Testing Satisfiability

Noga Alon * Asaf Shapira †

Abstract

Let Φ be a set of general boolean functions on n variables, such that each function depends on exactly k variables, and each variable can take a value from $[1, d]$. We say that Φ is *ϵ-far* from satisfiable, if one must remove at least ϵn^k functions in order to make the set of remaining functions satisfiable. Our main result is that if Φ is *ϵ-far* from satisfiable, then most of the induced sets of functions, on sets of variables of size $c(k, d)/\epsilon^2$, are not satisfiable, where $c(k, d)$ depends only on k and d. Using the above claim, we obtain similar results for k-SAT and k-NAEQ-SAT.

Assume we relax the decision problem of whether an instance of one of the above mentioned problems is satisfiable or not, to the problem of deciding whether an instance is satisfiable or *ϵ-far* from satisfiable. While the above decision problems are NP-hard, our result implies that we can solve their relaxed versions, that is, distinguishing between satisfiable and *ϵ-far* from satisfiable instances, in randomized *constant time*.

From the above result we obtain as a special case, previous results of Alon and Krivelevich [3] and of Czumaj and Sohler [8], concerning testing of graphs and hypergraphs colorability. We also discuss the problem of testing whether a graph G can be d-colored, such that it does not contain any copy of a colored graph from a fixed, given set of colored graphs.

1 Introduction.

A set of boolean functions on n variables is *satisfiable*, if there is an assignment to the n variables, that *simultaneously* satisfies all the functions in the set. For fixed integers k and d, and a small $\epsilon > 0$, let $\Phi = (V, F)$ be a set of boolean functions on n variables, where V is the set of variables, and F is the set of functions, and where each function depends on exactly k variables, and each variable can take a value from $[1, d]$. Suppose that at least ϵn^k functions should be deleted to make Φ satisfiable. Clearly Φ contains many non-satisfiable subformulas, some of which may be quite small in order. A natural question is: how many small unsatisfiable subformulas are there in Φ ? In what follows we call such boolean functions $f : d^k \to \{0, 1\}$, *(k,d)-functions*, and a set Φ of such functions, a *(k,d)-Function-SAT* instance.

In order to address the above question quantitatively, let us introduce a suitable notation. First, we call an instance of (k,d)-Function-SAT, Φ, on n variables *ϵ-far from satisfiable*, if after deleting any subset of less than ϵn^k functions from Φ, the remaining set of functions is still not satisfiable. Of course, it follows that Φ itself is not satisfiable. Further, given a subset of variables $R \subseteq V$ we denote by $\Phi[R] = (R, F_{|R})$ the set of functions that have all their variables in R. Let $SAT_{k,d}(n, \epsilon)$ denote the minimal value, such that for every instance of (k,d)-Function-SAT on n variables Φ, that is ϵ-far from satisfiable, if we pick a random subset of variables R, of size $SAT_{k,d}(n, \epsilon)$, then $\Phi[R]$ is not satisfiable with probability at least $3/4$.

We can define a similar measure for the problem of d-coloring a k-uniform hypergraph. Given a hypergraph $H = (V, E)$ and a subset of vertices $R \subseteq V$ we denote by $H[R] = (R, E_{|R})$, the hypergraph that has R as its set of vertices, and all the edges in E that have all their vertices in R, as its set of edges. A k-uniform hypergraph H is *ϵ-far* from d-colorable, if after removing any subset of edges of size less than ϵn^k, H is still not d-colorable. Now, Let $COL_{k,d}(n, \epsilon)$ denote the minimal value, such that for every k-uniform hypergraph on n vertices H, that is ϵ-far from d-colorable, if we pick a random subset of vertices R of size $COL_{k,d}(n, \epsilon)$, then $H[R]$ is not d-colorable with probability at least $3/4$. We can further define the corresponding functions for the problems k-CNF, and k-

*Schools of Mathematics and Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Email: noga@math.tau.ac.il. Research supported in part by a USA-Israeli BSF grant, by the Israel Science Foundation and by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University.

†School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Email: asafico@math.tau.ac.il.

NAEQ-CNF, as $CNF_k(n,\epsilon)$ and $NAEQ\text{-}CNF_k(n,\epsilon)$, respectively. Note that in these functions, the variables can take boolean values, thus d is fixed to be 2. The precise (obvious) definitions of these problems appear in the appendix.

Therefore, $SAT_{k,d}(n,\epsilon)$ means that if Φ is a set of (k,d)-functions, that is ϵ-*far* from satisfiable, then most of the induced sets of functions, on subsets of variables of size $SAT_{k,d}(n,\epsilon)$, are not satisfiable. Intuitively, it means that inside Φ there are many proofs of size $SAT_{k,d}(n,\epsilon)$ showing that Φ is not satisfiable, where a proof is in the form of a subset $F' \subset F$ of functions, that can not all be satisfied simultaneously. The analog intuition holds for the function $COL_{k,d}(n,\epsilon)$.

The somewhat artificially looking definition of $SAT_{k,d}(n,\epsilon)$ has a very natural algorithmic background in terms of *property testing*. Suppose our aim is to design an algorithm, which for a given (large enough) integer n and a (small enough) parameter $\epsilon > 0$, distinguishes with high probability between an input (k,d)-Function-SAT instance on n vertices, which is satisfiable, and one which is ϵ-far from satisfiable. The algorithm can query whether a specific function on k variables is in the instance. We call the problem of distinguishing between these two cases, *testing satisfiability*, and an algorithm for this problem is an ϵ-*tester*. In general, it is NP-complete to check satisfiability. However, given the assumption that the input is either satisfiable or ϵ-far from being such, one may hope to devise very efficient randomized algorithms, that distinguish between these two possibilities. We refer the reader to [11] for a discussion on general property testing, and in particular, on graph property testing, and to the comprehensive survey of Ron [15] on the field of property testing.

