Satisfiability with Index Dependency*

Hongyu Liang and Jing He

Institute for Theoretical Computer Science,
Tsinghua National Laboratory for Information Science and Technology (TNList),
Tsinghua University, Beijing, China
{hongyuliang86,hejing2929}@gmail.com

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 $m \geq 2$, 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 $k \geq 3$, 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, bandwidth, dichotomy.

1 Introduction

The Boolean Satisfiability Problem (SAT) has received extensive studies since its first proposed as a natural NP-complete problem in [7]. It is also well known that

* This work was supported in part by the National Natural Science Foundation of China Grant 60553001, 61073174, 61033001 and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

O. Cheong, K.-Y. Chwa, and K. Park (Eds.): ISAAC 2010, Part I, LNCS 6506, pp. 49–60, 2010.

3-SAT is still NP-complete [7] while 2-SAT is linear-time tractable [3], where by k-SAT we mean the 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 [18], and he proved in [19] 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 APX-hard [2,10].

People thus became interested in examining the complexity of certain restricted versions of SAT, such as Horn-SAT [11,20] and 1-in-3-SAT [16], 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 [16], 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 $k \geq 1$. 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. [1] 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 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 4 times [17], cannot be derived from Schaefer's theorem. An interesting point is that 3-SAT with each variable occuring at most 3 times is always satisfiable [17]. This "phase transition" phenomenon is generalized to k-SAT by Kratochvíl, Savický and Tuza [12]. (See [8] for a recent result for determining this threshold.) Other examples include the NP-completeness of the PLANAR-3-SAT problem [13], which is to decide whether a given 3-CNF formula with a planar incidence graph is satisfiable.

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 [16]. 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 doesn't 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 $\mathcal{A}$ and a vector b, and requiring that $\mathcal{A}(a_{i,1}, \ldots, a_{i,k})^{\mathrm{T}} = b$ 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. More precisely, there exists an integer m and a matrix $\mathcal{A} \in \mathbb{Z}^{(k-m)\times(m+1)}$ such that

$$\forall i \in \{1, 2, \ldots, t\}, (a_{i,m+1}, \ldots, a_{i,k})^{\mathrm{T}} = \mathcal{A}(a_{i,1}, \ldots, a_{i,m}, 1)^{\mathrm{T}}.$$

For instance, when $\mathcal{A} = \begin{pmatrix} 1 & 1 & 0 \\ 2 & -1 & -3 \end{pmatrix}$, 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$(m, \mathcal{A})$. 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 such classes of SAT problems. In the first part of this article (Section 3), we prove a dichotomy theorem for the case $m \geq 2$. More precisely, for any $k \geq m \geq 2$ 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 $P \neq NP$. We also prove that for any $k \geq m \geq 2$ and $\mathcal{A} \in \mathbb{Z}^{(k-m)\times(m+1)}$, #$k$-SAT$(m, \mathcal{A})$ is #P-complete and Max-k-SAT$(m, \mathcal{A})$ does not have a polynomial time approximation scheme, unless $P = NP$.

The second part (Section 4) of this paper is devoted to the case $m = 1$, where things become subtler. We develop a 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* [14] (also referred to as the *diameter* in [9]). 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 are able to prove that the two measures have the same magnitude up to some constant factor when $m = 1$. However, we will not present the proof in this version.) This guarantees that the algorithm in [9] 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.

Keeping the ideas in mind and the techniques at hand, we first show that for any $\mathcal{A} \in \mathbb{Z}^{1\times 2}$, WMax-2-SAT$(1, \mathcal{A})$ (the weighted version of MAX-SAT) and #2-SAT$(1, \mathcal{A})$ are both polynomial-time solvable. This is in sharp contrast with the hardness results of general #2-SAT [19] and Max-2SAT [10]. We then consider the case where $m = 1$ and $k = 3$. We show that there is a large (and surely non-trivial) class of $\mathcal{A} \in \mathbb{Z}^{2\times 2}$ for which both WMax-k-SAT$(1, \mathcal{A})$ and #k-SAT$(1, \mathcal{A})$ can be solved in polynomial time. This is harder to prove than the previous case $k = 2$. We then generalize the results to the case $k > 3$, showing that a large and natural class of $\mathcal{A}$ makes the problems solvable in DTIME$(n^{\text{polylog}(n)})$, which is generally believed not to contain any NP-hard problem. We also believe that, besides the obtained results, the concept of index-width and the techniques used are of their own interest.

