Approximation Hardness and Satisfiability of Bounded Occurrence Instances of SAT

Piotr Berman * Marek Karpinski † Alex D. Scott ‡

Abstract

We study approximation hardness and satisfiability of bounded occurrence uniform instances of SAT. Among other things, we prove the inapproximability for SAT instances in which every clause has exactly 3 literals and each variable occurs exactly 4 times, and display an explicit approximation lower bound for this problem. We also provide a tighter characterization of the uniformly bounded occurrence instances which are surely satisfiable.

1 Introduction

We define a k-SAT instance as a set of clauses that are disjunctions of exactly k literals. The decision problem asks for an assignment of truth values to the variables that satisfies all the clauses. The maximimization problem is to find a truth assignment that satisfies maximally many clauses. Cook [C71] has shown that k-SAT is NP-complete for $k \geq 3$, and Papadimitriou and Yannakakis have shown that k-SAT is MAX-SNP-complete for $k \geq 2$.

In this paper we study uniform and regular instances of SAT. An instance of (k, s)-SAT is a formula in which every clause has length k, and each variable occurs exactly s times. Another (somewhat longer) notation used for that problem is EsOCC-EkSAT (cf. [K01], [BK03]).

In this paper we show that there exists a (3,4)-SAT formula with 15 variables and 20 clauses which is not satisfiable, and use this to prove that MAX-(3,4)-SAT

*Dept. of Computer Science and Engineering, The Pennsylvania State University. Research done in part while visiting Dept. of Computer Science, University of Bonn. Work partially supported by NSF grant CCR-9700053, NIH grant 9R01HG02238-12 and DFG grant Bo 56/157-1. E-mail berman@cse.psu.edu.

†Dept. of Computer Science, University of Bonn. Research done in part while vising Princeton University, and IHÉS Institute, Bures-sur-Yvette. Work partially supported by DFG grants, Max-Planck Research Prize, DIMACS, and IST grant 14036 (RAND-APX). Research at the Max-Planck Institute for Mathematics was supported by the Special Program on Computational Complexity. E-mail marek@cs.uni-bonn.de.

‡Dept. of Mathematics, University College London. E-mail scott@math.ucl.ac.uk

is not approximable to within some constant factor. We provide also an explicit approximation lower bound for that problem (Theorem 1). The number 4 is easily seen to be the smallest number of occurrences for which the problem is hard to approximate (by a direct bipartite perfect matching argument). The best previous result in this direction was the result of Feige [F98] to the effect that (3,5)-SAT is hard to approximate to within a certain constant. We provide similar results for other cases as well, e.g. for (4,6)-SAT and (5,9)-SAT.

We have learned later that Tovey [T84] also displayed an unsatisfiable (3,4)-SAT formula and proved that the decision version of (3,4)-SAT is NP-complete. Further progress was obtained by Dubois [D90] who has shown that (4,6)- and (5,11)-SAT are NP-complete. In turn, Kratochvíl *et. al.* [KST93] defined $f(k)$ as the largest s such that (k,s)-SAT has a satisfiable formula and have shown the following: (a) if $s > f(k)$ then (k,s)-SAT is NP-complete, (b) $f(k+1) \leq 2f(k)+1$ and (c) $f(k) \geq \lfloor 2^k/ek \rfloor$.

In this paper we improve the results of Dubois [D90] and Kratochvíl *et. al.* [KST93] as follows. We show that if $s > f(k)$ then (k,s)-SAT is MAX-SNP-complete. We also give improved bounds on $f(k)$ for small values of k, for instance showing that $f(5) < 9$ (so that $f(k) < 9 \times 2^{k-5}$ for $k > 4$), and that $f(6) \geq 7$.

2 Small Enforcers

We need to show how to force the Boolean value of a variable within the limitations of a (k,s)-SAT instance. This means providing a set of k-clauses that is satisfied if and only if x_i is true, and where x_i (x) occurs $s-1$ times—so it can be used once more—and where auxiliary variables occur at most s times. To achieve regularity, we may have to add some arbitrary clauses for the auxiliary variables. Since these clauses are arbitrary, we will only provide their number, which may be a fraction, e.g. a fractional number 1/3 means that when forcing values of 3 variables we need to create one arbitrary clause. We represent a set of clauses as an array, where each row lists literals of one of the clauses. By aligning the occurences of a variables we make it easy to count.