Returning to the definition of the function $SAT_{k,d}(n,\epsilon)$, we can propose the following very simple algorithm for testing satisfiability. Given an input formula $\Phi = (V,F)$, choose uniformly at random $SAT_{k,d}(n,\epsilon)$ variables of Φ and denote the chosen set by R. Now, check whether the induced subformula $\Phi[R]$ is satisfiable. If the induced set is satisfiable, output "Φ is satisfiable", otherwise output "Φ is not satisfiable". To argue that the above algorithm provides a correct answer with probability at least 3/4, note that if Φ is satisfiable, then every subformula of it is satisfiable as well. Thus, in this case we *always* output a correct answer. On the other hand, if Φ is ϵ-far from satisfiable, it follows from the definition of $SAT_{k,d}(n,\epsilon)$ that a sample of size $SAT_{k,d}(n,\epsilon)$ induces a non-satisfiable set of functions with probability at least 3/4. Therefore, in this case we output a correct answer with probability at least 3/4. As is common in randomized algorithms, by repeating the algorithm an appropriate constant number of times, the constant 3/4 can be replaced by any desired constant $\alpha < 1$.

As we show in this paper, the function $SAT_{k,d}(n,\epsilon)$ can be bounded from above by a function of k, d and ϵ. Thus, for example, for the problem of k-CNF with fixed k, which is a special case of (k,2)-Function-SAT, we get that in order to distinguish between a satisfiable instance of k-CNF, and one that is ϵ-*far* from satisfiable, all we need to do is sample a subset of variables of size $O(1/\epsilon^2)$, and check if the induced k-CNF instance on this set is satisfiable. Therefore, for a fixed ϵ we have a *constant time* one-sided error randomized algorithm for distinguishing between satisfiable and ϵ-*far* from satisfiable instances. Note that this implies that for dense instances of k-CNF, i.e. instances that contain $\Omega(n^k)$ functions, we have a constant time algorithm that distinguishes between satisfiable instances, and those in which at most an $1-\epsilon$ fraction of the *instance's functions* can be satisfied. The same algorithmic aspect also holds for the rest of the testing problems we discuss in the paper.

Having in mind the above discussion, sometimes later in the paper we will refer to the problem of bounding the function $SAT_{k,d}(n,\epsilon)$ as the *testing (k,d)-Function-SAT* problem.

The problem of estimating $SAT_{k,d}(n,\epsilon)$ will be treated in this paper as an *asymptotic* one. This means that whenever needed, we will assume the number of variables n to be large enough, and the parameter ϵ to be small enough. It is important to observe that we are interested here only in formulas with $\Omega(n^k)$ functions. Indeed, if Φ is a set of functions on n variables, that is ϵ-far from satisfiable, it contains at least ϵn^k functions. It is also important to note that one can not hope to design a randomized polynomial algorithm, that distinguishes between instances that are satisfiable, and those in which an ϵ fraction of the functions of the instance (and not an ϵ fraction of *all* the functions) should be deleted in order to make them satisfiable, as this problem is known to be NP-hard already in the case of 3-CNF for any $\epsilon < 1/8$, see Håstad [12].

1.1 Context and previous results. The *Satisfiability* problem is clearly one of the most studied prob-

lems in theoretical computer science, and the related research is too rich to survey in this introduction. In the early 70's, Cook [7] was the first to show that the 3-CNF problem, which is a special case of the general satisfiability problem, is NP-Complete. It is also known that the problem of 3-NAEQ-CNF, is NP-Complete, see [10]. The problem of how well can one approximate the fraction of functions that can be simultaneously satisfied, has been an open problem for many years. Håstad [12] showed that for the case of 3-CNF, it is NP-hard to approximate the fraction of clauses that can be satisfied, to within $7/8+\epsilon$, for any $\epsilon > 0$. Zwick and Karloff [13] gave a 7/8 approximation algorithm for the 3-CNF problem, showing that the constant 7/8 is tight in the case of 3-CNF.

To the best of our knowledge the problem of testing satisfiability with one sided error has never been addressed in the past. The previous results relevant to testing satisfiability, are a *two-sided error* property-tester given by Andersson and Engebretsen [5] that uses a random set of variables of size $\tilde{O}(1/\epsilon^5)$ (in the full version of the paper we discuss the fundamental difference between testing with one-sided and two-sided error), and a one-sided (implicit) property tester by Frieze and Kannan [9] which deals only with the case of $d = 2$ and uses a random set of variables of size exponential in $1/\epsilon$. Recent work in progress on the Max-Cut problem by de-la Vega, Kannan and Karpinsky suggests that it might be possible to design a one-sided error property tester that will also estimate how *far* from satisfiable a given instance is, by an appropriate combination of the techniques in this new paper and in [9]. While this approach might give an alternative proof to some of our results here, it will certainly not be as efficient if one is only interested in deciding whether an instance is satisfiable or ϵ-far from satisfiable.

The more relevant results to our investigation here, are those of testing graph and k-uniform hypergraph d-colorability with one-sided error, that is, estimating the functions $COL_{2,d}(n,\epsilon)$ and $COL_{k,d}(n,\epsilon)$. It turned out quite early, that the function $COL_{2,d}(n,\epsilon)$ can be bounded from above by a function of ϵ and d only. This has been proven (implicitly) by Bollobás, Erdős, Simonovits and Szemerédi [6] for the case $d = 2$ and by Rödl and Duke [14] for every $d \geq 3$, see also [2]. All these papers rely on the Regularity Lemma of Szemerédi [16], and as is the case with most applications of the Regularity Lemma, the resulting bounds are extremely fast growing functions of $1/\epsilon$ (towers of height polynomial in $1/\epsilon$).

Motivated by testing d-colorability, Goldreich, Goldwasser and Ron [11] came up with a completely different approach for bounding $COL_{2,d}(n,\epsilon)$. Using direct combinatorial arguments (and thus avoiding the Regularity Lemma), they were able to prove that $COL_{2,2}(n,\epsilon) = O(\log(1/\epsilon)/\epsilon^2)$ (note that $COL_{2,2}$ corresponds to testing bipartiteness of ordinary graphs), and that for every fixed $d \geq 3$ one has $COL_{2,d}(n,\epsilon) = O(d^2 \log d/\epsilon^3)$, a tremendous progress compared to the bounds of [6] and [14]. Improving on these results, Alon and Krivelevich [3] proved that $COL_{2,d}(n,\epsilon) = O(d \log d/\epsilon^2)$ and that $COL_{2,2}(n,\epsilon) = O(\log^4(1/\epsilon)\log\log(1/\epsilon)/\epsilon)$. Independent of the work of [3], Czumaj and Sohler [8] gave a more general result showing that $COL_{k,d}(n,\epsilon) = O(k^2 d^2 \log d/\epsilon^2)$.