Due to space limitations, several proofs are omitted and will appear in the full version of this paper.

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 $\mathcal{A}$ 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 [14], 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 $|l(c_i) - l(c_j)| \leq k$ holds. The bandwidth of f is the minimum bandwidth of f under all possible layouts.

Definition 1. *Let $k \geq m \geq 1$ be two positive integers and $\mathcal{A} \in \mathbb{Z}^{(k-m)\times(m+1)}$. Define k-CNF$(m, \mathcal{A})$ to be the collection of all k-CNF f such that for any clause $c \in f$, it holds that $(\mathcal{I}(c, m+1), \ldots, \mathcal{I}(c, k))^T = \mathcal{A}(\mathcal{I}(c, 1), \ldots, \mathcal{I}(c, m), 1)^T$. With a little abuse of notation, we also use k-CNF$(m, \mathcal{A})$ to denote a formula chosen from this set.*[1]

Definition 2. *Let $k \geq 2, m \in [1, k]$ and $\mathcal{A} \in \mathbb{Z}^{(k-m)\times(m+1)}$. An instance of k-SAT$(m, \mathcal{A})$ consists of a string 1^n as well as a formula $f \in k$-CNF$(m, \mathcal{A})$ 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$(m, \mathcal{A})$, Max-k-CNF$(m, \mathcal{A})$, WMax-k-CNF$(m, \mathcal{A})$ *and* #k-CNF$(m, \mathcal{A})$ *to be the decision version, the optimization version, the weighted optimization version and the counting version of k-SAT$(m, \mathcal{A})$, respectively.*[2]

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

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

We examine the complexity of variants of 3-SAT(2,$\mathcal{A}$). In this case $\mathcal{A}$ is just a 3-dimensional integer vector (a, b, c), and every clause of an instance has the form

[1] For example, any clause of a formula $f \in$ 4-CNF$(2, \begin{pmatrix} 1 & 1 & 0 \\ 2 & -1 & -3 \end{pmatrix})$ has the form $y_i \vee y_j \vee y_{i+j} \vee y_{2i-j-3}$.

[2] In our notation, D-k-SAT$(k, \emptyset)$, Max-k-SAT$(k, \emptyset)$ and #k-SAT$(k, \emptyset)$ correspond to the canonical problems k-SAT, Max-k-SAT and #k-SAT, respectively.

$y_i \vee y_j \vee y_{ai+bj+c}$. It is easy to find some special vectors $\mathcal{A}$ for which 3-SAT(2,$\mathcal{A}$) degenerates to 2-SAT: When $\mathcal{A} = (0,1,0)$ or $(1,0,0)$ it is obvious, and when $\mathcal{A} = (0,0,c)$ it suffices to examine the two 2-SAT instances where x_c is set to 0 and 1 respectively. Therefore 3-SAT(2,$\mathcal{A}$) can be solved in polynomial time under these choices of $\mathcal{A}$.

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

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

We also obtain the following inapproximability results of the corresponding optimization problems.

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

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

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

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

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

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

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

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

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

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

- D-k-SAT($m,\mathcal{A}$) *is either in P or NP-complete, assuming $P \neq NP$;*
- #k-SAT($m,\mathcal{A}$) *is always #P-complete;*
- Max-SAT($m,\mathcal{A}$) *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 $\mathcal{A} \in \mathbb{Z}^{1\times 2}$, all the variants of 2-SAT(1,$\mathcal{A}$) defined previously are polynomial-time

solvable. For $k \geq 3$, there is a large class of non-trivial choices of $\mathcal{A} \in \mathbb{Z}^{(k-1)\times 2}$ for which the variants of k-SAT(1,$\mathcal{A}$) can all be solved in polynomial time or quasi-polynomial time (DTIME($n^{\text{polylog}(n)}$)). In this short version, we will focus on the case $k \leq 3$. We also 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 $\mathbb{Z}^{1\times 2}$:

$$\mathscr{A}_1 := \left\{ \begin{pmatrix} 0 & b_1 \\ a_2 & b_2 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \right\} \bigcup \left\{ \begin{pmatrix} a_1 & b_1 \\ 0 & b_2 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \right\} \bigcup \left\{ \begin{pmatrix} 1 & 0 \\ a_2 & b_2 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \right\}$$
$$\bigcup \left\{ \begin{pmatrix} a_1 & b_1 \\ 1 & 0 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \right\} \bigcup \left\{ \begin{pmatrix} a_1 & b_1 \\ a_1 & b_1 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \right\};$$
$$\mathscr{A}_2 := \left\{ \begin{pmatrix} 1 & b_1 \\ 1 & b_2 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \right\}; \mathscr{A}_3 := \left\{ \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \right\};$$
$$\mathscr{A}_4 := \left\{ \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \in \mathbb{Z}^{2\times 2} \mid b_1 b_2 \neq 0 \text{ and } (a_1 - 1)/b_1 = (a_2 - 1)/b_2 \right\};$$
$$\mathscr{A} := \mathscr{A}_1 \cup \mathscr{A}_2 \cup \mathscr{A}_3 \cup \mathscr{A}_4.$$

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

Theorem 7. *For any $\mathcal{A} \in \mathscr{A}$, the problems* 3-SAT(1,$\mathcal{A}$), WMax-3-SAT(1,$\mathcal{A}$) *and* #3-SAT(1,$\mathcal{A}$) *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 1. *Fix $k \geq 3$ and $\mathcal{A} \in \mathbb{Z}^{(k-1)\times 2}$. Given any formula $f \in k$-CNF$(1, \mathcal{A})$, 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$(1, \mathcal{A})$. We construct a layout l for f as follows. For any ordered clause $c = (y_{i_1} \vee \ldots \vee y_{i_k}) \in f$, 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 $|j - i| \leq \mathcal{W}(f)$ and $|k - i| \leq \mathcal{W}(f)$, and therefore $|l(c_1) - l(c_2)| \leq 2^k |j - k| + 2^k < 2^{k+1}(\mathcal{W}(f) + 1)$. Thus, f has bandwidth $O(\mathcal{W}(f))$ under layout l. □

Given any k-CNF formula f, we can first apply Lemma 1 to get an equivalent formula f' with bandwidth $O(\mathcal{W}(f))$, and then use the algorithms in [9] (see Theorem 3 in [9]) 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.

Lemma 2. 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 2 implies that we can solve variants of SAT efficiently on instances with $\mathcal{W}(f) = O(\log n)$. Since for any $\mathcal{A} = \begin{pmatrix} 1 & b_1 \\ 1 & b_2 \end{pmatrix}$ and $f \in$ 3-CNF(1,$\mathcal{A}$) it holds that $\mathcal{W}(f) \leq \max\{|b_1|, |b_2|, |b_1 - b_2|\} = O(1)$, we have:

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

Given Lemma 2, 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 $\mathcal{A} = (a, b)$ and $f \in$ 2-CNF(1,$\mathcal{A}$) 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 $i_1, i_2, \ldots, i_t$, $t \geq 2$, 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 $r(i) \leq 2n + 1$. We construct f' by replacing all literals y_i in f with $y_{\mathcal{F}_n(i)}$. It is easy to see that $\mathcal{W}(f') \leq 1$, and by Lemma 2 we are done. □

Lemma 3. *The result of Theorem 7 holds for all $\mathcal{A} \in \mathscr{A}_1$.*

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

Next we deal with $\mathcal{A} = \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix} \in \mathscr{A}_3$, trying to find a family of injective mappings $\{\mathcal{F}_n\}$ as suggested above. There is a simple construction of $\{\mathcal{F}_n\}$ when $|a_1|, |a_2| \geq 2$ 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 $|m_1|, |m_2| \leq \lceil \log n \rceil$, 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 4. *The result of Theorem 7 holds for all* $\mathcal{A} \in \mathscr{A}_3$.

