2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Almost Optimal Canonical Property Testers for Satisfiability

Christian Sohler *
Department of Computer Science
TU Dortmund University
Dortmund, Germany
Email: christian.sohler@tu-dortmund.de

***Abstract*—In the (k,d)-Function-SAT problem we are given a set of n variables $\{X_1,\ldots,X_n\}$ that can take values from the set $\{1,\ldots,d\}$ and a set of Boolean constraints on these variables, where each constraint is of the form $f:\{1,\ldots,d\}^k \to \{0,1\}$, i.e. the constraint depends on exactly k of these variables. We will treat k and d as constants. The goal is to determine whether the set of constraints has a satisfying assignment, i.e. an assignment to the variables such that all constraints simultanuously map to 1. In this paper, we study (k,d)-Function-SAT in the property testing model for dense instances. We call an instance ε-far from satisfiable, if every assignment violates more than εn^k constraints. A property testing algorithm is a randomized algorithm that, given oracle access to the set of constraints, must accept with probability at least $3/4$ all satisfiable inputs and rejects with probability at least $3/4$ all inputs, which are ε-far from satisfiable.**

We analyze the canonical non-adaptive property testing algorithm with one-sided error: Sample r variables and accept, if and only if the induced set of constraints has a satisfying assignment. The value of r will be called the *sample complexity* of the algorithm. We show that there is an $r_0 = \tilde{O}(1/\varepsilon)$ such that for any instance that is ε-far from satisfiable, the probability, that a random sample on $r \geq r_0$ variables is satisfiable, is at most $1/4$. This implies that the above algorithm is a property tester. The obtained sample complexity is nearly optimal for canonical testers as a lower bound of $\Omega(1/\varepsilon)$ on the sample complexity is known.

Previously, a tester with sample complexity $o(1/\varepsilon^2)$ was only known for the very special case of testing bipartiteness in the dense graph model [3]. Our new general result improves the best previous result for testing satisfiability (and even for the special case of 3-colorability in graphs) from sample complexity $\tilde{O}(1/\varepsilon^2)$ to $\tilde{O}(1/\varepsilon)$. It also slightly improves the sample complexity for the special case of bipartiteness. Improving the sample complexity for (k,d)-Function-SAT (or special cases of it) had been posed in several papers as an open problem [3], [4], [17]. This paper solves this problem nearly optimally for canonical testers and, in the case of $k=2$, also for non-adaptive testers as there is a lower bound of $\Omega(1/\varepsilon^2)$ on the query complexity of non-adaptive testers for bipartiteness in the dense graph model [6], where the query complexity denotes the number of queries asked about the graph (for a canonical tester in graphs, the query complexity is the square of its sample complexity).

As a byproduct, we obtain an algorithm, which, given a satisfiable set of constraints, computes in time $O(n/\varepsilon^{O(1)} + 2^{\tilde{O}(1/\varepsilon)})$ a solution, which violates at most εn^k constraints.

***Keywords*-Property Testing, Satisfiability Problems**

* Supported by DFG grant So 514/3-2.

I. Introduction

The problem (k,d)-Function-SAT is a fairly general satisfiability problem that contains problems like 3-SAT, k-SAT, and k-colorability of hypergraphs as special cases. In this problem we deal with Boolean functions of the form $f:\{1,\ldots,d\}^k \to \{0,1\}$. In this paper, we will assume k and d to be constants. We are given a set $V=\{X_1,\ldots,X_n\}$ of variables that can take values from $\{1,\ldots,d\}$ and a set E of Boolean functions each defined on k of these variables. We will also refer to these functions as constraints. A constraint f is satisfied by an assignment to the variables in V, if f evaluates to 1. The set E is satisfied by an assignment to the variables in V, if *all* functions in E evaluate simultanously to 1. If there exists a satisfying assignment for E, we call E satisfiable. Satisfiability problems play a central role both in practice and in theory and a decent amount of research has been put into trying to solve them as efficiently and accurately as possible. It is well-known that even solving special cases of (k,d)-Function-SAT like the ones mentioned above is NP-hard. Since we do not know how to solve (k,d)-Function-SAT efficiently, a natural question is whether we can at least approximately decide whether a given instance is satisfiable. A question of this type is typically studied in the framework of property testing that was introduced by Rubinfeld and Sudan [18].

The property testing problem considered in this paper can also be viewed as the following promise problem. We are either given an instance that is satisfiable or an instance, in which we have to delete more than εn^k constraints to make it satisfiable. In the latter case we say that the instance is ε-far from satisfiable. Can we efficiently distinguish between these two cases?

A natural approach to this problem is to sample a small subset S of variables and accept, iff the induced set of constraints, i.e. the set of all constraints that include only variables of S, is satisfiable. Such an algorithm is also called a *canonical property tester* [13]. Obviously, if an input instance is satisfiable, then also the subset of constraints is satisfiable and a canonical tester accepts. This raises the question how large S has to be to guarantee that with reasonable probability we do not accept instances of the second type. Similarly to [4] we define $\mathrm{SAT}_{k,d}(n,\varepsilon)$ to

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.59

be the minimal value, such that for every instance E of (k,d)-Function-SAT on n variables $\{X_1,\ldots,X_n\}$ that is ε-far from satisfiable, a random, uniformly distributed subset of the variables of size $\text{SAT}_{k,d}(n,\varepsilon)$ is not satisfiable with probability at least $3/4$. Furthermore (and also similarly to [4]), we define $\text{COL}_{k,d}$ as the minimum value, such that for every k-uniform hypergraph that is ε-far from having a proper d-coloring in the dense hypergraph model, a hypergraph induced by a random sample of $\text{COL}_{k,d}$ vertices is not d-colorable with probability at least $3/4$. In this context, a hypergraph is d-colorable, if there is an assigment $\Phi : V \to \{1,...,d\}$ such that there is no monochromatic edge, i.e. no edge e such that $|\{\Phi(v) : v \in e\}| = 1$.