1.2 The Main results. Consider the following generalization of k-CNF, which we denote by (k,d)-CNF. We are given a set of functions on n variables, where each variable v_i is restricted to take a value from $[1,d]$. Each function is of the form $(v_1 \neq c_1 \vee \ldots \vee v_k \neq c_k)$, and $1 \leq c_i \leq d$. We may write the clauses of a (k,d)-CNF instance, for short, as $(v_{1,c_1} \vee \ldots \vee v_{k,c_k})$. Note that k-CNF is just (k,2)-CNF, where each variable can take a value from $\{1,2\}$. Further notice that just as one can describe any boolean function whose variables take boolean values $f : \{0,1\}^k \to \{0,1\}$, as a set of k-CNF clauses, one can also describe any boolean function, whose variables take *non*-boolean values $g : [d]^k \to \{0,1\}$, as a set of (k,d)-CNF clauses. The formal (simple) argument for this intuitive fact will be described in section 3.

Denote by $f(k,d,\epsilon)$ the number of variables that suffice to test (k,d)-CNF. This is the analog of $SAT_{k,d}(n,\epsilon)$ for instances of (k,d)-CNF. We drop the parameter n, because as we will prove later, this function is independent of n. Our first result in the course of the paper is the following theorem

THEOREM 1.1.
For every fixed k and d, and sufficiently small $\epsilon > 0$,

$$f(k,d,\epsilon) \leq (100d^{k-1}\log d)/\epsilon^2.$$

Using this theorem we will get the following corollaries, which form the main part of this paper.

COROLLARY 1.1.
For every fixed k and d, every sufficiently small $\epsilon > 0$, and every sufficiently large n

$$SAT_{k,d}(n,\epsilon) \leq f(k,d,\epsilon/2^{d^k}) \leq (100d^{k-1}2^{2d^k}\log d)/\epsilon^2.$$

COROLLARY 1.2.

For every fixed k, every sufficiently small $\epsilon > 0$, and every sufficiently large n

$$CNF_k(n,\epsilon) \leq f(k,2,\epsilon) \leq (100 \cdot 2^{k-1})/\epsilon^2.$$

COROLLARY 1.3.

For every fixed k, every sufficiently small $\epsilon > 0$, and every sufficiently large n

$$NAEQ\text{-}CNF_k(n,\epsilon) \leq f(k,2,\epsilon) \leq (100 \cdot 2^{k-1})/\epsilon^2.$$

We also get the following version of the main result of Czumaj and Sohler [8]:

COROLLARY 1.4.

For every fixed k and d, every sufficiently small $\epsilon > 0$, and every sufficiently large n

$$COL_{k,d}(n,\epsilon) \leq f(k,d,\epsilon) \leq (100d^{k-1}\log d)/\epsilon^2.$$

In section 3, we explain how to slightly improve the dependence on k and d in this last result of [8].

Some of the proof techniques we employ in this paper are motivated by the d-colorability testing algorithm in Alon and Krivelevich [3], and by the d-colorability of k-uniform hypergraphs testing algorithm in Czumaj and Sohler [8], but the analysis here requires several additional arguments.

The rest of the paper is organized as follows: In section 2 we show how to test instances of (k,d)-CNF. In sections 3, we show how to derive the above corollaries, by reducing the problems to testing (k,d)-CNF satisfiability. In section 4 we give concluding remarks and discuss some open problems.

During the course of the proof we make no serious attempts to optimize the constants involved. Also, we omit routinely floor and ceiling signs to simplify the presentation. While proving upper bounds for all the functions, we sometimes generate a random subset R of size s as a union of random subsets whose sizes sum to s.

2 Testing (k,d)-CNF.

Let $\Phi = \Phi(V,C)$ be an instance of (k,d)-CNF, where V and C are respectively the sets of variables and clauses in Φ (we call the *functions* of a (k,d)-CNF instance *clauses*, because of their resemblance to clauses in k-CNF). Given a subset of variables $V_0 \subseteq V$, Let $\Phi[V_0]$ denote the induced formula on V_0. (Notice that V_0 is a set of variables, not literals). In this section we present the proof of Theorem 1.1, namely that testing (k,d)-CNF can be done by inspecting the induced subformula on a random subset of variables, of size at most $(100d^{k-1}\log d)/\epsilon^2$.

It will be convenient to generate a random subset of variables $R \subset V$ of size $|R| = s = (100d^{k-1}\log d)/\epsilon^2$ in s rounds, each time choosing uniformly at random a single variable $v_j \in V$. This in principle may result in choosing one variable several times and thus getting a set of cardinality less than s. We remind the reader that each clause is of the form $(v_{1,c_1} \vee \ldots \vee v_{k,c_k})$ where the literal v_{i,c_i} is short for $v_i \neq c_i$, and that an assignment ϕ satisfies a clause $(v_{1,c_1} \vee \ldots \vee v_{k,c_k})$, if for some i, $\phi(v_i) \neq c_i$, otherwise the clause is *false*. Further, remember that each variable v_i can take a value from $[d]$.

First we need to introduce some notation. We may write $v_i \in R$ for some *variable* v_i, as well as $v_{i,c_i} \in R$ for some *literal* v_{i,c_i}, where the first notation is clear, and the second means that v_{i,c_i} is a literal created from some variable $v_i \in R$.