Proof. Fix $a_1, a_2 \in \mathbb{Z} \setminus \{0\}$. Let $f \in$ 3-SAT$(1, \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix})$ with $\mathcal{I}(f) \subseteq [-n, n], n \geq 2$. 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$(1, \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix})$ 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 $\{G_i = (V_i, E_i) \mid i \in [1, c^*]\}, c^* \geq 1$. 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, $l(v_{i_1}) = \ldots = l(v_{i_m}) = t$, 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 5. *There exists* $c_1 > 0$*, which is independent of* n*, such that* $l(v) \leq c_1 \log n$ *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 $l(v_i) \leq |u_1| + |u_2|$. 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 [15]), 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 $|s_l| \leq c \log n$ for some constant c because $p_l \geq 2$. 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 $\mathbf{u} = (u_1, u_2)$ to these equations, and $l(v_i) \leq |u_1| + |u_2|$ 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 $(u_1, u_2) = \left(\begin{vmatrix} s_{l_1} & r'_{l_1} \\ s_{l_2} & r'_{l_2} \end{vmatrix} / \begin{vmatrix} r_{l_1} & r'_{l_1} \\ r_{l_2} & r'_{l_2} \end{vmatrix}, \begin{vmatrix} r_{l_1} & s_{l_1} \\ r_{l_2} & s_{l_2} \end{vmatrix} / \begin{vmatrix} r_{l_1} & r'_{l_1} \\ r_{l_2} & r'_{l_2} \end{vmatrix} \right)$ and hence

$|u_1|, |u_2| \leq O(|s_{l_1}| + |s_{l_2}|) \leq c' \log n$ 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 [15]), we know that there exists one solution for which $|u_1|, |u_2| \leq O(s_1) \leq c'' \log n$ for some constant c''. Therefore $l(v_i) \leq 2 \max\{c', c''\} \log n$. □

Lemma 6. *There exists $c_2 > 0$, which is independent of n, such that $r(v) \leq c_2(\log n + 1)$ 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 5, 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 $|e_1'| \leq |e_1|+1, |e_2'| \leq |e_2|+1$ and $|e_1'+e_2'| \leq |e_1+e_2|+1$. 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 $h(v_{t_2}) \leq h(v_{t_1}) + 1$ for any edge $(v_{t_1}, v_{t_2}) \in E$, we have $h(v_i) \leq l(v_i)$.

On the other hand we prove $l(v_i) \leq h(v_i)$ 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) $u_1^* u_2^* \geq 0$, and obviously $l(v_i) \leq |u_1^*| + |u_2^*| = |u_1^* + u_2^*|$, or (2) $u_1^* u_2^* < 0$, in which case $l(v_i) \leq \max\{|u_1^*|, |u_2^*|\}$ (for example, if $u_1^* > 0, u_2^* < 0$ and $|u_1^*| \geq |u_2^*|$, 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 $r(v_i) \leq 6l(v_i)+4$ and the result is straightforward by Lemma 5. □

We continue the proof of Lemma 4. 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 6. By Lemmas 5, 6, 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 $\mathcal{F}(i) \leq n^3$. 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 $c(v_i) = c(v_j), |l(v_i) - l(v_j)| \leq 1, |r(v_i) - r(v_j)| \leq O(\log n)$ and thus $|\mathcal{F}_n(i) - \mathcal{F}_n(j)| \leq O(\log n)$ holds. Lemma 2 then again gives us a polynomial-time algorithm. Finally, we can just perform the exhaustive search when $n \leq N_0$, since N_0 is a constant. □

By a simple reduction to 3-SAT$(1, \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix})$, we can also prove that Theorem 7 holds for $\mathcal{A} \in \mathscr{A}_4$. Hence, combining with Lemmas 3,4 and 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 $\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}$, for which it is provable that the formulas

have index-width $\Omega(n/\log n)$ under any relabeling of the variables. We omit the details.

Finally, we mention the following theorem for the case $k > 3$, the proof of which is omitted from this short version. It basically follows a similar line as above, with some more careful (and more complicated) analysis in dealing with the parameters.