In the remainder of the paper we will use the term *sample complexity* to denote the number of sampled variables. We will assume that our algorithm can query in constant time (recall that we treat k and d as constants) for a set Q of k variables the set of constraints that contain only variables from Q (since we assume that each constraint is defined on exactly k variables, this will be the set of constraints whose set of variables is Q). The *query complexity* of an algorithm will denote the number of queries about functions in E. If a canonical algorithm samples a set S of r variables then it has to query for the $\Theta(r^k)$ possible constraints defined on S. Thus, if a canonical tester has sample complexity $\Theta(r)$ then its query complexity is $\Theta(r^k)$.

A. Results

In this paper we prove that, for constant k and d, $\text{SAT}_{k,d}(n,\varepsilon) = \widetilde{O}(1/\varepsilon)$, i.e. the sample complexity of a canonical tester for (k,d)-Function-SAT is $\widetilde{O}(1/\varepsilon)$. This sample complexity is almost optimal as it is known that $\text{SAT}_{k,d}(n,\varepsilon) = \Omega(1/\varepsilon)$. This can be seen by considering an instance that has a set S of εn special variables and an all zero constraint for every set of k variables that contains at least one variable from S. This instance is $c\cdot\varepsilon$-far from satisfiable for a constant $c > 0$ and in order to recognize it, one needs to sample at least one variable from S, which requires to sample $\Omega(1/\varepsilon)$ variables. The previously best upper bound on the sample complexity was $\widetilde{O}(1/\varepsilon^2)$ [4].

Theorem 1. *For constant d and k,*

$$SAT_{k,d}(n,\varepsilon) = O(\frac{\log^2(1/\varepsilon)\log\log(1/\varepsilon)}{\varepsilon}).$$

Furthermore, the problem of d-coloring a hypergraph is a special case of (k,d)-Function-SAT since one can have a variable for each vertex and then replace every hyperedge e by a constraint not allowing to assign the same value $c \in \{1,\ldots,d\}$ to all vertices of e.

Corollary 2. *For constant d and k,*

$$COL_{k,d}(n,\varepsilon) = O(\frac{\log^2(1/\varepsilon)\log\log(1/\varepsilon)}{\varepsilon}).$$

Again, we improve the best previous results from $\widetilde{O}(1/\varepsilon^2)$ to $\widetilde{O}(1/\varepsilon)$. Even for the case of testing 3-colorability in the dense graph model the best previous result on the sample complexity was $\widetilde{O}(1/\varepsilon^2)$ [4][8][3]. The only special case, where a sample complexity of $\widetilde{O}(1/\varepsilon)$ was known, was for testing bipartiteness in the dense graph model, i.e. $k = d = 2$ and required a relatively complex proof [3]. Our new result also slightly improves this bound by a factor of $\Theta(\log^2(1/\varepsilon))$.

Another property, which can be described as an instance of $(2,d)$-Function-SAT, is the graph property of not having a homomorphism into a fixed graph $H = (V', E')$ (a graph homomorphism is a map $f : V \to V'$ with $(u,v) \in E$ implying $(f(u), f(v)) \in E'$). This property generalizes d-colorability, which is equivalent to not having a homomorphism into the complete graph on d vertices K_d. Note that by a lower of $\Omega(1/\varepsilon^2)$ on the query complexity of any non-adaptive tester for bipartiteness in the dense graph model [6], this also implies that our algorithm with its sample complexity of $\widetilde{O}(1/\varepsilon)$ is nearly optimal among non-adaptive property testers.

Corollary 3. *The property of not having a homomorphism into a fixed (constant size) graph H can be tested in the dense graph model with a canonical tester with sample complexity*

$$O(\frac{\log^2(1/\varepsilon)\log\log(1/\varepsilon)}{\varepsilon}).$$

The more specialized problem of being a blow-up of a constant size graph (which can also be phrased as an instance of $(2,d)$-Function-SAT) is already known to be non-adaptively testable in the dense graph model with $\widetilde{O}(1/\varepsilon)$ query complexity [5], where a graph $G = (V,E)$ is a blow-up of a graph $H = (V_H, E_H)$, $V_H = \{1,\ldots,d\}$, if the vertices in G can be clustered in upto d clusters such that a vertex in cluster i is connected to a vertex in cluster j, if and only if $(i,j) \in E_H$.

B. Techniques

The algorithm we consider samples a set of variables and checks whether the set of constraints that include only sampled variables is satisfiable. Such an algorithmic approach is fairly standard for property testing. The difficult part is the analysis of the algorithm. Here, we build upon a previous proof technique introduced in [8] for coloring of hypergraphs and, independently, in [3] for coloring of graphs (both building upon earlier work by Goldreich, Goldwasser and Ron [11]) and then generalized and first applied to satisfiability in [4] (see also [9]). In order to prove that an ε-far instance is rejected, the main idea of this technique is to show that, a random sample of the variables will contain with probability at least $3/4$ a collection of subsets of sample variables such that

- each assignment to the variables is already violated on the constraints induced by at least one of these subsets,
- the size of each subset is small, i.e. $\widetilde{O}(1/\varepsilon)$.

These subsets then constitute a 'proof of rejectance', i.e. whenever we find such a collection of sets, the sample set is not satisfiable and so the canonical algorithm rejects.

In order to construct the subsets, it will be convenient to view the sample as being sampled sequentially. Furthermore, we will fix a sequence σ of values that will be assigned to variables added to the subset in the order they are considered, i.e. the ith variable added will receive the ith value from the sequence. At the end we will use the union bound over all sequences σ to argue that the analysis works for all sequences. For the proof we design a deterministic rule that chooses whether or not to add the next sample variable to the subset. The decision is based only on the current variable, its number in the sequence, and the variables that were previously added to the subset and their assigned values.