Suppose $\Phi = (V,C)$ is a formula on n variables. Given a pair (S,ϕ), of a subset $S \subseteq V$ and an assignment $\phi : S \to [d]$ which satisfies $\Phi[S]$, for every $v_i \in V \setminus S$ let $L(v_i) = L_{(S,\phi)}(v_i)$ denote the possible assignments to v_i that will not create a *false* clause with $k-1$ literals from S, that were not satisfied by ϕ (a literal v_{i,c_i} is satisfied if and only if $\phi(v_i) \neq c_i$). Define, in a similar manner, for every $2 \leq j \leq k-1$, and every set of j variables $\{v_1,\ldots,v_j\} \subseteq V \setminus S$, the set $L(v_1,\ldots,v_j) = L_{(S,\phi)}(v_1,\ldots,v_j)$ to be the set of assignmets that can be assigned to $v_1,\ldots,v_j$ that will not create a *false* clause with $k-j$ literals from S, that were not satisfied by ϕ. An important observation is that $\{c_1,\ldots,c_j\} \notin L(v_1,\ldots,v_j)$, if and only if for some clause $(v_{1,c_1} \vee \ldots \vee v_{j,c_j} \vee u_{1,t_1} \vee \ldots \vee u_{k-j,t_{k-j}})$ we have $\{u_1,\ldots,u_{k-j}\} \subseteq S$, and also $\phi(u_1) = t_1, \ldots, \phi(u_{k-j}) = t_{k-j}$. For $1 \leq j \leq k-1$, let L_j denote the collection of all sets $L(v_1,\ldots,v_j)$, for every $\{v_1,\ldots,v_j\} \subseteq V$. We call a clause containing $k-j$ literals from S, that were not satisfied by ϕ, and a set of j literals $\{v_{1,c_1},\ldots,v_{j,c_j}\} \subseteq V \setminus S$ a *witness* to the fact that $\{c_1,\ldots,c_j\} \notin L(v_1,\ldots,v_j)$.

If $S = \emptyset$, we set $L(v_1,\ldots,v_j) = [d]^j$ for every $\{v_1,\ldots,v_j\} \subseteq V$. If a satisfying assignment $c : V \to [d]$ of Φ, coincides with ϕ on S, then for every $\{v_1,\ldots,v_j\} \subseteq V \setminus S$ the assignment of $\{v_1,\ldots,v_j\}$ in c belongs to $L(v_1,\ldots,v_j)$. For this reason, the set $L(v_1,\ldots,v_j)$ is called the set of *feasible assignments* for $\{v_1,\ldots,v_j\}$. Further, denote by *Conflict* the set of

variables, v, for which $L(v) = \emptyset$.

Given a variable v, denote by $\delta(v,t,j)$, the number of feasible assignments that are deleted from L_j, if we add v to S and assign it the value t. We also define, $\delta(v) = \min_{t\in L(v)} \sum_{j=1}^{k-1} \delta(v,t,j)n^{k-j-1}$. We call a variable *heavy* if $\delta(v) > \epsilon n^{k-1}/5$, and denote by *Heavy* the set of *heavy* variables.

CLAIM 2.1. *If Φ is ϵ-far from satisfiable, then for any (S,ϕ), we have $|Conflict \cup Heavy| \geq \epsilon n/5$.*

Proof. Assuming the contrary, we give an algorithm that deletes less than ϵn^k clauses from Φ, and gets a satisfiable subfurmula of Φ, contradicting the fact that Φ is ϵ-far from satisfiable. The algorithm consists of two phases. In phase one the algorithm goes sequentially over all variables in $V \setminus (Conflict \cup Heavy)$, and for each variable v, assigns v the assignment t from $L(v)$, that minimizes the sum $\sum_{j=1}^{k-1} \delta(v,t,j)n^{k-j-1}$. The algorithm also removes all the clauses that are witnesses to the fact that some assignments should be deleted from $L_1,\ldots,L_{k-1}$. After we assign a value to some variable we update all the sets $L(v_1,\ldots,v_j)$, if needed. If for some variable $v \in V \setminus (Conflict \cup Heavy)$, we have $L(v) = \emptyset$, the algorithm halts. In phase two we assign all the variables in $Conflict \cup Heavy$ the value 1, and remove any false clause that was created.

Notice that by definition of the algorithm, as it never creates a false clause, *if* it terminates, it gets a satisfiable subformula of Φ, Φ', and an assignment that satisfies Φ'. The only step in which the algorithm can halt, is if in phase one we had for some variable v, $L(v) = \emptyset$. We argue that this is impossible. Indeed, for every variable v, in $V \setminus (Conflict \cup Heavy)$, we *initially* had $L(v) \neq \emptyset$. As the algorithm deletes any clause that is a witness to the fact that we should delete some feasible assignment from some set $L(v_1,\ldots,v_j)$, and in particular from $L(v)$, we conclude that this event can not occur.

We claim that for every v,t and j, the value of $\delta(v,t,j)$ never increases. Indeed, assume some $\delta(v,t,j)$ increases, thus assigning v the value t deletes from L_j an assignment from some $L(v_1,\ldots,v_j)$, which it did not delete before. This can only happen if v and $v_1,\ldots,v_j$, co-occur in some clause c, with a literal $u \neq p$, and u was assigned the value p. But in this case c is a witness to the fact that we must delete one assignment from $L(v,v_1,\ldots,v_j)$, thus it should have been removed when u was assigned a value. Note that as the values of $\delta(v,t,j)$ never increase, so do the values of $\delta(v)$.

Let us estimate the number of clauses that were removed. Note that for every j,t and $v \notin S$, there are at most $\binom{n}{k-j-1}$ clauses containing v, that are witnesses to the fact that some assignment $\{c_1,\ldots,c_j\}$ should be removed from $L(v_1,\ldots,v_j)$, when we assign v the value t, and add it to S. (This is because any assignment creates at most n false literals). Thus, for every variable v, assigning it the value t, results in deleting at most $\sum_{j=1}^{k-1} \delta(v,t,j)\binom{n}{k-j-1} \leq \sum_{j=1}^{k-1} \delta(v,t,j)n^{k-j-1}$ clauses. In phase one we deal with variables that do not belong to $Heavy$, thus from the definition of the function $\delta(v)$, and from the fact that it does not increase while the algorithm proceeds, we conclude that we can assign each variable in its turn a value, for which we will delete at most $\epsilon n^{k-1}/5$ clauses. As there are at most n variables, in phase one we do not remove more than $\epsilon n^k/5$ clauses. In phase two as we assume that $|Conflict \cup Heavy| \leq \epsilon n/5$, we remove in this phase at most $\epsilon n/5 \cdot \binom{n}{k-1} \leq \epsilon n^k/5$ clauses. All together we remove less than ϵn^k clauses, as claimed.

