l(v_i) \leqslant 2\max\{c', c''\} \log n$. □

Lemma 7. *There exists $c_2 > 0$, which is independent of n, such that $r(v) \leqslant c_2(\log n + 1)$ for all $v \in V$.*

Proof. Let $v_i \in V$ and v_j be the root of $G_{c(v_i)}$. From the proof of Lemma 6, for any edge $(v_{t_1}, v_{t_2}) \in E$ where $t_1/j = a_1^{e_1} a_2^{e_2}$, we have $t_2/j = a_1^{e_1'} a_2^{e_2'}$ for some $(e_1', e_2') \in \{(e_1+1, e_2), (e_1-1, e_2), (e_1, e_2+1), (e_1, e_2-1), (e_1+1, e_2-1), (e_1-1, e_2+1)\}$, from which follows $|e_1'| \leqslant |e_1|+1, |e_2'| \leqslant |e_2|+1$ and $|e_1'+e_2'| \leqslant |e_1+e_2|+1$. Define $h(v_i) = \min_{i/j = a_1^{u_1} a_2^{u_2}; u_1, u_2 \in \mathbb{Z}} \max\{|u_1|, |u_2|, |u_1+u_2|\}$. Since $h(v_j) = 0$ and $h(v_{t_2}) \leqslant h(v_{t_1}) + 1$ for any edge $(v_{t_1}, v_{t_2}) \in E$, we have $h(v_i) \leqslant l(v_i)$.

On the other hand we prove $l(v_i) \leqslant h(v_i)$ as follows. Assume (u_1^*, u_2^*) witnesses $h(v_i)$; that is, $i/j = a_1^{u_1^*} a_2^{u_2^*}$ and $h(v_i) = \max\{|u_1^*|, |u_2^*|, |u_1^*+u_2^*|\}$. We have either 1) $u_1^* u_2^* \geqslant 0$, and obviously $l(v_i) \leqslant |u_1^*| + |u_2^*| = |u_1^* + u_2^*|$, or 2) $u_1^* u_2^* < 0$, in which case $l(v_i) \leqslant \max\{|u_1^*|, |u_2^*|\}$. (For example, if $u_1^* > 0, u_2^* < 0$ and $|u_1^*| \geqslant |u_2^*|$, we have $a_1^{u_1^*} a_2^{u_2^*} = a_1^{|u_1^*|-|u_2^*|}(a_1/a_2)^{|u_2^*|}$ and thus $|u_1^*|$ edges suffice to connect v_i and v_j; other cases can be proven similarly.) So $l(v_i) = h(v_i)$.

Note that $r(v_i)$ is at most the number of vertices at the same level with v_i. For any fixed value $l(v_i)$, the number of pairs (u_1, u_2) satisfying $l(v_i) = \max\{|u_1|, |u_2|, |u_1+u_2|\}$ is at most $2(2l(v_i)+1) + 2(l(v_i)+1) = 6l(v_i)+4$. Thus $r(v_i) \leqslant 6l(v_i)+4$ and the result is straightforward by Lemma 6. □

We continue the proof of Lemma 5. Define $\mathcal{F}_n : [-n, n] \to \mathbb{Z}$ as $\mathcal{F}_n(i) = c(v_i) \cdot n + l(v_i) \cdot \lceil c_2 \rceil (\lceil \log n \rceil + 2) + r(v_i)$ for any $i \in [-n, n]$, where c_2 is the constant ensured by Lemma 7. By Lemmas 6, 7, there exists $N_0 > 0$ such that for all $n > N_0$, we have $l(v_i) \cdot \lceil c_2 \rceil (\lceil \log n \rceil + 2) + r(v_i) < n$ and $\mathcal{F}(i) \leqslant n^3$. Using the idea of division with remainders, we have $\mathcal{F}_n(i) = \mathcal{F}_n(j) \Leftrightarrow i = j$, showing the injectivity of $\mathcal{F}_n$ for $n > N_0$. Furthermore, for any two indices i, j appearing in the same clause of f, we have $c(v_i) = c(v_j), |l(v_i) - l(v_j)| \leqslant 1, |r(v_i) - r(v_j)| \leqslant O(\log n)$ and thus $|\mathcal{F}_n(i) - \mathcal{F}_n(j)| \leqslant O(\log n)$ holds. Lemma 3 then again gives us a polynomial-time algorithm. Finally, we can just perform the exhaustive search when $n \leqslant N_0$, since N_0 is a constant. □

By a simple reduction to 3-SAT$\left(1, \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix}\right)$, we can also prove that Theorem 7 holds for $\mathcal{A} \in \mathscr{A}_4$. Hence, combining Lemmas 4, 5 with Corollary 2, Theorem 7 follows.