The crux of the analysis is to construct a good rule to add or discard variables (and this is only interesting for the case that the input in ε-far from satisfiable as otherwise the algorithm always accepts). Previously [8], [4], this rule was based on the current assignment and the current sample variable. Our new idea is to design a more complex rule that also takes the history of constructing this assignment into account. In more detail, we will introduce the notion of the *potential* of a variable and the rule will be to add only variables, whose potential is above a certain threshold. The potential has the nice property that it decreases monotonically as more variables are included (this stands in contrast to a notion of heaviness that has previously been used as a criterion to add variables). Furthermore, the sample sequence will be subdivided into different stages and the threshold to decide whether or not to add a variable to the subset will be different for different stages. In general, the early stages try to add only variables with very high potential. These variables may occur less frequently but it will also suffice to include few of them to prove that the current assignment is not a satisfying assignment. This means that it suffices to consider short sequences σ, which makes it easier to apply a union bound to prove that the argument works for all of them. In the later stages, we will lower the threshold for inclusion in the sample set. This has the effect that we need to include more variables to obtain a proof for infeasibility but in instances that are ε-far there will be many such variables, so we can find enough of them with high probability and the union bound (now over longer sequences) will work again. The fact that the potential is monotonically decreasing allows us to combine the different stages. A somewhat similar idea has been used in [3] in the context of testing bipartiteness in graphs, where the sample set was split into $O(\log(1/\varepsilon))$ subsets and the ith subset dealt with vertices of degree roughly n/e^i.

C. Related Work

The first result in the direction of testing satisfiability was a one-sided error tester by Frieze and Kannan [10]. They showed that one can approximate Max-$(k,2)$-Function-SAT with additive error εn^k with a sample complexity, which is exponential (for $k > 2$) in $1/\varepsilon$. Then Andersson and Engebretsen developed a tester with two-sided error and sample complexity $\widetilde{O}(1/\varepsilon^5)$ [1]. Alon, Fernandez de la Vega, Kannan and Karpinski gave a tester that can estimate the distance to satisfiability and that has a sample complexity of $\widetilde{O}(1/\varepsilon^{12})$ variables [2]. Very recently, Karpinski and Schudy showed that in time $O(n/\varepsilon^2 + 2^{O(1/\varepsilon^2)})$ one can find an approximate solution for dense Max-(k,d)-Function-SAT with additive error εn^k [15].

The problem of testing (k,d)-Function-SAT with one-sided error was first studied by Alon and Shapira [4]. They obtained a property testing algorithm that samples a set of $O(1/\varepsilon^2)$ variables. The special case of testing k-colorability in graphs and hypergraphs has received a lot of attention. It has been first studied in the context of property testing by Goldreich, Goldwasser and Ron [11], who showed that $\mathrm{COL}_{2,2}(n,\varepsilon) = \widetilde{O}(1/\varepsilon^2)$ and that $\mathrm{COL}_{2,d}(n,\varepsilon) = \widetilde{O}(d^2/\varepsilon^3)$. Alon and Krivelevich improved the result to $\mathrm{COL}_{2,2}(n,\varepsilon) = \widetilde{O}(1/\varepsilon)$ and $\mathrm{COL}_{2,d}(n,\varepsilon) = \widetilde{O}(d/\varepsilon^2)$ [3]. Independently, Czumaj and Sohler[8] showed that $\mathrm{COL}_{k,d}(n,\varepsilon) = \widetilde{O}(k^2d^2/\varepsilon^2)$. Finally, Alon and Shapira proved the most general result of this form known upto now by showing that $\mathrm{SAT}(n,\varepsilon) = O(d^{k-1}2^{d^{2k}}\log d/\varepsilon^2)$ [4] and $\mathrm{COL}_{k,d} = O(d^{k-1}\log d/\varepsilon^2)$.

As has been noted in [4], there are earlier papers [7], [16] that imply that $\mathrm{COL}_{2,d}(n,\varepsilon)$ is bounded by some function independent of n. These papers are based on the regularity lemma and the bounds are significantly weaker than the ones in the papers cited above.

Adaptivity vs. Non-Adaptivity.: An interesting direction of research is to study the power of adaptive testers. It is known that any tester for graph properties with query complexity $q(\varepsilon)$ implies a non-adaptive canonical tester with query complexity $q(\varepsilon)^2$ [13]. Gonen and Ron [14] showed that for graph with maximum degree $O(\varepsilon n)$ one can test bipartiteness in $\widetilde{O}(1/\varepsilon^{3/2})$ time matching a lower bound by Bogdanov and Trevisan [6]. The question without the degree bound of $O(\varepsilon n)$ remained open. Goldreich and Ron [12] proved that there are properties which can be tested adaptively with $\widetilde{O}(1/\varepsilon)$ query complexity but which require $\Omega(1/\varepsilon^{3/2})$ query complexity for any non-adaptive tester. Furthermore, they showed in [12] that there are also properties where adaptivity does not help (much).

In this paper, we only consider one-sided canonical testers, i.e. testers that sample variables uniformly at random and decide based on the sample. These testers are non-adaptive and we do not know whether adaptivity can be used to improve upon our results.

II. Preliminaries

A Boolean function $f : \{1, \ldots, d\}^k \to \{0, 1\}$ is called a (k, d)-*function*. Let $V = \{X_1, \ldots, X_n\}$ be a set of n variables and let E be a set of (k, d)-functions on the variables from V. For a function $f \in E$ we write $V(f)$ to denote the set of k variables f is defined on. An assignment to the variables in V is a function $\Phi_V : V \to \{1, \ldots, d\}$. We use the index of an assignment to denote its domain. Given an assignment $\Phi_V : V \to \{1, \ldots, d\}$ we use $\Phi_{V|U}$ to denote the restriction of Φ_V to the set U. A function $f \in E$ is satisfied under an assignment Φ_V, if f maps to 1 for the values assigned by Φ_V to $V(f)$. In this case Φ is called a satisfying assignment. Furthermore, if for a given assignment Φ at least one constraint evaluates to 0, we say that the assignment violates the set of constraints, or for short, is a violating assignment. A subset $E' \subseteq E$ is satisfied by an assignment Φ_V, if every function $f \in E'$ is satisfied by Φ_V. The problem (k, d)-Function-SAT is to decide on input $H = (V, E)$ whether there exists an assignment to the variables in V such that all functions in E are satisfied.