Let now Φ be a formula on n variables, which is ϵ-*far* from satisfiable. While exposing random variables $r_1,\ldots,r_s$ of R we construct an auxiliary tree T. Each vertex t of T is labeled by a quadruple (S,ϕ,x,σ), where S is a subset of V, ϕ is an assignment that satisfies $\Phi[S]$, x is a variable in $V \setminus S$, and $\sigma \in \{$"open", "closed"$\}$ is the state of the vertex. We refer to the labels of t as $S(t)$, $\phi(t)$, $x(t)$ and $\sigma(t)$, respectively. The value of $x(t)$ may be not set yet, in which case we say that $x(t)$ is *void*. Initially T has only the root t_0 with $S(t_0) = \emptyset$, $\phi(t_0) = \emptyset$, $x(t_0)$ being void, and $\sigma(t_0) =$ "open".

Suppose now that $j-1$ variables of R have already been exposed, and let T be the current tree. Let t be an open leaf of T, labeled by $(S,\phi,$'void', "open"$)$. The pair (S,ϕ) defines the sets $Conflict = Conflict(t)$ and $Heavy = Heavy(t)$, and the lists of feasible assignments $L_1,\ldots,L_{k-1}$ as described above. We say that round j is *successful* for t if $r_j \in Conflict(t) \cup Heavy(t)$. If $r_j \in Conflict(t)$, we set $x(t) = r_j$ and $\sigma(t) =$ "closed". If $r_j \in Heavy(t)$, we set $x(t) = r_j$ and for each assignment $a \in L(r_j)$ create in T a son of t, labeled by $(S \cup \{r_j\},\phi',$'void', "open"$)$, where ϕ' is obtained by extending $\phi : S \to [d]$ by $\phi'(r_j) = a$. If r_j misses the set $Conflict(t) \cup Heavy(t)$, we do nothing related to t. If t is a closed leaf, then round j is successful for t

for any choice of r_j. It is important to note that for a fixed leaf t of T, round j is successful with probability at least $\epsilon/5$, given *any* history, by claim 2.1.

CLAIM 2.2. *The depth of T is at most $5d^{k-1}/\epsilon$.*

Proof. First notice that the initial size of each L_j, is at most $\sum_{\{v_1,\ldots,v_j\}\subseteq V} d^j = \binom{n}{j}d^j \leq \frac{d^j}{j!}n^j$. Therefore, the initial value of $W = \sum_{j=1}^{k-1} |L_j| \cdot n^{k-j-1}$ is at most $\sum_{j=1}^{k-1} \frac{d^j}{j!}n^j \cdot n^{k-j-1} \leq d^{k-1}n^{k-1}$. If we create a new son for some $t \in T$ at step ℓ, then the random variable r_ℓ belongs to $Heavy(t)$. This means by the definition of the set $Heavy(t)$ that for any assignment p in $L(r_\ell)$, we have $\sum_{j=1}^{k-1} \delta(r_\ell, p, j)n^{k-j-1} > \epsilon n^{k-1}/5$, and hence, the value of W decreases by at least $\epsilon n^{k-1}/5$. Thus we can not make more than $5d^{k-1}/\epsilon$ steps down from the root of T.

CLAIM 2.3. *If after round j all leaves of the tree T are labeled "closed", then the induced formula $\Phi[\{r_1,\ldots,r_j\}]$ is not satisfiable.*

Proof. Note first that by the construction of T at any round j for any vertex $t \in T$ with a label (S, ϕ, x, σ) one has $S(t) \subseteq \{r_1 \ldots, r_j\}$, $x(t) \in \{r_1, \ldots, r_j\}$.

Let now $c : V \to [d]$ be an assignment to V. In order to show that c creates some false clause in the induced formula $\Phi[\{r_1, r_2, \ldots, r_j\}]$, we start with the root t_0 of T and traverse T guided by c.

Suppose we are at a vertex t of T, labelled by (S, ϕ, x, σ). Based on the pair $(S(t), \phi(t))$ define the lists of feasible assignments $L(x)$ as described above. Now, if $c(x(t)) \in L(x(t))$, we choose the son of t in which $x(t)$ is assigned $c(x(t))$ and move to it. Suppose now that at a vertex t we have for the first time $c(x(t)) \notin L(x(t))$. This means that some set of variables $\{u_1, \ldots, u_{k-1}\} \subseteq S(t)$ co-occurs with $x(t)$ in Φ and creates a false clause $(\ell_1 \vee \ldots \vee \ell_k)$, where ℓ_i are *literals* created from $u_1, \ldots, u_{k-1}, x(t)$ respectively. But it is easy to see that $\phi(t)$ and c coincide on $S(t)$. Therefore, under c all k literals of the clause $(\ell_1 \vee \ldots \vee \ell_k) \in C(\Phi)$ are evaluated false, thus creating a false clause. As $S(t) \subseteq \{r_1, \ldots, r_j\}$ and $x(t) \in \{r_1, \ldots, r_j\}$ we get that c is not a proper assignment of the induced formula $\Phi[\{r_1, \ldots, r_j\}]$.

Recall that by the construction of T we have $L(x(t)) = \emptyset$ for every closed leaf $t \in T$. As we assume that after round j all leaves of T are closed, the above described traversal procedure eventually ends up in a vertex t with $c(x(t)) \notin L(x(t))$. Hence, c is not a satisfying assignment for $\Phi[\{r_1, \ldots, r_j\}]$.