If we want to use a proof of MAX-SNP-completeness to find an unattainable approximation ratio for a problem, it is important to have as small size of the instance translation as possible, and by extension, to minimize the size of all the gadgets that collectively form a solution. Converting an unsatisfiable clause set into an enforcer is in practice wastful, because our examples of small unsatisfiable set are composed from several enforcers.

If a clause was obtained as an implication from a set of clauses, we annotate each literal with a superscript in parenthesis that indicates the number of occurrences of each variable in this set. From this perspective, an enforcer for (k,s)-SAT has the form of $x^{(i)}$ where $i < s$. If there exists such an enforcer, we can produce an unsatisfiable set as $x_1^{(i)}, ..., x_k^{(i)}$ and $\neg x_1 \vee ... \vee \neg x_k$. We represent a set of clauses as an array, where each row lists literals of one of the clauses. By aligning the occurences of variables we make it easy to count.

2.1 Enforcers in (3,4)-SAT

The enforcer given here is somewhat smaller than the one described by Tovey [T84]. It contains 19 clauses, rather than 22 in the construction of [T84].

We can force variable x to be true using the following six 3-clauses:

$$x^{(3)} \equiv \begin{array}{ccccc} x & a & \neg b & & \\ & a & b & \neg c & \\ & & b & c & \neg d \\ & & b & c & d \\ x & \neg a & & & \neg d \\ x & \neg a & & & d \end{array}$$

Because we need to add some arbitrary clause to have the fourth occurence of c, we altogether use 19/3 clauses to force x.

Three enforcers together with $\neg x_1 \vee \neg x_2 \vee \neg x_3$ create a nonsatisfiable formula with 20 clauses and 15 variables.

To make our conventions more clear, we will prove that the forcing indeed occurs. Suppose that x is false and all 6 of the above clauses are true. Because we can delete x from these clauses, we know that $\neg a \vee \neg d$ and $\neg a \vee d$, hence $\neg a$. Now we can delete a from the remaining clauses and the first clause yields $\neg b$. Now we can delete b and the second clause yields $\neg c$. Consequently, the third and fourth clauses yield $\neg d$ and d, a contradiction.

2.2 Enforcers in (4,6)-SAT

To define a set of clauses that forces x withing the constraints of (4,6)-SAT we define the notion of *supervised implication* which in actuality is a set of six 4-clauses:

$$x^{(1)} \rightarrow (\ y^{(2)} \rightarrow z^{(2)}\ ) \equiv \begin{array}{cccccccc} \neg x & \neg y & z & \neg a & & & & \\ & \neg y & z & a & \neg b & & & \\ & & & a & b & c & d & \\ & & & a & b & \neg c & d & \\ & & & a & b & & \neg d & e \\ & & & a & b & c & \neg d & \neg e \end{array}$$

Note that if we have two supervised implications, the same variable can play the role of c in both, and a variable can play the role of d in one and the role of e in the other. As a result, only b's have the deficit of occurences, and to remove this deficit we add one clause to every four supervised implications. Therefore one such implication requires 6.25 clauses.

Literal x forced to be true by 4-clauses:

$$x^{(5)} \quad \equiv \quad \begin{array}{rcl} \neg x^{(1)} & \rightarrow & (\ a^{(2)} \rightarrow b^{(2)}\ ) \\ \neg x^{(1)} & \rightarrow & (\ b^{(2)} \rightarrow c^{(2)}\ ) \\ \neg x^{(1)} & \rightarrow & (\ c^{(2)} \rightarrow a^{(2)}\ ) \\ \neg x & \rightarrow & (\ a \vee b \vee c\ ) \\ \neg x & \rightarrow & (\ \neg a \vee \neg b \vee \neg c\ ) \end{array}$$

One can see that we force x using 20.75 clauses.

2.3 Forcing a Variable in (5,9)-SAT

The set of clauses that forces a variable to be false withing the limitations of (5,9)-SAT has a complicated construction. We define the following auxiliary notion: $\mathcal{F}(ix, jy)$ is a set of clauses in which x occurs i times, y occurs j times and which cannot be satisfied if both x and y are false, but which is otherwise satisfiable. Let $f(ix, jy) = |\mathcal{F}(ix, jy)|$. We will use a similar notion for a single variable (a contradiction if it is false) or for three variables (a contradiction if they are all false).