Definition 4. *An instance (V, E) of (k, d)-Function-SAT is ε-far from satisfiable, if every assignment to the variables in V violates more than εn^k constraints from E.*

For a subset of variables $U \subseteq V$ we use $E[U] = \{f \in E : V(f) \subseteq U\}$ to denote the subset of functions from E induced by U, i.e. that depend only on variables from U and we write $H[U] = (U, E[U])$.

An algorithm with access to instances of (k, d)-Function-SAT is a *property tester*, if it accepts every satisfiable instance with probability at least $3/4$ and rejects every instance that is ε-far from satisfiable with probability at least $3/4$.

Similarly to [4], our approach to the general problem of testing (k, d)-Function-SAT is via the special case of testing (k, d)-CNF. A (k, d)-CNF clause has the form $(X_{i_1} \neq c_1 \vee X_{i_2} \neq c_2 \vee \cdots \vee X_{i_k} \neq c_k)$ where $c_1, \ldots, c_k \in \{1, \ldots, d\}$. It evaluates to 0 for every assignment that simultanuously assigns c_1 to X_{i_1}, c_2 to X_{i_2}, ..., and c_k to X_{i_k} and to 1 for every remaining assignment. A (k, d)-CNF is a special case of a (k, d)-function. The problem (k, d)-CNF is to decide on input $H = (V, E)$, where E is a set of (k, d)-CNFs, whether there exists an assignment to the variables in V such that every $C \in E$ is satisfied. Thus, (k, d)-CNF is a special case of (k, d)-Function-SAT. We will treat (k, d)-CNF clauses as sets of k constraints, i.e. $(X_1 \neq c_1 \vee X_2 \neq c_2)$ denotes the same clause as $(X_2 \neq c_2 \vee X_1 \neq c_1)$.

In the paper we will prove the following main technical result.

Theorem 5. *Let k and d be fixed constants. Then there is $r = O(\frac{\log^2(1/\varepsilon) \cdot \log\log(1/\varepsilon)}{\varepsilon})$ such that the algorithm that samples r variables uniformly at random and accepts if and only if the induced subset of (k, d)-CNFs is satisfiable, is a property tester for the (k, d)-CNF problem.*

A simple argument given in [4] shows that this theorem implies Theorem 1. From now on, we will therefore focus on proving Theorem 5. We will also need the following definition of assigned variables.

Definition 6. *Given an instance (V, E) of (k, d)-CNF, a pair (U, Φ_U) with $U \subseteq V$ and $\Phi_U : U \to \{1, \ldots, d\}$ is called a set of assigned variables.*

III. Proof technique

We next describe the high-level proof technique to analyze canonical testers, which was introduced in [8] (and, independently in [3]) and which is based on earlier work in [11]. This technique was first designed to analyze canonical property testers for colorability problems and later used in the proof for testing satisfiability [4]. We will state it in the general setting of testing satisfiability. Recall that on input $H = (V, E)$ a canonical algorithm samples a set S of r variables uniformly at random and accepts, if and only if there is a satisfying assignment for $H[S]$. Since any subset of a satisfiable input is also satisfiable, we know that the algorithm accepts any satisfiable input $H = (V, E)$. Thus, we only need to prove that any input, which is ε-far from satisfiable, is rejected with probability at least $3/4$. Therefore, in the following, we will assume that the input $H = (V, E)$ is ε-far from satisfiable.

We will view the sample set S as being drawn sequentially and with repetition. The chance of variables being sampled multiple times only decreases our probability of success. Thus, our sample set S will be the sequence of variables $\langle s_1, \ldots, s_r \rangle$. We will also use $S^{(i)}$ to denote the sequence $\langle s_1, \ldots, s_i \rangle$. With slight abuse of notation, we will also use $S = S^{(r)}$ and $S^{(i)}$ to denote the sets $\{s_1, \ldots, s_r\}$ and $\{s_1, \ldots, s_i\}$, respectively.

In order to prove that the canonical tester rejects with probability $3/4$ on an ε-far input, we will consider the following slightly more complicated algorithm and argue that if this algorithm rejects a sample S, then so does the canonical tester. The input to the algorithm consists of the input instance (V, E) and the parameters $n = |V|, \varepsilon, d$ and k. The parameter r determines the sample size of the canonical tester and depends on ε, d and k. The parameter m is determined later in the analysis. It depends on ε, d and k and determines the length of a sequence of values that will be assigned to some of the variables of the random sample set. The construction is done in a way that if for every such sequence the subset of sample variables that have been assigned a value violates some constraint then the set of constraints induced by the sample variables is not satisfiable. More details follow below.

ProofOfSatisfiabilityTester$(V, E, n, \varepsilon, d, k)$

Sample a sequence of variables $S = \langle s_1, \ldots, s_r \rangle$

uniformly at random with repetition from V
for every sequence $\sigma = \langle c_1, \ldots, c_m \rangle$ of values from $\{1, \ldots, d\}$ **do**
  $j = 1$; $U_\sigma^{(0)} = \emptyset$; $\Phi_{U_\sigma^{(0)}}$ = empty assignment
  for $i = 1$ **to** r **do**
    $U_\sigma^{(i)} = U_\sigma^{(i-1)}$; $\Phi_{U_\sigma^{(i)}} = \Phi_{U_\sigma^{(i-1)}}$
    if SELECT$(U_\sigma^{(i-1)}, \Phi_{U_\sigma^{(i-1)}}, i, S^{(i)})$ **then**
      $U_\sigma^{(i)} = U_\sigma^{(i)} \cup \{s_i\}$;
      define $\Phi_{U_\sigma^{(i)}}(s_i) = c_j$; $j = j + 1$
  if $\Phi_{U_\sigma^{(r)}}$ is a satisfying assignment for $E[U_\sigma^{(r)}]$ **then**
    accept
reject