We note that the above proofs rely on the existence of $\{\mathcal{F}_n\}$, whereas in some cases the non-existence of such mapping can be proved. Such an example is given by setting $\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}$, in which case every clause we encounter is like $(y_i \vee y_{i+1} \vee y_{2i})$. Assume that there exists $\{\mathcal{F}_n\}$ where $\mathcal{F}_n : [-n, n] \to \mathbb{Z}$ such that $\max\{|\mathcal{F}_n(i+1) - \mathcal{F}_n(i)|, |\mathcal{F}_n(2i) - \mathcal{F}_n(i)|\} \leqslant O(\log n)$. Since for any $i \in [1, n]$ there exists a list of integers $\{i_0 = 1, i_1, \ldots, i_k = i\}$ where $i_{l+1} \in \{i_l + 1, 2i_l\}$ and $k = O(\log n)$, we have $|\mathcal{F}(i) - \mathcal{F}(1)| \leqslant O(\log^2 n)$ for all $1 \leqslant i \leqslant n$, which contradicts the injectivity of $\mathcal{F}_n$ for large enough n.

Finally, we show the following theorem for the case $k > 3$. It basically follows a similar line as above, with some more careful (and more complicated) analysis in dealing with the parameters.

Theorem 8. *Let $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-1)\times 2}$ with $k \geqslant 4$. Then the problems k-SAT$(1, \mathcal{A})$, WMax-k-SAT$(1, \mathcal{A})$ and #k-SAT$(1, \mathcal{A})$ are*

- *solvable in polynomial time if $A[1,1] = A[2,1] = \cdots = A[k-1,1] = 1$;*
- *solvable in time $n^{O(\log^{k-3}(n))}$ if $A[1,2] = A[2,2] = \cdots = A[k-1,2] = 0$.*

Proof. The first case is very easy, so we focus on the second case where $A[1,2] = \cdots = A[k-1,2] = 0$. Let $a_i = A[i,1]$, $1 \leqslant i \leqslant k-1$. As what is done in the proof of Lemma 5, we construct a graph $G = (V, E)$ where $V = [-n, n]$ and E is the set of all pairs (i, j) that can appear in the same input clause. For $v \in V$, define $c(v), l(v)$ and $r(v)$ analogously. We will show that $l(v) = O(\log n)$ and $r(v) = O(\log^{k-2}(n))$. Having this, we can define an injective function $\mathcal{F}(i) = n + l(v_i) \cdot c\lceil \log^{k-2}(n) + 1 \rceil + r(v_i)$ for some suitable constant c and large enough n. Applying this mapping to all the indices appeared in the input CNF formula will produce a new CNF with index-width $O(\log^{k-2}(n))$. Then by using Lemma 3, the problem can be solved in time $n^{O(1)} 2^{O(\log^{k-2}(n))} = n^{O(\log^{k-3}(n))}$.

It remains to bound $l(v_i)$ and $r(v_i)$ for all $v_i \in V$. Let v_j be the root of $G_{c(v_i)}$. Similar to the argument used in Lemma 6, we have $i/j = \prod_{m=1}^{k-1} a_m^{u_m}$ where $u_1, \ldots, u_{k-1} \in \mathbb{Z}$, and $l(v_i) \leqslant \sum_{m=1}^{k-1} |u_m|$ for all such u_m's. Let $\{p_1, \ldots, p_t\}$ be the set of primes that are divisors of some a_m. For any $m \in [1, k-1]$, assume $a_m = (-1)^{r_{m,0}} \prod_{l=1}^{t} p_l^{r_{m,l}}$, where $r_{m,0} \in \{0, 1\}$ and $r_{m,l} \in \mathbb{Z}^+ \cup \{0\}$ for $l > 0$. Note that these integers are all constants when $a_1, \ldots, a_{k-1}$ are fixed. Suppose $i/j = (-1)^{s_0} \prod_{l=1}^{t} p_l^{s_l}$, where $s_0 \in \{0, 1\}$ and $s_l = O(\log n)$ for $l > 0$. Comparing the two expressions of i/j we have