CLAIM 2.4. *After $(100d^{k-1}\log d)/\epsilon^2$ rounds, all leaves of T are closed with probability at least $3/4$.*

Proof. As every vertex of T has at most d sons and by Claim 2.2 T has depth at most $5d^{k-1}/\epsilon$, it can be embedded naturally in the d-ary tree $T_{d,5d^{k-1}/\epsilon}$ of depth $5d^{k-1}/\epsilon$. Moreover, this embedding can be prefixed even before exposing R and T. Note that the number of leaves of $T_{d,5d^{k-1}/\epsilon}$ is at most $d^{5d^{k-1}/\epsilon}$.

Fix a leaf t of $T_{d,5d^{k-1}/\epsilon}$. The probability that t is an open leaf of T after $(100d^{k-1}\log d)/\epsilon^2$ rounds is at most the probability that the total number of successful rounds on the path from the root of T to t is less than $5d^{k-1}/\epsilon$. For the path from the root to t, each round has probability of success at least $\epsilon/5$, given any history, by claim 2.1. Therefore, the probability that t is an open leaf after $(100d^{k-1}\log d)/\epsilon^2$ steps can be bounded from above by the probability that the Binomial random variable $B(100d^{k-1}\log d/\epsilon^2, \epsilon/5)$ is less than $5d^{k-1}/\epsilon$. Using the Chernoff bound (see e.g. [4]), the latter probability is at most

$$\exp\left\{-\frac{\left(\frac{20d^{k-1}\log d}{\epsilon} - \frac{5d^{k-1}}{\epsilon}\right)^2}{\frac{40d^{k-1}\log d}{\epsilon}}\right\} \leq$$

$$\exp\left\{-\frac{\left(\frac{15d^{k-1}\log d}{\epsilon}\right)^2}{\frac{40d^{k-1}\log d}{\epsilon}}\right\} <$$

$$e^{-\frac{5.5d^{k-1}\log d}{\epsilon}} <$$

$$d^{-\frac{5.5d^{k-1}}{\epsilon}} .$$

Thus, by the union bound we conclude that the probability that some leaf of T is open after $(100d^{k-1}\log d)/\epsilon^2$ steps, is at most $d^{5d^{k-1}/\epsilon}d^{-5.5d^{k-1}/\epsilon} < 1/4$.

Theorem 1.1 now follows from claims 2.3 and 2.4.

3 Applications.

In this section we show how to obtain the corollaries mentioned in the introduction, by reducing these problems to testing (k,d)-CNF. We remind the reader that we denote by $f(k, d, \epsilon)$ the number of variables that suffice for testing (k,d)-CNF, as described in section 2.

Proof. (of Corollary 1.1) We divide the proof into two parts. First we show how to reduce general functions into an equivalent set of (k,d)-CNF clauses, and then show how to use the (k,d)-CNF tester. Let $f : d^k \to \{0,1\}$ be a general boolean function on k variables, where each variable can take a value from $[d]$. Clearly we may represent f as an instance of (k,d)-CNF, where we have a clause $(v_1 \neq a_1 \vee \ldots \vee v_k \neq a_k)$, for each assignment $(a_1, \ldots, a_k)$ that does *not* satisfy f. Now, given an instance, Φ, of (k,d)-Function-SAT, we can *implicitly* create an instance, Φ', of (k,d)-CNF by representing each function as a set of (k,d)-CNF as described above. We also remove any duplication from Φ'. Now we turn to show how to use the (k,d)-CNF tester, in order to test (k,d)-Function-SAT. We claim that choosing $f(k,d,\epsilon/2^{d^k})$ variables suffices. Clearly if Φ is satisfiable so is Φ', as well as any induced subformula of Φ'. So assume Φ is ϵ-*far* from satisfiable, we show that Φ' is $\epsilon/2^{d^k}$-*far* from satisfiable. Assume there is some assignment that does not satisfy less than $\epsilon/2^{d^k} n^k$ clauses in Φ', and notice that as there are no more than 2^{d^k} boolean functions on k variables taking values from $[d]$, we conclude that the same assignment does not satisfy less than ϵn^k functions in Φ, which is a contradiction, as we assumed that Φ is ϵ-*far* from satisfiable. Now, choosing $f(k,d,\epsilon/2^{d^k})$ variables and taking all the functions that include them, is equivalent to picking $f(k,d,\epsilon/2^{d^k})$ random variables from Φ' and looking at the induced formula on Φ', and thus ensures, by Theorem 1.1, that the induced formula on Φ is not satisfiable with probability at least $\frac{3}{4}$, as needed.

Proof. (of Corollary 1.2) Follows from Theorem 1.1 with $d = 2$.

Proof. (of Corollary 1.3) Given an instance Φ of k-NAEQ-SAT, *implicitly* create an instance of k-CNF, Φ', where for each clause, c, in Φ, put c in Φ', plus another clause that has the same set of variables as c, but all the signs of the literals are flipped (that is for, e.g., the clause $c = (v_1 \vee \overline{v_2} \vee v_3)$ we put c and the clause $(\overline{v_1} \vee v_2 \vee \overline{v_3})$). The proof of correctness follows trivially from Theorem 1.1.

Proof. (of Corollary 1.4) Given a k-uniform hypergraph, H, we *implicitly* create a (k,d)-CNF instance, Φ, such that Φ contains a variable for each vertex in H. For each edge $(v_1, \ldots, v_k)$ we put d clauses in Φ, $(v_1 \neq 1 \vee \ldots \vee v_k \neq 1), \ldots, (v_1 \neq d \vee \ldots \vee v_k \neq d)$. Clearly if H is d-colorable, Φ is satisfiable. Assume there is some assignment that does not satisfy less than ϵn^k clauses. For each variable v that was assigned the value i, we color the vertex v with the color i. Clearly under this coloring no more than ϵn^k edges are not properly colored. We conclude that if H is ϵ-*far* from d-colorable, then ϕ is ϵ-*far* from satisfiable. Now in order to test if H is d-colorable, we pick $f(k,d,\epsilon)$ vertices, and for every induced edge, we create the d corresponding clauses. This is equal to picking $f(k,d,\epsilon)$ variables from Φ, thus from Theorem 1.1, with probability at least $\frac{3}{4}$, the resulting formula is not satisfiable.