The algorithm aims at constructing a 'proof' that S is not satisfiable. Such a proof will consist of a collection $\{(U_\sigma, \Phi_{U_\sigma})\}_\sigma$ of subsets $U_\sigma \subseteq S$ such that for each assignment Φ_S of values to the variables in S, there exists a set U_σ such that Φ_S agrees with Φ_{U_σ} with respect to the assignment to the variables in U_σ, i.e. $\Phi_{S|U_\sigma} = \Phi_{U_\sigma}$, and Φ_{U_σ} is not satisfying. In this context, σ will be a sequence of values from $\{1, \ldots, d\}$ and given S any such sequence will uniquely identify the set U_σ and the assignment Φ_{U_σ} (but there maybe $\sigma \neq \sigma'$ with $U_\sigma = U_{\sigma'}$ and $\Phi_{U_\sigma} = \Phi_{U_{\sigma'}}$). Obviously, if for each assignment of values to S there exists a σ such that $H[U_\sigma]$ is violated for this assignment, then the union of the U_σ is not satisfiable (and so is S). In order to identify the variables in U_σ, for any σ the algorithm iterates over the sample sequence and constructs sets $U_\sigma^{(0)} := \emptyset, U_\sigma^{(1)}, \ldots, U_\sigma^{(r)} =: U_\sigma$ and the assignments $\Phi_{U_\sigma^{(0)}}, \Phi_{U_\sigma^{(1)}}, \ldots, \Phi_{U_\sigma^{(r)}} =: \Phi_{U_\sigma}$, where $\Phi_{U_\sigma^{(0)}}$ is the empty assignment and where here and in the remainder of the paper we use $U_\sigma^{(i)}$ and $\Phi_{U_\sigma^{(i)}}$ to refer to the final values of $U_\sigma^{(i)}$ and $\Phi_{U_\sigma^{(i)}}$, i.e. the values that have been obtained after the i-th iteration of the inner **for**-loop of algorithm PROOFOFSATISFIABILITYTESTER. The algorithm uses a *deterministic* subroutine SELECT to determine whether $U_\sigma^{(i)} = U_\sigma^{(i-1)}$ or $U_\sigma^{(i)} = U_\sigma^{(i-1)} \cup \{s_i\}$. In the latter case, we say that SELECT *adds* variable s_i. We define $\Phi_{U_\sigma^{(i)}}(X) = \Phi_{U_\sigma^{(i-1)}}(X)$ for all $X \in U_\sigma^{(i-1)}$ and, in the case that s_i is added, we define $\Phi_{U_\sigma^{(i)}}(s_i) = c_j$, where $j = |U_\sigma^{(i)}|$, i.e the number of variables accepted until step i. We will assume that SELECT never accepts if $|U_\sigma^{(i-1)}| = m$ as otherwise, the next assigned value would be undefined. The exact design of SELECT in our algorithms will not be required at this point and is the main technical challenge of the proof.

The following important observation was (implicitly) made in [8] and follows by induction and the fact that SELECT is deterministic.

Observation 7. *Let $\sigma = \langle c_1, \ldots, c_m \rangle$. The result of* SELECT$(U_\sigma^{(i-1)}, \Phi_{U_\sigma^{(i-1)}}, i, S^{(i)})$ *depends only on $S^{(i)}$ and the sequence $\langle c_1, \ldots, c_{k-1} \rangle$, where k is the current value of variable j used in the algorithm when* SELECT$(U_\sigma^{(i-1)}, \Phi_{U_\sigma^{(i-1)}}, i, S^{(i)})$ *is invoked.*

The following lemma has been implicitly proved in [8]. We include the simple proof for completeness.

Lemma 8. *If algorithm* PROOFOFSATISFIABILITYTESTER *rejects a sequence of variables S, then $E[S]$ is not satisfiable.*

Proof: Assume $E[S]$ is satisfiable with satisfying assignment Φ_S. We show that Φ_S can be used to construct a sequence of colors $\sigma = \langle c_1, \ldots, c_m \rangle$ for which algorithm PROOFOFSATISFIABILITYTESTER accepts. We inductively define a sequence σ' of colors as follows. If our current sequence $\sigma' = \langle c_1, \ldots, c_{k-1} \rangle$ has length $k-1$ and if s_i is the next variable added by SELECT, then $c_k = \Phi_S(s_i)$. If for a given k no variable is selected until the algorithm finishes, then we define σ to be any sequence that has σ' as a prefix. Note that by the observation above, the next variable (if it exists) is uniquely determined and so the construction is well-defined. Furthermore, observe that our construction ensures that Φ_U is always a satisfying assignment for U. Hence, the algorithm accepts. ■

In order to prove that the canonical algorithm rejects, by Lemma 8, it suffices to prove that PROOFOFSATISFIABILITYTESTER rejects. This can be done by proving that for a fixed sequence σ the acceptance probability is small and then use the union bound over all sequences. This approach has been used in the previous papers and we will require slightly more involved arguments.

Corollary 9. *If (for some values of m and r) algorithm* PROOFOFSATISFIABILITYTESTER *rejects every input $H = (V, E)$ that is ε-far from satisfiable with probability at least $3/4$ then the canonical tester is a property tester with sample complexity r.*

IV. POTENTIAL OF VARIABLES

Assigning values to some of the variables affects all clauses that contain some of these variables. For example, a clause $(X_{i_1} \neq c_1 \vee X_{i_2} \neq c_2 \vee \cdots \vee X_{i_k} \neq c_k)$ is satisfied, if, say, X_{i_1} is set to a value different from c_1. Similarly, if X_{i_1} is set to c_1 we can replace the clause by the smaller clause $(X_{i_2} \neq c_2 \vee \cdots \vee X_{i_k} \neq c_k)$. Given a set of assigned variables, we will consider the set of reduced constraints that can be deduced from E in a similar way as in the example above.