$$\bigoplus_{m=1}^{k-1} r_{m,0} u_m = s_0 \text{ and } \forall l \in [1, t] : \sum_{m=1}^{k-1} r_{m,l} u_m = s_l.$$

We know that there exists a solution $\boldsymbol{u} =$

$(u_1, \ldots, u_{k-1})$ to these equations, and $l(v_i) \leqslant \sum_{m=1}^{k-1} |u_m|$ for any solution $\boldsymbol{u}$. If there exists $k-1$ linearly independent equations (out of the latter t ones), then there is a unique solution $\boldsymbol{u}$ and it satisfies $u_i \leqslant O(k \cdot \max_{l\in[1,t]}\{s_l\}) = O(\log n)$ for all $i \in [1, m]$. Otherwise, there are at most $k-2$ (which is a constant) independent linear diophantine equations with constant coefficients in front of the variables. By the existing theory of linear diophantine equations (see [18]), there exists a solution $\boldsymbol{u} = (u_1, \ldots, u_{k-1})$ with $u_m \leqslant O((k \cdot \max_{l\in[1,t]}\{s_l\}) = O(\log n)$ for all m, that also satisfies the parity equation involving s_0. (Observe that if the minimum solution violates the parity constraint, then one of its neighbor solutions must obey it, otherwise all solutions will violate the constraint, which contradicts the fact that there exists one solution. For more details about the solution space of linear diophantine equations, we refer the readers to [19-20].) Hence, $l(v_i) \leqslant O((k-1)\log n) = O(\log n)$.

Now we prove $r(v_i) \leqslant O(\log^{k-2}(n))$ by a similar technique to that used in Lemma 7. Define

$$h(v_i) = \min_{\substack{(u_1,\ldots,u_{k-1})\in\mathbb{Z}^{k-1}:\\ i/j=\prod_{m\in[1,k-1]} a_m^{u_m}}} \left\{ \max_{\substack{S\subseteq[1,k-1]\\ S\neq\varnothing}} \left\{ \left| \sum_{m\in S} u_m \right| \right\} \right\}.$$

We show that $h(v_i) = l(v_i)$. For any $(i_1, i_2) \in E$, given $i_1/j = \prod_{m\in[1,k-1]} a_m^{u_m}$, we know that $i_2/j = \prod_{m\in[1,k-1]} a_m^{u'_m}$ for some $\boldsymbol{u'}$, such that either 1) $|u'_{m_1} - u_{m_1}| = 1$ for some m_1 and $u'_m = u_m$ for all $m \neq m_1$, or 2) $u'_{m_1} = u_{m_1} + 1, u'_{m_2} = u_{m_2} - 1$ for some m_1, m_2 and $u'_m = u_m$ for all $m \in [1, k-1] \setminus \{m_1, m_2\}$. Therefore, $|\sum_{m\in S} u'_m| \leqslant 1 + |\sum_{m\in S} u_m|$ holds for all $S \subseteq [1, k-1]$, and consequently $h(v_{i_2}) \leqslant h(v_{i_1}) + 1$. Thus $h(v_i) \leqslant l(v_i)$.

Next we prove $l(v_i) \leqslant h(v_i)$. Let $\boldsymbol{u}$ be such that $i/j = \prod_{m=1}^{k-1} a_m^{u_m}$. It suffices to prove that $l(v_i) \leqslant \max_{S\subseteq[1,k-1],S\neq\varnothing}\{u_S\}$ for every such $\boldsymbol{u}$, where $u_S = |\sum_{m\in S} u_m|$. Let $S^+ = \{u_m \mid u^*_m \geqslant 0\}$ and $S^- = \{u_m \mid u_m < 0\}$. Then $\max_{S\subseteq[1,k-1],S\neq\varnothing}\{u_S\} = \max\{u_{S^+}, u_{S^-}\}$.