Our goal is to define $\mathcal{F}(8x)$ and to find $f(8x)$.

$$\mathcal{F}(8x) = \{x \vee \neg y_0 \vee \neg y_1 \vee \neg y_2 \vee \neg y_3\} \cup \mathcal{F}(8y_0, 1x) \cup \bigcup_{i=1}^{3} \mathcal{F}(8y_i, 2x).$$

Indeed, if x is false, then the only way to avoid contradiction is if y_i is false for some $i = 0, 1, 2, 3$, but then either both x and y_0 are false, and we have a contradiction from $\mathcal{F}(8y_0, 1x)$, or both x and y_1 are false, and we have a contradiction from $\mathcal{F}(8y_1, 2x)$ etc. Conversely, if y is true, both the initial clause and each of the 4 clause sets used in the definition is satisfiable.

We define $\mathcal{F}(8x, 1z)$ as

$$\mathcal{F}(8x, 1z) = \{x \vee z \vee \neg y_0 \vee \neg y_1 \vee \neg y_2\} \cup \mathcal{F}(8y_0, 2x) \cup \mathcal{F}(8y_1, 2x) \cup \mathcal{F}(8y_2, 3x).$$

We define $\mathcal{F}(8x, 2z)$ as

$$\mathcal{F}(8x, 2z) = \{x \vee z \vee \neg y_0 \vee \neg y_1 \vee \neg y_2\} \cup \mathcal{F}(8y_0, 3x) \cup \mathcal{F}(8y_1, 3x) \cup \mathcal{F}(8y_2, 1x, 1y).$$

We define $\mathcal{F}(8x, 1a, 1b)$ as

$$\mathcal{F}(8x, 1a, 1b) = \{x \vee a \vee b \vee \neg y_0 \vee \neg y_1\} \cup \mathcal{F}(8y_0, 3x) \cup \mathcal{F}(8y_1, 4x).$$

One can see that

$$f(8x) = 1 + f(8x, 1y) + 3f(8x, 2y) = 2 + 5f(8x, 2y) + f(8x, 3y) =$$
$$7 + 11f(8x, 3y) + 5f(8x, 1y, 1z) = 12 + 16f(8x, 3y) + 5f(8x, 4y).$$

To define $\mathcal{F}(8x, 3y)$ we need to define three types of *supervised implications*, i.e. special sets of clauses. A supervised implication of type I can be viewed in two ways, and it is realized by nine 5-clauses:

$$\begin{array}{rcl} w^{(1)} \wedge\ x^{(2)} & \rightarrow & (z^{(3)} \rightarrow y^{(2)}\ ) \\ w^{(1)} \rightarrow (\ z^{(3)} & \rightarrow & (x^{(2)} \rightarrow y^{(2)}\ )) \end{array} \quad \equiv \quad \begin{array}{ccccccccc} \neg w & \neg x & y & \neg z & \neg a & & & & \\ & \neg x & y & \neg z & a & \neg b & & & \\ & & & \neg z & a & b & \neg c & \neg d & \\ & & & & a & b & \neg c & d & \neg e \\ & & & & a & b & c & \neg d & \neg e \\ & & & & a & b & c & d & \neg e \\ & & & & a & b & \neg c & d & e \\ & & & & a & b & c & \neg d & e \\ & & & & a & b & c & d & e \end{array}$$

A supervised implication of type II can be realized with two supervised implications of type I, and thus we use 18 5-clauses:

$$x^{(2)} \rightarrow (\ y^{(4)} \rightarrow z^{(4)}\ ) \qquad \equiv \qquad \begin{array}{lcl} x^{(1)} & \rightarrow & (\ z^{(3)} \rightarrow (x^{(2)} \rightarrow y^{(2)}\ ) \\ x^{(1)} & \rightarrow & (\ \neg z^{(3)} \rightarrow (x^{(2)} \rightarrow y^{(2)}\ ) \end{array}$$