Definition 10. *Given a set E of constraints and a set of assigned variables (U, Φ_U), $U \subseteq V$, a clause $C = (X_{i_1} \neq c_1 \vee \cdots \vee X_{i_l} \neq c_l)$ is called a* reduced clause*, if there exists a clause $(X_{i_1} \neq c_1 \vee \cdots \vee X_{i_l} \neq c_l \vee X_{i_{l+1}} \neq c_{i_{l+1}} \vee \cdots \vee X_{i_k} \neq c_{i_k}) \in E$ with $\Phi_U(X_{i_j}) = c_{i_j}$ for $j = l+1, \ldots, k$. We use $E[U, \Phi_U]$ to denote the set of all reduced clauses of E for (U, Φ_U).*

In our proof we will use the set of reduced clauses to describe additional constraints imposed by a set of assigned variables on the assignment of the remaining variables. Note that if a constraint intersects more than one variable from U then we may have reduced clauses for any subset of the intersection. This will be required in the proof of Claim 17. Also note that Φ_U is not satisfying, if the empty clause is in $E[U, \Phi_U]$. There may be many clauses of length k in E that, under the given assignment of U, are reduced to the same clause C with $|C| = \ell < k$. In the following definition of the weight of a variable, we will therefore take care of this fact by weighting reduced clauses $C \in E[U, \Phi_U]$, $|C| = \ell$, by a factor of $n^{k-\ell}$. This weighting factor reflects the maximum possible number of witnesses in E for a given reduced constraint of arity ℓ.

Our next step is to define the *weight* weight(U, Φ_U, X_j) of a variable X_j with respect to a satisfying assignment Φ_U to the variables in U. The weight of a variable X_j reflects the minimum (weighted) number of new constraints that will be added to the current set $E[U, \Phi_U]$, if the assignment is extended to a satisfying assignment for the set of variables $U \cup X_j$.

Let $T(U, \Phi_U, X_j)$ be the set of values c such that there is no clause $(X_j \neq c) \in E[U, \Phi_U]$, i.e. the set of valid extensions of the current assignment Φ_U to the set $U \cup \{X_j\}$. The weight of a variable $X_j \notin U$ reflects the minimum (weighted) number of new constraints that will be added to $E[U, \Phi_U]$, when the assignment Φ_U is extended to an assignment $\Phi_{U\cup\{X_j\}}$ with $\Phi_{U\cup\{X_j\}}(X) = \Phi_U(X)$ for all $X \in U$ and $\Phi_{U\cup\{X_j\}}(X_j) \in T(U, \Phi_U, X_j)$. If $X_j \in U$ then the weight of X_j is 0. Note that the number of witnesses for a reduced constraint that includes X_j is at most $n^{k-\ell-1}$, i.e. a factor n smaller than the weight of the reduced constraint, which reflects the maximum number of overall witnesses. This will become important later in the proof.

Definition 11 (Weight of a variable)**.** *The weight weight(U, Φ_U, X_j) of a variable X_j with respect to a set of assigned variables (U, Φ_U) is defined as follows. If $X_j \in U$ then weight$(U, \Phi_U, X_j) = 0$. Otherwise, if $T(U, \Phi_U, X_j) = \emptyset$, then weight$(U, \Phi_U, X_j) = \infty$. Otherwise,*

$$\textit{weight}(U, \Phi_U, X_j) = d^k \cdot \min_{c\in T(U,\Phi_U,X_j)} \sum_{\ell=1}^{k-1} n^{k-\ell} \cdot \big|\{C \mid C \in E[U \cup \{X_j\}, \Phi_{U\cup\{X_j\},c}] \setminus E[U, \Phi_U] \textit{ and } |C| = \ell\}\big|,$$

where $\Phi_{U\cup X_j,c}$ is the extension of Φ_U to the set $U \cup \{X_j\}$ with $\Phi_{U\cup X_j}(X_j) = c$ and $\Phi_{U\cup X_j}(X) = \Phi_U(X)$ for $X \in U$.

Next, we will introduce the *potential* of a variable with respect to a sequence $S^{(i)}$, which for fixed σ will be the minimum weight of the variable attained during the processing of the sequence. We will show (and this is the main technical result) that if the assignment that is defined by $S^{(i)}$ and σ is satisfying, then the average potential of the variables is more than εn^k. Later we will use that the weight of a variable is at least its potential and that the potential is monotonically decreasing. The second property will be essential to obtain the improved sample size of $\widetilde{O}(1/\varepsilon)$.

Definition 12 (Potential of a variable X_j)**.** *The potential pot$(S^{(i)}, \sigma, X_j)$ of a variable X_j with respect to a sequence of variables $S^{(i)}$ and a sequence of values σ is defined as*

$$\textit{pot}(S^{(i)}, \sigma, X_j) = \min_{0\le h\le i} \textit{weight}(U_\sigma^{(h)}, \Phi_{U_\sigma^{(h)}}, X_j),$$

The average potential with respect to a sequence of variables $S^{(i)}$ and a sequence of values σ is defined as avgpot$(S^{(i)}, \sigma) = \sum_{j=1}^{n} \textit{pot}(S^{(i)}, \sigma, X_j)$.

The above definition depends implicitly on the function SELECT, since SELECT influences $U_\sigma^{(h)}$ and $\Phi_{U_\sigma^{(h)}}$. Since our SELECT procedure will depend on the potential, we have to make sure that no circularity is introduced. By Observation 7 this will not be the case, if SELECT$(U_\sigma^{(i-1)}, \Phi_{U_\sigma^{(i-1)}}, i, S^{(i)})$ depends only on pot$(S^{(i-1)}, \sigma, X_j)$.