We will prove the equivalent statement $l(v_i) \leqslant \max\{u_{S^+}, u_{S^-}\}$ by induction on k. The base case $k = 3$ is proved in Lemma 7. Assume $k \geqslant 4$ and the statement holds for all smaller k. If one of S^+ and S^-, say S^+, is empty, then $l(v_i) \leqslant \sum_{m=1}^{k-1} |u_m| = u_{S^-} = h(v_i)$. Otherwise, we pick arbitrarily $u_{m_1} \in S^+$ and $u_{m_2} \in S^-$, and without loss of generality, we assume $|u_{m_1}| \geqslant |u_{m_2}|$. Observe that in the graph G, through $|u_{m_2}|$ edges one can reach from i a new vertex i' such that $i'/j = \prod_{m=1}^{k-1} a_m^{u'_m}$, where $u'_{m_1} = u_{m_1} - |u_{m,2}|, u'_{m_2} = 0$, and $u'_m = u_m$ for all other m. Since u'_{m_2} becomes 0, we can ignore it and only consider the rest elements, and thus reduce to the case $k' = k-1$. For the new case, we can similarly define $S'^+, S'^-, u'_{S'^+}$ and $u'_{S'^-}$. Now by the induction hypothesis, $l(v_{i'}) \leqslant \max\{u'_{S'^+}, u'_{S'^-}\}$. It is easy to see that $u'_{S'^+} = u_{S^+} - |u_{m_2}|$ and $u'_{S'^-} = u_{S^-} - |u_{m_2}|$. Hence, we have $l(v_i) \leqslant |u_{m_2}| + l(v_{i'}) \leqslant \max\{u_{S^+}, u_{S^-}\}$. This finishes the induction proof.

Note that $r(v_i)$ is at most the number of vertices at the same level with v_i, which is at most the number of solutions $\boldsymbol{u}$ to (1). Let $l = h(v_i) = l(v_i)$. For each $S \subseteq [1, k-1]$ with $|S| = t$, there are at most $(2l+1)^{t-1}$ solutions $\boldsymbol{u}$ satisfying $|\sum_{m\in S} u_m| = l$ (since $|u_m| \leqslant l$ for all m). Thus $r(v_i) \leqslant 2^{k-1}(2l+1)^{k-2} = O(\log^{k-2}(n))$, completing the proof of Theorem 8. □

5 Discussions and Open Problems

We list several interesting questions that are left for future work.

• Can we decide the complexity of 3-SAT(1, $\boldsymbol{\mathcal{A}}$) for every $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 2}$? It is tempting to conjecture that 3-SAT(1, $\boldsymbol{\mathcal{A}}$) is polynomial-time solvable for every $\boldsymbol{\mathcal{A}}$. However, proving this requires new techniques and insights into the structure of the instances. More ambitiously, can we characterize the complexity of k-SAT(1, $\boldsymbol{\mathcal{A}}$) for every $k \geqslant 3$ and $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{(k-1)\times 2}$?

• Can we design polynomial-time approximation algorithms for 3-SAT(2, $\boldsymbol{\mathcal{A}}$) for some $\boldsymbol{\mathcal{A}} \in \mathbb{Z}^{1\times 3} \setminus \mathit{Easy}\mathcal{A}$ with performance guarantee better than 7/8 (the tight approximation threshold for 3-SAT)? Can we achieve faster exact-algorithms for the index-dependent SAT variants?

• Can we prove threshold behaviors for random instances of k-SAT($m, \boldsymbol{\mathcal{A}}$) similar to that of random k-SAT? (Here the "random instances" should be defined carefully.)

References

[1] Cook S A. The complexity of theorem proving procedures. In *Proc. ACM STOC*, May 1971, pp.151-158.

[2] Aspvall B, Plass M F, Tarjan R E. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 1979, 8(3): 121-123.

[3] Valiant L G. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 1979, 8(2): 189-201.

[4] Valiant L G. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 1979, 8(3): 410-421.