Type III is realized with one type II and two type I and one normal clause, thus we use 37 5-clauses:

$$s^{(7)} \wedge t^{(3)} \rightarrow u_1^{(7)} \wedge u_2^{(7)} \wedge u_3^{(7)} \quad \equiv \quad \begin{array}{rcl} s^{(2)} & \rightarrow & (u_1^{(4)} \rightarrow u_2^{(4)}\ ) \\ s^{(2)} \wedge t^{(1)} & \rightarrow & (u_2^{(2)} \rightarrow u_3^{(3)}\ ) \\ s^{(2)} \wedge t^{(1)} & \rightarrow & (u_3^{(3)} \rightarrow u_1^{(2)}\ ) \\ s \wedge t & \rightarrow & (u_1 \vee u_2 \vee u_2) \end{array}$$

Finally, $\mathcal{F}(8x, 3y)$ is formed from two type III implications and three normal clauses, for the total of $f(8x, 3y) = 77$ 5-clauses:

$$\mathcal{F}(8x,3y) \equiv x^{(8)} \wedge y^{(3)} \equiv \begin{array}{rcl} \neg x^{(7)} \wedge \neg t^{(3)} & \rightarrow & a_1^{(7)} \wedge a_2^{(7)} \wedge a_3^{(7)} \\ \neg x \wedge a_1 \wedge a_2 \wedge a_3 & \rightarrow & b \\ b \wedge a_1 \wedge a_2 \wedge a_3 & \rightarrow & c \\ c^{(7)} \wedge b^{(3)} & \rightarrow & d_1^{(7)} \wedge d_2^{(7)} \wedge d_3^{(7)} \\ c \wedge b & \rightarrow & \neg(d_1 \wedge d_2 \wedge d_3) \end{array}$$

One can use the above idea to get $f(8x, 4y) = 39$. This leads to $f(8x) = 12 + 16 \times 77 + 5 \times 39 = 1439$. We also need to add some extra clauses to increase the number of occurences of some auxiliary variables to 9.

3 NP-completeness

Suppose that we can force a variable to be true withing constraints of (k, s)-SAT, i.e. there exists a set $\mathcal{F}(x)$ of clauses of length k in which each variable occurs at most s times, x occurs at most $s-1$ times and such that all clauses in $\mathcal{F}(x)$ can be satisified if and only if x is false. Then we can construct an unsatisfiable formula of (k, s)-SAT: start with a clause $x_1 \vee \ldots \vee x_k$ and for each $i = 1, \ldots, k$ we add set of clauses $\mathcal{F}(x_i)$; we need to take care that if $i \neq j$ then $\mathcal{F}(x_i)$ and $\mathcal{F}(9x_j)$ have no variables in common. If some of the variables occurs less than s times, we can add extra clauses.

We can use the same approach to translate 3-SAT into (k, s)-SAT. First, we reduce the number of occurences of each variables to 3. Then we replace a clause c with $c \vee x_4 \vee x_k$, using different variables for every clause. Next, we add $\mathcal{F}(x)$ for each new variable. Finally, we add extra clauses to assure that each variable occurs exactly s times. With little care, these extra clauses can be easy to satisfy.

We can conclude that if in (k, s)-SAT we can force a Boolean value of a variable, then (k, s)-SAT is NP-complete.

4 MAX-SNP Hardness

Given an instance of MAX-2-SAT, we can increase the length of clauses by inserting literals that are forced to be false. In this manner we can create a MAX-SAT instance with the same minimum number of unsatisfied clauses in which all clauses have a desired length. Towards the inapproximability result, we start from instances constructed in [BK03]. In that paper, in order to show the hardness of approximating MAX-2-SAT restricted to a certain class of instances, one have to construct the classes of clause sets with the following properties:

1. It is NP-hard to distinguish on whether the minimum number of unsatisfied clauses is at least $(5 - \varepsilon)k$ or at most $(4 + \varepsilon)k$;

2. all variables occur the same number of times except for those that occur in $4k$ copies of a gadget called *replacements of equations of a single auxiliary variable* (Fig. 6 in [BK03]), here we will call this gadget RESAV;

In Theorem 12 of [BK03] the size of these class sets is given as $256k$ for the case when each variable occurs at most 4 times, and RESAV contains one clase with only one literal and one variable with only 3 occurences. This proves the following lemma.