The core idea behind the notion of the potential of a variable is that it allows us to deal with the effect that assigning variables may increase the weight of other variables. For example, if an assignment causes a new constraint of the form $(X_j \neq c)$ then this may lead to an increase of weight$(U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}, X_j)$, namely, in the case when the previous minimum in the definition of the weight was achieved by c. In contrast to the weight of a variable its potential is monotonically decreasing as i increases.

The interesting point is now that, if the input is ε-far from satisfiable and if $\Phi_{U_\sigma^{(r)}}$ is a satisfying assignment, then the average potential is more than εn^k. This is stated in the following key technical lemma whose proof is postponed to the next section.

Lemma 13. *If a set of constraints is ε-far from satisfiable then for any sequence of variables $S^{(r)}$ and sequence of values σ the following holds. If $\Phi_{U_\sigma^{(r)}}$ is a satisfying assignment for $U_\sigma^{(r)}$ then*

$$\textit{avgpot}(S^{(r)}, \sigma) > \varepsilon n^k.$$

V. PROOF OF THE MAIN RESULT

We will now prove Theorem 5. To do so, we will define a procedure SELECT such that algorithm PROOFOFSATISFIABILITYTESTER rejects any input that is ε-far from satisfiable with probability at least $3/4$. By Corollary 9 this is sufficient to show that the canonical tester is a property tester. We will start by giving the rough idea of how to define SELECT and then refine the definition further.

Lemma 13 tells us that for an arbitrary fixed satisfying assignment to some variables we have average potential of more than εn^k. Thus, in expectation a random variable will

have potential at least εn^k. Since the maximum potential of a variable is $O(n^k)$, this implies that a random vertex is likely to have high potential as well. We will define SELECT in such a way that it adds only variables with high potential. By the above argument we can conclude that SELECT adds variables reasonably often and we only need to argue that, for a fixed sequence σ, at some point i the assignment $\Phi_{U_\sigma^{(i)}}$ will not be a satisfying assignment with sufficiently high probability and then we can apply the union bound over all sequences. By the above discussion, we know that there is a good probability of SELECT adding a variable. Furthermore, we know that the potential of a variable is a lower bound on its weight and its weight is defined as the minimum increase in the weighted number of reduced constraints or ∞ if we cannot assign a value to this variable without violating a constraint from E. Thus, for example, if we guarantee that every added variable has a potential of more than $\varepsilon n^k/2$, we get that either more than $\varepsilon n^k/2$ weighted reduced constraints are added or we have proved that the current assignment is not satisfying. However, the maximum weighted number of constraints is $O(n^k)$ and so we cannot have more than $\varepsilon n^k/2$ new reduced constraints more than $O(1/\varepsilon)$ times and so at some point we will add a variable, which proves that the current assignment is not satisfying.

Simply defining SELECT to be adding variables of potential $\Omega(\varepsilon n^k)$ will not lead to an improvement of existing results. The reason is that we only know the expected potential of a variable to be more than εn^k. But this allows for extreme inputs that either have εn variables with potential n^k or n variables with potential εn^k or anything in between. We will therefore subdivide the sample sequence into $\lceil \log(1/\varepsilon) \rceil$ subsequences of length r_{sub} called stages. Thus, the sample size of the algorithm will be $r = \lceil \log(1/\varepsilon) \rceil \cdot r_{\text{sub}}$. Such an approach has been used before in [3] to obtain a tester with sample complexity $\widetilde{O}(1/\varepsilon)$ for the case of bipartiteness in graphs. The ℓ-th stage will be the subsequence $\langle s_{(\ell-1)\cdot r_{\text{sub}}+1}, \ldots, s_{\ell \cdot r_{\text{sub}}} \rangle$. During each stage we will use a different SELECT procedure. We use SELECT_ℓ to refer to the select procedure in stage ℓ. SELECT_ℓ will add s_i, if

(1) the current assignment $\Phi_{U_\sigma^{(i-1)}}$ is satisfying $E[U_\sigma^{(i-1)}]$, and
(2) $\text{pot}(S^{(i-1)}, \sigma, s_i) \geq n^k/2^{\ell+1}$.

An important observation is that for a fixed sequence of values σ in stage ℓ we can add at most $O(2^\ell)$ variables until the current assignment is violated. This follows from the fact that the potential of a variable is a lower bound for its weight, that the weight is a lower bound for the weighted number of newly added constraints and that the overall weighted number of constraints is $O(n^k)$.

We show the following lemma.

Lemma 14. *Let* $r_{sub} = 32 \cdot (k \cdot d^{2k} \cdot \ln(d) + \ln(10 \cdot \lceil \log(1/\varepsilon) \rceil)) \cdot \frac{8 \cdot d^k \cdot \lceil \log(1/\varepsilon) \rceil}{\varepsilon} = O(\log\log(1/\varepsilon) \cdot \log(1/\varepsilon)/\varepsilon)$. *At the end of stage ℓ, with probability at least $1 - \frac{1}{10 \cdot \lceil \log 1/\varepsilon \rceil}$, simultanuously for every sequence σ, we have*

(a) $\Phi_{U_\sigma^{(\ell \cdot r_{sub})}}$ is not satisfying $E[U_\sigma^{(\ell \cdot r_{sub})}]$, or
(b) there are fewer than $\frac{\varepsilon \cdot 2^\ell \cdot n}{8 \cdot d^k \cdot \lceil \log(1/\varepsilon) \rceil}$ variables X_j with $pot(S^{(\ell \cdot r_{sub})}, \sigma, X_j) \geq n^k/2^{\ell+1}$.