[5] Arora S, Lund C, Motwani R, Sudan M, Szegedy M. Proof verification and the hardness of approximation problems. *J. ACM*, 1998, 45(3): 501-555.

[6] Håstad J. Some optimal inapproximability results. *J. ACM*, 2001, 48(4): 798-859.

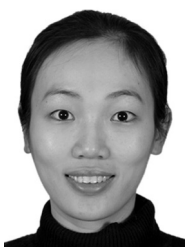
[7] Henschen L, Wos L. Unit refutations and Horn sets. *J. ACM*, 1974, 21(4): 590-605.

[8] Yamasaki S, Doshita S. The satisfiability problem for the class consisting of Horn sentences and some non-Horn sentences in propositional logic. *Infor. Control*, 1983, 59(1-3): 1-12.

[9] Schaefer T J. The complexity of satisfiability problems. In *Proc. ACM STOC*, May 1978, pp.216-226.

[10] Allender E, Bauland M, Immerman N, Schnoor H, Vollmer H. The complexity of satisfiability problems: Refining Schaefer's theorem. *J. Comput. System Sci.*, 2009, 75(4): 245-254.
[11] Tovey C A. A simplified NP-complete satisfiability problem. *Discrete Appl. Math.*, 1984, 8(1): 85-89.
[12] Kratochvíl J, Savický P, Tuza Z. One more occurrence of variables makes satisfiability jump from trivial to NP-complete. *SIAM J. Comput.*, 1993, 22(1): 203-210.
[13] Gebauer H, Szabó T, Tardos G. The local lemma is tight for SAT. In *Proc. the 22nd ACM-SIAM Symposium on Discrete Algorithms* (*SODA*), Jan. 2011, pp.664-674.
[14] Lichtenstein D. Planar formulae and their uses. *SIAM J. Comput.*, 1982, 11(2): 329-343.
[15] Monien B, Sudborough I H. Bandwidth constrained NP-complete problems. In *Proc. ACM STOC*, May 1981, pp.207-217.
[16] Georgiou K, Papakonstantinou P A. Complexity and algorithms for well-structured k-SAT instances. In *Proc. the 11th International Conference on Theory and Applications of Satisfiability Testing* (*SAT*), May 2008, pp.105-118.
[17] Rosen K H. Elementary Number Theory and its Applications (5th Edition), Addison Wesley, 2004.
[18] Borosh I, Flahive M, Rubin D, Treybig B. A sharp bound for solutions of linear diophantine equations. *P. Am. Math. Soc.*, 1989, 105(4): 844-846.
[19] Borosh I, Flahive M, Treybig B. Small solution of linear Diophantine equations. *Discrete Math.*, 1986, 58(3): 215-220.
[20] Bradley G H. Algorithms for Hermite and Smith normal matrices and linear diophantine equations. *Math. Comput.*, 1971, 25(116): 897-907.

Hong-Yu Liang received his B.E. degree in computer science from Tsinghua University in 2008. He is currently a Ph.D. candidate at the Institute for Interdisciplinary Information Sciences, Tsinghua University. His research interests lie in complexity theory, approximation algorithms, and graph algorithms.



Jing He received her B.E. degree in computer science from Tsinghua University in 2008. She is now a Ph.D. candidate at the Institute for Interdisciplinary Information Sciences, Tsinghua University. Her current research interests include algorithmic problems in social networks, approximation algorithms, and natural language processing.

Appendix

Proof of Lemma 3. Let $w = \mathcal{W}(f)$ and define $S_i = [i-n-1, i-n+w-2]$ for all $i \in [1, n']$, where $n' = 2n-w+2$. Let $\mathscr{S} = \{S_i \mid i \in [1, n']\}$. By definition for any clause $c \in f$ there exists $S \in \mathscr{S}$ s.t. $\mathcal{I}(c) \subseteq S$. Let $f|_S$ denote the sub-formula of f obtained by taking the conjunction of all clauses $c \in f$ for which $\mathcal{I}(c) \subseteq S$.