Lemma 1. *There exists a family of sets of 2-clauses such that each, for some n, consists of $252n$ 2-clauses and $4n$ 1-clauses for which it is NP-hard to distinguish between the systems where $(25n - \varepsilon)n$ clauses can be satisfied and systems where at most $(251 + \varepsilon)n$ clauses can be satisfied (for $\varepsilon < 1/2$). Moreover, $4n$ variables in this system occur 3 times, while other variables have exactly 4 occurences.*

We can transform a system from this theorem into an instance of Max-(3,4)-SAT in two stages. First, we increase the length of all clauses to 3 by inserting $260n$ variables that are forced to be false. Forcing them to be force requires $(260 \times 19/3 = 1646 + 2/3)n$ 3-clauses. Second, we need to increase the number of occurences of $4n$ variables using $4/3\ n$ clauses. Thus we increased the number of clauses in the system from $256n$ to $1904n$ and we can conclude that

Theorem 1. *There exists a family of instances of Max-(3,4)-SAT such that each, for some n, consists of $1904n$ clauses, for which it is NP-hard to distinguish between the systems where $(1900 - \varepsilon)n$ clauses can be satisfied and systems where at most $(1899 + \varepsilon)n$ clauses can be satisfied (for $\varepsilon < 1/2$).*

This entails the following corollary.

Corollary 1. *It is NP-hard to approximate MAX-(3,4)-SAT to within any factor below 1.00052.*

Moreover, Theorem 12 of [BK03] shows that the size of the difficult clause sets for the case when each variable occurs at most 6 times is 168, and RESAV contains two variables with only 5 occurances. This shows the following.

Lemma 2. *There exists a family of sets of 2-clauses such that each, for some n, consists of $168n$ 2-clauses for which it is NP-hard to distinguish between the systems where $(164-\varepsilon)n$ clauses can be satisfied and systems where at most $(163+\varepsilon)n$ clauses can be satisfied (for $\varepsilon < 1/2$). Moreover, $8n$ variables in this system occur 5 times, while other variables have exactly 6 occurences.*

We can transform a system from this theorem into an instance of Max-(4,6)-SAT in two stages. First, we increase the length of all clauses to 4 by inserting $336n$ variables that are forced to be false. Forcing them to be false requires $336 \times 20.75 = 6972$ 3-clauses. Second, we need to increase the number of occurences of $8n$ variables using $2n$ clauses. Thus we increased the number of clauses in the system from $168n$ to $7144n$ and we can conclude that

Theorem 2. *There exists a family of instances of Max-(4,6)-SAT such that each, for some n, consists of $7144n$ clauses, for which it is NP-hard to distinguish between the systems where $(7140 - \varepsilon)n$ clauses can be satisfied and systems where at most $(7139 + \varepsilon)n$ clauses can be satisfied (for $\varepsilon < 1/2$).*

In the same fashion we can show that Max-(5,9)-SAT is also MAX-SNP-hard, but the calculation of the provable inapproximability ratio is a bit difficult. A system of 2-clauses in which there is a difficult gap of almost n clauses would have about $100n$ clauses, and thus it would require an insertion of about $300n$ variables forced to be false—so we increase the clause length to 5. For each of the $300n$ variables we need roughly 1500 clauses that force it, and thus we get a system with ca. $400,000n$ clauses. (It may be possible to decrease this estimate by a factor of 2.)

Note that if there exists an unsatisfiable formula of (k, s)-SAT, then there exists a system in which some variable x occurs less then s times and which can be satisfied if and only if x is false. Such a system can be used to define an Eq-reduction that transform $(2, s)$-SAT instances into instances of (k, s)-SAT, using the same approach as in the last two theorems. Therefore we can conclude that

Theorem 3. *If $s > f(k)$ then (k, s)-SAT is MAX-SNP-complete.*

5 Satisfiability of Small Occurrence Instances

We now turn to lower bounds on $f(k)$, the largest integer s such that every instance of (k, s)-SAT is satisfiable. Lower bounds on $f(k)$ were given by Kratochvíl, Savický and Tuza [KST93], who used the Lovász Local Lemma to show that $f(k) \geq \lfloor 2^k/ek \rfloor$; they also showed $f(i) = i$ for $i \leq 3$ and $4 \leq f(4) \leq 6$.