Note that for the case of testing graph d-colorability, that is the case of $k = 2$, our estimation of $COL_{2,d}(n,\epsilon)$ matches the result of Alon and Krivelevich [3]. For general k we get the same result as that of Czumaj and Sohler [8], but with a larger dependence on k and d. In fact, if one changes the definition of a *heavy* vertex in the proof of [8], to the way we define a *heavy* variable, one can slightly improve the result of [8], and show that $COL_{k,d}(n,\epsilon) = O(kd \log d/\epsilon^2)$. We omit the details.

Let S be a *fixed* set of (not necessarily distinct) graphs on k vertices, where each graph is also associated with some forbidden coloring of its vertices. Consider the problem of testing whether there is a d coloring of G, ϕ, such that for any copy C, of any graph from S, ϕ does not color C in the forbidden coloring. Suppose we want to distinguish between graphs that can be so colored, and those from which at least ϵn^2 edges should be removed in order to be so colored.

Alon [1] has recently shown, that in general there is no one-sided error testing algorithm for this problem, that uses a random set of vertices of size polynomial in $1/\epsilon$. In fact, even for the very special case of the problem (with $|S| = 1$, $d = 1$ and $k = 3$) of testing if a graph is triangle free, one needs to pick a random set of vertices of size $(1/\epsilon)^{\Omega(\log(1/\epsilon))}$.

Assume we relax this problem to that of deciding whether G can be properly colored in the above sense, or for any d coloring of G, there are at least ϵn^k copies of graphs from S that are improperly colored. We call this problem the *extended d-coloring* problem (with respect to S.)

Note that for $d = 1$ the problem is trivial, as we only want to distinguish between graphs that contain no copy of graphs from S, and those that contain at least ϵn^k copies, thus choosing $\Theta(1/\epsilon)$ vertices suffices. The problem is harder for $d > 1$. We get the following claim,

CLAIM 3.1. *Extended d-colorability can be tested using a randomly chosen subset of $f(k,d,\epsilon/|S|)$ vertices.*

Proof. Given a graph, G, we *implicitly* create a (k,d)-CNF instance, Φ, such that Φ contains a variable for each vertex in G. For each set of vertices $\{v_1,\ldots,v_k\}$, that include a copy of a graph from S, and for every assignment $v_1 = c_1,\ldots,v_k = c_k$, that creates an improperly colored copy of some graph from S, we put the clause $(v_1 \neq c_1 \vee \ldots \vee v_k \neq c_k)$ in Φ. We then remove any duplications from Φ. Clearly if G is extended d-colorable, Φ is satisfiable. Assume there is some assignment that does not satisfy less than $\frac{\epsilon}{|S|}n^k$ clauses. For each variable v that was assigned the value i, we color the vertex v with the color i. Clearly under this coloring no more than ϵn^k copies of graphs from S are improperly colored. We conclude that if G is *ϵ-far* from being extended d-colorable, then ϕ is *$\epsilon/|S|$-far* from satisfiable.

It follows that for every fixed S and d, the extended d-coloring problem with respect to S, can be tested by inspecting an induced subgraph on $O((1/\epsilon)^2)$ randomly chosen vertices.

4 Concluding remarks and open problems.

We have shown how to obtain one-sided error property-testers for a number of satisfiability problems. As a byproduct, we have obtained as a consequence, previously known results about testing colorability of graphs and k-uniform hypergraphs. We have also discussed the problem of testing extended d-colorability. All the results use a random set of variables/vertices of size $O(1/\epsilon^2)$. There is still a gap between the trivial lower bound of $\Omega(1/\epsilon)$ and our upper bounds.

A natural question that can be asked, is whether one can devise a more efficient test, if the functions are restricted to be of a specific type. Two natural types of functions are linear equations and multivariate polynomials over $GF(d)$ (for a prime power d). As the following claims show, designing much better tests for these special cases would not be easy.

CLAIM 4.1. *Testing graph 2-colorability is not harder than testing linear equations on two variables over $GF(2)$.*

Proof. Given a graph $G = (V,E)$, where $|V| = n$, consider the set of equations, Φ, on the variables $x_1,\ldots,x_n$, over $GF(2)$, where for every edge $(v_i,v_j) \in E$, Φ contains the equation $x_i + x_j = 1$. Clearly if G is 2-colorable, Φ is satisfiable, and also if G is ϵ-far from satisfiable, so is Φ.

Alon and Krivelevich [3] found a complicated proof for the existence of a test for graph 2-colorability, which chooses a random subset of vertices of size $\tilde{\Theta}(1/\epsilon)$. Thus getting a test for general linear equations, that uses a set of variables of size $O(1/\epsilon)$, would improve the result of [3] by a poly-logarithmic factor, and does not seem to be easy.

CLAIM 4.2. *For any prime power p, testing (k,p)-CNF is not harder than testing k-variate polynomials over $GF(p)$.*

Proof. Given an instance of (k,p)-CNF, $\Phi = (V,C)$, with $|V| = n$, consider the following set of k-variate polynomials over $GF(p)$, on a set of variables $x_1,\ldots,x_n$, which we denote by Φ'. For every clause $(v_1 \neq c_1 \vee \ldots \vee v_k \neq c_k)$ in Φ, Φ' contains the k-variate polynomial $\Pi_{i=1}^{k}\Pi_{t\in\{1,\ldots,p\}\setminus c_i}(x_i - t) = 0$. Clearly if Φ is satisfiable so is Φ', and also if Φ is ϵ-far from satisfiable, so is Φ'.

By Corollary 1.1, testing *general* boolean functions (which we call (k,d)-Function-SAT), is as easy as testing (k,d)-CNF, thus getting a test for k-variate polynomials over $GF(d)$, which would use $o(1/\epsilon^2)$ variables, would imply an equivalent result for testing (k,d)-Function-SAT (at least for a prime power d). As even for the very special case of testing 3-colorability of graphs, we do not know how to get a test which uses $o(1/\epsilon^2)$ vertices, this also seems a hard task.