Theorem 8. *Let $\mathcal{A} \in \mathbb{Z}^{(k-1)\times 2}$ with $k \geq 4$. 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 $A[1,1] = A[2,1] = \ldots = A[k-1,1] = 1$;*
- *solvable in time $n^{O(\log^{k-3}(n))}$ if $A[1,2] = A[2,2] = \ldots = A[k-1,2] = 0$.*

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,$\mathcal{A}$) for every $\mathcal{A} \in \mathbb{Z}^{1\times 2}$? It is tempting to conjecture that 3-SAT(1,$\mathcal{A}$) is polynomial-time solvable for every $\mathcal{A}$. 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,$\mathcal{A}$) for every $k \geq 3$ and $\mathcal{A} \in \mathbb{Z}^{(k-1)\times 2}$?
- Can we design polynomial-time approximation algorithms for 3-SAT(2,$\mathcal{A}$) for some $\mathcal{A} \in \mathbb{Z}^{1\times 3} \setminus Easy\mathcal{A}$ 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($m, \mathcal{A}$) similar to that of random k-SAT? (Here the "random instances" should be defined carefully.)

Acknowledgements. The authors are grateful to the anonymous referees for their helpful comments and suggestions on an early version of this paper.

References

1. Allender, E., Bauland, M., Immerman, N., Schnoor, H., Vollmer, H.: The complexity of satisfiability problems: refining Schaefer's theorem. J. Comput. System Sci. 75(4), 245–254 (2009)
2. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
3. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inf. Process. Lett. 8(3), 121–123 (1979)
4. Borosh, I., Flahive, M., Rubin, D., Treybig, B.: A Sharp Bound for Solutions of Linear Diophantine Equations. Proceedings of the American Mathematical Society 105(4), 844–846 (1989)
5. Borosh, I., Flahive, M., Treybig, B.: Small solution of linear Diophantine equations. Discrete Mathematics 58(3), 215–220 (1986)

6. Bradley, G.H.: Algorithms for Hermite and Smith Normal Matrices and Linear Diophantine Equations. Mathematics of Computation 25(116), 897–907 (1971)
7. Cook, S.A.: The complexity of theorem proving procedures. In: Proceedings of the 3rd ACM STOC, pp. 151–158 (1971)
8. Gebauer, H., Szabó, T., Tardos, G.: The local lemma is tight for SAT. CoRR (Computing Research Repository), arXiv:1006.0744 (2010)
9. Georgiou, K., Papakonstantinou, P.A.: Complexity and algorithms for well-structured k-SAT instances. In: Kleine Büning, H., Zhao, X. (eds.) SAT 2008. LNCS, vol. 4996, pp. 105–118. Springer, Heidelberg (2008)
10. Håstad, J.: Some optimal inapproximability results. J. ACM 48(4), 798–859 (2001)
11. Henschen, L., Wos, L.: Unit refutations and Horn sets. J. ACM 21(4), 590–605 (1974)
12. Kratochvíl, J., Savický, P., Tuza, Z.: One more occurrence of variables makes satisfiability jump from trivial to NP-complete. SIAM J. Comput. 22(1), 203–210 (1993)
13. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. 11(2), 329–343 (1982)
14. Monien, B., Sudborough, I.H.: Bandwidth constrained NP-complete problems. In: Proceedings of the 13th ACM STOC, pp. 207–217 (1981)
15. Rosen, K.H.: Elementary number theory and its applications, 5th edn. Addison-Wesley, Reading (2005)
16. Schaefer, T.J.: The complexity of satisfiability problems. In: Proceedings of the 10th ACM STOC, pp. 216–226 (1978)
17. Tovey, C.A.: A simplified satisfiability problem. Discrete Appl. Math. 8(1), 85–89 (1984)
18. Valiant, L.G.: The complexity of computing the permanent. Theoret. Comput. Sci. 8(2), 189–201 (1979)
19. Valiant, L.G.: The complexity of enumeration and reliability Problems. SIAM J. Comput. 8(3), 410–421 (1979)
20. Yamasaki, S., Doshita, S.: The satisfiability problem for the class consisting of Horn sentences and some non-Horn sentences in propositional logic. Infor. Control 59(1-3), 1–12 (1983)