The Lovász Local Lemma has also been used in the study of the related problem of hypergraph colouring; indeed this was one of its original applications (see [EL75], [AS00], [B01], [MR02]). A hypergraph $H = (V, E)$ is *2-colourable* if there is a partition $V = V_1 \cup V_2$ such that each edge contains vertices from both V_1 and V_2. For $k \geq 2$, let $g(k)$ be the largest integer such that every k-uniform hypergraph in which every vertex is contained in at most $g(k)$ edges is 2-colourable. It follows from a result of Seymour [S74] that $g(k) \geq k - 1$ for every k, and after significant work it is now known when $g(k) \geq k$: the Local Lemma shows $g(k) \geq k$ for $k \geq 9$, Alon and Bregman [AB88] showed that $g(k) \geq k$ for $k \geq 8$, and finally Thomassen [T92] showed $g(k) \geq k$ for $k \geq 4$; while the Fano plane shows that $g(3) < 3$. More recently, McDiarmid [M97] showed that it is possible to improve on bounds for $g(k)$ obtained from the Local Lemma by using the 'lopsided' version of the Local Lemma given by Erdős and Spencer [ES91]. In this section we use a similar approach to obtain bounds on $f(k)$.

Let G be a graph with vertex set X, and let $\{A_x\}_{x\in X}$ be a collection of events in some probability space. We say that G is a *lopsidependency graph* for $\{A_x\}_{x\in X}$ if, for each $x \in X$, and each subset $Y \subset X \setminus \Gamma^*(x)$, we have

$$\mathbb{P}(A_x | \bigwedge_{y\in Y} \overline{A}_y\} \leq \mathbb{P}(A_x).$$

Here we write $\Gamma(x)$ for the set of neighbours of x and set $\Gamma^*(x) = \{x\} \cup \Gamma(x)$. We can now state the Lopsided Local Lemma.

Theorem 4. *[ES91] Suppose that $G = (X, E)$ is a lopsidependency graph for a collection of events $(A_x)_{x\in X}$. If there are real numbers $(q_x)_{x\in X}$ such that, for each $x \in X$,*

$$\mathbb{P}(A_x) \leq q_x \prod_{y\in\Gamma(x)} (1 - q_y)$$

then $\mathbb{P}(\bigwedge_{x\in X} \overline{A}_x) > 0$.

Some notation: for a clause C, we write C^+ for the set of variables that occur in C without negation, and C^- for the set of variables that appear negated. We write $V(C) = C + \cup C^-$ for the set of all variables that occur in C.

Theorem 5. *Suppose $d \geq k \geq 1$ are positive integers and there is $x \in (0, 1)$ such that*

$$x^{1/k}((1 - x)^{\lfloor d/2\rfloor} + (1 - x)^{\lceil d/2\rceil}) > 1. \tag{1}$$

Then every instance of (k, d)-SAT has a satisfying assignment.

Proof. Let $\mathcal{C}$ be the set of clauses in an instance of (k, d)-SAT, and let X be the set of variables occurring in clauses from $\mathcal{C}$. We define a graph G with vertex set $\mathcal{C}$ and an edge between clauses C and C' if and only if there is a variable x that occurs negated in one clause and without negation in the other. Given a sequence $(p_i)_{i\in X}$ of reals $p_i \in [0, 1]$, we generate a random assigment for X by setting each variable $i \in X$ true independently with probability p_i and false otherwise. Note that then

$$\mathbb{P}(C \text{ not satisfied}) = \prod_{i\in C^-} p_i \prod_{\in C^+} (1 - p_i).$$

For $C \in \mathcal{C}$, let A_C be the event that C is not satisfied. We first show that G is a lopsidependency graph for the events $(A_C)_{C\in\mathcal{C}}$. Indeed, let C be a clause in $\mathcal{C}$ and let $\{C_j : j \in J\}$ be a set of clauses that does not contain C or any of its neighbours. Let B be the event that C_j is satisfied for every $j \in J$. Let X_0 be the set of variables that occur both in C and in some C_j: changing the sign of all occurrences of some of these variables if necessary, we may assume that all variables in X_0 occur without negation in C. By the definition of G, it follows that these variables do not occur negated in any of the C_j. Thus $\overline{A}_C$ and B are both increasing in the variables in X_0. Since A and B are independent if we condition on the assignment restricted to X_0, it follows from the FKG inequality [FKG71], or the inequality of Harris [H60], that $\overline{A}$ and B are positively correlated, and hence A and B are negatively correlated, which is the condition required for a lopsidependency graph.