Another open problem that should be addressed is a slight variation of the problem we have addressed in this paper. Notice that for the case of k-hypergraph d-colorability we show that if an instance H is *ϵ-far* from being d-colorable, then *almost all* the induced k-hypergraphs on sets of size roughly $1/\epsilon^2$ are not d-colorable. The other question that can be asked is what is the size of the smallest induced k-hypergraph that is not d-colorable, that *must* be included in any H, that is *ϵ-far* from d-colorable. See [3] and [6] for a discussion on these questions and some results for the case of graphs. The analogous questions for satisfiability and its variants are also interesting.

References

[1] N. Alon, Testing subgraphs in large graphs, Proc. 42^{nd} IEEE FOCS, IEEE (2001), to appear.

[2] N. Alon, R. A. Duke, H. Lefmann, V. Rödl and R. Yuster, The algorithmic aspects of the Regularity Lemma, Proc. 33^{rd} IEEE FOCS, Pittsburgh, IEEE (1992), 473-481. Also: J. of Algorithms 16 (1994), 80-109.

[3] N. Alon and M. Krivelevich, *Testing k-colorability*, SIAM J. Discrete Math., to appear.

[4] N. Alon and J. H. Spencer, **The probabilistic method**, Second Edition, Wiley, New York, 2000.

[5] G. Andersson and L. Engebretsen, *Property Testers For Dense Non-Boolean Constraint Satisfaction Programs*, in preparation.

[6] B. Bollobás, P. Erdös, M. Simonovits and E. Szemerédi, Extremal graphs without large forbidden subgraphs, *Annals of Discrete Mathematics* 3 (1978), 29–41.

[7] S. Cook, The complexity of theorem-proving procedures, *In Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151-158, 1971.

[8] A. Czumaj and C. Sohler, Testing Hypergraph Coloring, Proc. 28^{th} ICALP, Lecture Notes in Computer Science Vol. 2076 (F. Orejas et. al., eds.), Springer Verlag (2001), 493-505.

[9] A. Frieze and R. Kannan, Quick approximation to matrices and applications, *Combinatorica*, 19, (1999), 175-220.

[10] M.R. Garey and D.S. Johnson, *Computers and Intractability*, Freeman and Company, San Francisco, 1979.

[11] O. Goldreich, S. Goldwasser and D. Ron, Property testing and its connection to learning and approximation, *Proceedings of the* 37^{th} *Annual IEEE FOCS* (1996), 339–348. Also: Journal of the ACM, 45(1998), 653-750.

[12] J. Håstad, Some optimal inapproximability results, In Proc. 29th Ann. ACM Symp. on Theory of Comp., pages 1–10. ACM, 1997.

[13] Howard Karloff, Uri Zwick, A 7/8-approximation algorithm for MAX 3SAT? In Proc. of 38th FOCS (1997), 406–415.

[14] V. Rödl and R. Duke, On graphs with small subgraphs of large chromatic number, *Graphs and Combinatorics* 1 (1985), 91–96.

[15] D. Ron, Property Testing. To appear in P.M. Pardalos, S. Rajasekaran, J. Reif, and J. D. P. Rolim, editors, Handbook of Randomized Algorithms. Kluwer Academic Publishers, 2001.

[16] E. Szemerédi, Regular partitions of graphs, In: *Proc. Colloque Inter. CNRS* (J. C. Bermond, J. C. Fournier, M. Las Vergnas and D. Sotteau, eds.), 1978, 399–401.

5 Appendix.

This short section contains the precise definitions of the problems considered in this paper.

DEFINITION 5.1. *k-CNF (or k-SAT)*

INPUT: A set of boolean functions on n variables, where each function is of the form $(l_1 \vee \ldots \vee l_k)$, and each *literal* l_i is either v_i, or $\overline{v_i}$, for some *variable* v_i.

An instance, Φ, of k-CNF on a set V of n variables, is said to be satisfiable, if there is a truth assignment, $\phi : V \to \{0, 1\}$, to the n variables, that *simultaneously* satisfies all the functions in Φ. An instance, Φ, of k-CNF on n variables, is said to be ϵ-*far* from satisfiable, if any assignment does not satisfy at least ϵn^k functions from Φ.

DEFINITION 5.2. *(k,d)-CNF*

INPUT: A set of boolean functions on n variables, where each function is of the form $(v_{1,i_1} \vee \ldots \vee v_{k,i_k})$, where the literal v_{t,i_t} is short for $v_t \neq i_t$, for $1 \leq i_t \leq d$.

An instance, Φ, of (k,d)-CNF on a set V of n variables, is said to be satisfiable, if there is an assignment, assigning to each variable v_i a value $\phi(v_i) \in [1, d]$, that *simultaneously* satisfies all the functions in Φ. An instance, Φ, of (k,d)-CNF on n variables is said to be ϵ-*far* from satisfiable, if any assignment does not satisfy at least ϵn^k functions from Φ.

DEFINITION 5.3. *(k,d)-Function-SAT.*

INPUT: A set of general boolean functions on n variables, where each function depends on exactly k variables, and each variable can take a value from $[1, d]$.

An instance, Φ, of (k,d)-Function-SAT on a set V of n variables, is said to be satisfiable, if there is an assignment $\phi : V \to [1, d]$, to the n variables, that *simultaneously* satisfies all the functions in Φ. An instance, Φ, of k-CNF on n variables, is said to be ϵ-*far* from satisfiable, if any assignment does not satisfy at least ϵn^k functions from Φ.

DEFINITION 5.4. *k-NOT-ALL-EQUAL-CNF (k-NAEQ-CNF)*

Exactly like k-CNF, only now a satisfying assignment is one in which for each clause, at least one literal evaluates $false$, and at least one literal evaluates $true$.

DEFINITION 5.5. *k-Hypergraph d-Colorability*

INPUT: A k-uniform Hypergraph (k-Hypergraph) on n vertices.

A k-Hypergraph, H, is said to be d-colorable, if one can color the vertices of H using d colors, such that no edge of H is monochromatic. A k-Hypergraph H is ϵ-*far* from d-colorable, if any coloring of its vertices results in at least ϵn^k monochromatic edges.