We construct a directed graph G consisting of $n'+2$ layers as follows. Layer 0 contains a single vertex v_{start} and layer $n'+1$ contains a single vertex v_{end}. For $1 \leqslant i \leqslant n'$, layer i consists of vertices labeled by different partial assignments satisfying $f|_{S_i}$. More specifically, every vertex in layer i is labeled as $v_{\mathcal{B}}^{S_i}$, where $\mathcal{B} \in \{0,1\}^{|S_i|}$ is a satisfying assignment for $f|_{S_i}$. Then, for every two vertices in consecutive layers (say, layers i and $i+1$), we add an edge from the one in Layer i to another if and only if the two partial assignments associated with them are consistent. Finally we add an edge from v_{start} to every node in layer 1, and an edge from every node in layer n' to v_{end}. It is easy to see that G is constructible in time $n^{O(1)}2^{O(w)}$.

We claim that there is a 1-1 correspondence between all satisfying assignments of f and all the paths from v_{start} to v_{end}. If f has a satisfying assignment $\mathcal{B}$, we can pick for each $S_i \in \mathscr{S}$ the vertex $v_{\mathcal{B}|_{S_i}}^{S_i}$ where $\mathcal{B}|_{S_i}$ denotes the assignment $\mathcal{B}$ restricted on $\{x_i \mid i \in S_i\}$. It is transparent that $(v_{\text{start}}, v_{\mathcal{B}|_{S_1}}^{S_1}, \ldots, v_{\mathcal{B}|_{S_{n'}}}^{S_{n'}}, v_{\text{end}})$ is a path, and this mapping from assignments to paths is injective. Conversely, assuming the existence of a path $(v_{\text{start}}, v_{\mathcal{B}_1}^{S_1}, \ldots, v_{\mathcal{B}_{n'}}^{S_{n'}}, v_{\text{end}})$, it is easy to see that $\mathcal{B}_i$ and $\mathcal{B}_j$ are consistent for all $i, j \in [1, n']$. Such a set of pairwise-consistent partial assignments naturally induces a total assignment $\mathcal{B}$ for f. Since $\forall c \in f$ $\exists S_i \in \mathscr{S}$ s.t. $\mathcal{I}(c) \subseteq S_i$, we know that c is satisfied by $\mathcal{B}_i = \mathcal{B}|_{S_i}$ and hence f is satisfied by $\mathcal{B}$. This mapping from paths to assignments is also injective, since we have $\mathcal{B}_i = \mathcal{B}|_{S_i}$ for all $i \in [1, n']$. Therefore the number of paths between v_{start} and v_{end} equals that of satisfying assignments of f. Finally, counting the number of such paths can be done in time polynomial in the number of vertices in G, which is $n^{O(1)}2^{O(w)}$ (just perform some matrix multiplications on the adjacency matrix of G). Thus we can solve both SAT and #SAT in time $n^{O(1)}2^{O(w)}$.

To solve WMax-SAT we need to modify the construction of G. For any $1 \leqslant i \leqslant n'$, Layer i now contains $2^{|S_i|}$ vertices $\{v_{\mathcal{B}}^{S_i} \mid \mathcal{B} \in \{0,1\}^{|S_i|}\}$ corresponding to all possible assignments of $f|_{S_i}$. For any two vertices in consecutive layers, say $v_{\mathcal{B}}^{S_i}$ and $v_{\mathcal{B}'}^{S_{i+1}}$, there is an edge from the former to the latter if and only if $\mathcal{B}$ and $\mathcal{B}'$ are consistent, and this edge is associated with a weight, which is defined as the sum of the weights of all clauses $c \in f|_{S_{i+1}}$ for which c is satisfied by $\mathcal{B}'$ but not yet satisfied by $\mathcal{B}$. We also add an edge from v_{start} to every node $v_{\mathcal{B}}^{S_1}$ in layer 1 with weight equal to the sum of weights of all clauses $c \in f|_{S_1}$ satisfied by $\mathcal{B}$, and an edge from every node in layer n' to v_{end} with weight 0. By a similar argument, we obtain that there is an assignment of f satisfying a total weight of at least w if and only if there is a path in G with weight at least w. Finding a path in G with maximum weight can be done in time $n^{O(1)}2^{O(w)}$, using the standard weighted-shortest-path algorithm with all weights negated. Therefore WMax-SAT can also be solved in time $n^{O(1)}2^{O(w)}$. □