Let $x \in (0,1)$ satisfy (1). We will choose probabilities $(p_i)_{i\in X}$ so that, for every clause C,

$$\prod_{i\in C^-} p_i \prod_{i\in C^+} (1-p_i) < x(1-x)^{d(C)},$$

where $d(C)$ is the degree of C in G. Now it follows from (1) that, for $0 \leq i \leq k$, we have

$$x^{1/k}((1-x)^i + (1-x)^{d-i}) > 1$$

(note that $(1-x)^i + (1-x)^{d-i}$ is minimized for integers in this range when $i = \lceil d/2 \rceil$ or $\lfloor d/2 \rfloor$). Thus we can choose numbers $\eta_0, \ldots, \eta_d \in (0,1)$ such that, for $0 \leq i \leq d$,

$$\eta_i < x^{1/k}(1-x)^{d-i}$$

and

$$\eta_i + \eta_{d-i} = 1.$$

For each $i \in X$, let $pos(i)$ be the number of positive occurrences of i in clauses of $\mathcal{C}$. We define $p_i = \eta_{pos(i)}$.

For a clause C and a variable $a \in V(C)$, we define $d_C(a)$ as follows: if a occurs without negation in C then $d_C(a)$ is the number of negated occurrences of a in clauses of $\mathcal{C}$; if $\neg a$ occurs in C then $d_C(a)$ is the number of non-negated occurrences of a in clauses of $\mathcal{C}$. Since C is adjacent to a clause C' exactly when there is a variable that occurs negated in one clause and without negation in the other, we have $d(C) \leq \sum_{a\in V(C)} d_C(a)$. Now if a is not negated in C then $p_a = \eta_{d-d_C(a)}$, while if a is negated then $p_a = \eta_{d_C(a)}$. So

$$\begin{aligned}
\mathbb{P}(C \text{ unsatisfied}) &= \prod_{i\in C^-} p_i \prod_{i\in C^+} (1-p_i) = \prod_{i\in C^-} \eta_{d-d_C(a)} \prod_{i\in C^+} (1-\eta_{d_C(a)}) \\
&= \prod_{i\in V(C)} \eta_{d-d_C(a)} < \prod_{i\in V(C)} x^{1/k}(1-x)^{d_C(a)} \\
&\leq x(1-x)^{d(C)}.
\end{aligned}$$

It therefore follows from Theorem 4 that, with positive probability, all clauses of $\mathcal{C}$ are satisfied and, in particular, it follows that $\mathcal{C}$ is satisfiable. $\square$

We can now prove our lower bounds on $f(k)$.

Corollary 2. *All instances of* $(6,7)$*-SAT,* $(7,13)$*-SAT,* $(8,23)$*-SAT and* $(9,41)$*-SAT are satisfiable.*

Proof. Apply Theorem 5 with $x = 1/21$, $x = 1/47$, $x = 1/100$ and $x = 1/200$ respectively. $\square$

Dubois [D90] showed that if every instance of (r,s)-SAT is satisfiable then every instance of $(r+1, s+\lfloor s/r \rfloor)$-SAT is satisfiable. Corollary 2 therefore implies the following.

Corollary 3. *Every instance of* $(k,k+1)$*-SAT is satisfiable for* $k \geq 6$*.*

We remark that Theorem 5 is not enough to show that every instance of $(5,6)$-SAT is satisfiable. Indeed, considering an instance in which each variable occurs three times without negation and three times with negation, we see that we would need some $x \in (0,1)$ such that $2x^{1/5}(1-x)^3 > 1$, but the left hand side has maximum less than 0.95. However, we conjecture the following.

Conjecture 1. *Every instance of* $(5,6)$*-SAT is satisfiable.*

Theorem 5 also gives an asymptotic lower bound on $f(k)$ that is slightly better than that of Kratochvíl, Savický and Tuza [KST93]. However, it should be possible to obtain better bounds by employing techniques from hypergraph colouring (see, for instance, Radhakrishnan and Srinivasan [RS00]).

Finally, we remark that the arguments above show only the existence of a solution for instances of (r,s)-SAT with $s \leq f(r)$. Beck [B91] gave an algorithmic version of the local lemma, which provides that a randomized algorithm finding a satisfying assignment with a large probability, but his proof requires a much smaller value of s. For fixed r, and $s \leq f(r)$, is there a polynomial-time algorithm that finds a satisfying assignment for all instances of (r,s)-SAT?

Acknowledgments.

We thank Johan Håstad for drawing our attention to the problem of approximating (3,4)-SAT and Jörg Rothe for informing us about previous results on that problem.

References

[AS00] N. Alon and J. Spencer, *The Probabilistic Method*, 2nd ed. (Wiley, New York, 2000)

[AB88] N. Alon and Z. Bregman, *Every 8-uniform 8-regular hypergraph is 2-colourable*, Graphs Combin. **4**:303–305, 1988

[A95] S. Arora, C. Lund, *Hardness of Approximations*, in *Approximation Algorithms*, D. S. Hochbaum (ed.), PWS Publishing, Boston 1995, 399-446

[B91] J. Beck, *An algorithmic approach to the Lovász local lemma I*, in *Random Structures and Algorithms* **2**:343-365, 1991

[BK03] P. Berman and M. Karpinski, *Improved Approximation Lower Bounds on Small Occurrence Optimization*, Elec. Coll. on Comp. Compl., Report 8 (2003)

[B01] B. Bollobás, Random graphs, Second edition, Cambridge Studies in Advanced Mathematics 73, Cambridge University Press, Cambridge, 2001, xviii+498 pp.

[C71] S.C. Cook, *The complexity of theorem-proving procedures*, Proc. 3rd ACM STOC, 151-158, 1971

[D90] O. Dubois, *On the r,s-SAT satisfiability problem and a conjecture of Tovey*, Discrete Applied Math. **26**:51-60, 1990

[EL75] P. Erdős and L. Lovász, Problems and results on 3-chromatic hypergraphs and some related questions, *Infinite and finite sets*, Vol. II, Colloq. Math. Soc. Janos Bolyai, Vol. 10, North-Holland, Amsterdam, 1975, pp. 609–627

[ES91] P. Erdős and J. Spencer, *Lopsided Lovász local lemma and latin transversals*, Discrete Applied Math. **30**:151–154, 1991

[F98] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, *Journal of ACM*, **45(4)**:634–652, July 1998

[FKG71] C.M. Fortuin, P.W. Kasteleyn and J. Ginibre, *Correlation inequalities on some partially ordered sets*, Communications in Mathematical Physics **22**:89–103, 1971

[H97] J. Håstad, *Some optimal inapproximability results*, Proc. 29th ACM STOC, 1-10, 1997

[H00] J. Håstad, *On bounded occurrence constraint satisfaction*, Information Processing Letters **74**:1-6, 2000

[H60] T.E. Harris, *A lower bound for the critical probability in a certain percolation process*, Proceedings of the Cambridge Philosophical Society **56**:13–20, 1960

[K01] M. Karpinski, *Approximating bounded degree instances of NP-hard problems,* Proc. 13th Symp. on Fundamentals of Computation Theory, LNCS 2138, Springer, 2001, 24-34

[KST93] J. Kratochvíl, P. Savický and Z. Tuza, *One more occurence of variable makes satisfiability jump from trivial to NP-complete*, SIAM J. Comput. **8**:203-210, 1993

[M97] C. McDiarmid, *Hypergraph colouring and the Lovász Local Lemma*, Discrete Math. **167/168**:481–486, 1997

[MR02] M. Molloy and B. Reed, Graph colouring and the probabilistic method, Algorithms and Combinatorics 23, Springer-Verlag, Berlin, 2002, xiv+326 pp

[PY91] C. Papadimitriou and M. Yannakakis, *Optimization, approximation and complexity classes*, JCSS **43**:425-440, 1991

[P94] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, New York, 1994, 315-319

[RS00] J. Radhakrishnan and A. Srinivasan, *Improved bounds and algorithms for hypergraph 2-colouring*, Rand. Structures and Algorithms **16**:4–32, 2000

[S74] P.D. Seymour, *On the two colouring of hypergraphs*, Quart. J. Math. Oxford Series 3 **25**:303–312, 1974

[T92] C. Thomassen, *The even cycle problem for directed graphs*, J. Amer. Math. Soc. **5**:217–229, 1992

[T84] C.A. Tovey, *A simplified satisfiability problem*, Discrete Applied Math. **8**:85-89, 1984.

[T01] L. Trevisan, *Non-approximability results for optimization problems on bounded degree instances*, Proc. 33rd ACM STOC, 2001