Proof: We will prove the result for a fixed σ and in the end apply the union bound. We will use $A(i)$ to denote the event that $\Phi_{U_\sigma^{(i)}}$ is not satisfying $E[U_\sigma^{(i)}]$ and $B(i)$ to denote the event that there are fewer than $\varepsilon \cdot 2^\ell \cdot n/(8d^k \lceil \log(1/\varepsilon) \rceil)$ variables X_j with $\text{pot}(S^{(i)}, \sigma, X_j) \geq n^k/2^{\ell+1}$. We consider the subsequence $\langle s_{(\ell-1)\cdot r_{\text{sub}}+1}, \ldots, s_{\ell \cdot r_{\text{sub}}} \rangle$. If for some index $i, (\ell-1) \cdot r_{\text{sub}} \leq i \leq \ell \cdot r_{\text{sub}}$ event $A(i)$ or $B(i)$ occurs, then by monotonicity $A(\ell \cdot r_{\text{sub}})$ or $B(\ell \cdot r_{\text{sub}})$ occurs at the end of the stage. Thus, we need to estimate the probability that both conditions are not satisfied throughout the whole stage. In order to do so, we first observe that

$$\mathbf{Pr}[\text{SELECT}_\ell \text{ adds } s_i \mid A(i-1) \text{ and } B(i-1) \text{ did not occur}] \geq \frac{\varepsilon \cdot 2^\ell}{8 \cdot d^k \cdot \lceil \log(1/\varepsilon) \rceil}.$$

Thus, as long as $A(i-1)$ and $B(i-1)$ are not satisfied, there is some good probability that SELECT adds the next variable. As our next step, we will prove that SELECT cannot add too many variables.

Claim 15. SELECT_ℓ *adds at most $k \cdot 2^{\ell+1} \cdot d^{2k}$ variables during stage ℓ.*

Proof: The number of (k,d)-CNF clauses is at most $d^k \cdot n^k$. If we weight any clause with $k' < k$ variables with a factor of $d^k \cdot n^{k-k'}$ as it is done in the definition of the weight of a variable, then we know that the weighted number of (k',d)-CNF clauses for $k' < k$ is at most $d^{2k} \cdot n^k$ and their sum (over all $1 \leq k' \leq k-1$) is at most $(k-1) \cdot d^{2k} \cdot n^k$. Furthermore, assigning a value to any variable added by SELECT_ℓ will increase the weighted number of constraints by at least $n^k/2^{\ell+1}$ or the assignment $\Phi_{U_\sigma^{(i)}}$ will no longer be satisfying. Thus, we conclude that at most $(k-1) \cdot 2^{\ell+1} \cdot d^{2k} + 1 \leq k \cdot 2^{\ell+1} \cdot d^{2k}$ variables are added during stage ℓ. ■

Corollary 16. *The sum of variables added by* $\text{SELECT}_1, \ldots, \text{SELECT}_\ell$ *is at most $k \cdot 2^{\ell+2} \cdot d^{2k}$.*

Note that the above corollary also allows us to define the value of m used in PROOFOFSATISFIABILITYTESTER as $m = k \cdot 2^{\lceil \log(1/\varepsilon) \rceil + 2} \cdot d^{2k}$.

Define Z_i to be the $0-1$-random variable, that is 1, if events $A(i-1+(\ell-1) \cdot r_{\text{sub}})$ and $B(i-1+(\ell-1) \cdot r_{\text{sub}})$ did not occur and SELECT_ℓ adds $s_{i+(\ell-1) \cdot r_{\text{sub}}}$ or if event $A(i-1+(\ell-1) \cdot r_{\text{sub}})$ or $B(i-1+(\ell-1) \cdot r_{\text{sub}})$ occured; $Z_i = 0$, otherwise.

By Claim 15, if $\sum_{i=1}^{r_{\text{sub}}} Z_i > k \cdot 2^{\ell+1} \cdot d^{2k}$ then condition (a) or (b) of Lemma 14 is satisfied at the end of stage ℓ. Let Y_i be a $0-1-$random variable that is 1 with probability $\frac{\varepsilon \cdot 2^\ell}{8 \cdot d^k \cdot \lceil \log(1/\varepsilon) \rceil}$. Clearly, for any x we have $\mathbf{Pr}[\sum_{i=1}^{r_{\text{sub}}} Y_i \geq x] \leq \mathbf{Pr}[\sum_{i=1}^{r_{\text{sub}}} Z_i \geq x]$. Thus, we know that the probability that condition (a) or (b) is satisfied is at least $\mathbf{Pr}[\sum_{i=1}^{r_{\text{sub}}} Y_i > k \cdot 2^{\ell+1} \cdot d^{2k}]$. Setting $r_{\text{sub}} = 32 \cdot (k \cdot d^{2k} \cdot \ln(d) + \ln(10 \cdot \lceil \log(1/\varepsilon) \rceil)) \cdot \frac{8 \cdot d^k \cdot \lceil \log(1/\varepsilon) \rceil}{\varepsilon} = O(\log\log(1/\varepsilon) \cdot \log(1/\varepsilon)/\varepsilon)$ to be the number of variables used in stage ℓ we get $\mathbf{E}[\sum_{i=1}^{r_{\text{sub}}} Y_i] = 32 \cdot (k \cdot d^{2k} \cdot \ln(d) + \ln(10 \cdot \lceil \log(1/\varepsilon) \rceil)) \cdot 2^\ell$. Chernoff's bound implies

$$\begin{aligned}
\mathbf{Pr}[\sum_{i=1}^{r_{\text{sub}}} Y_i \leq k \cdot d^{2k} \cdot 2^{\ell+1}] &\leq \mathbf{Pr}[\sum_{i=1}^{r_{\text{sub}}} Y_i \leq \frac{1}{2}\mathbf{E}[\sum_{i=1}^{r_\ell} Y_i]] \\
&\leq \exp\left(\frac{1}{8} \cdot \mathbf{E}[\sum_{i=1}^{r_\ell} Y_i]\right) \\
&\leq \frac{1}{10 \cdot \lceil \log 1/\varepsilon \rceil} \cdot \frac{1}{d^{k \cdot 2^{\ell+2} \cdot d^{2k}}}
\end{aligned}$$

By Corollary 16 the sum of variables added until the end of stage ℓ is at most $k \cdot 2^{\ell+2} \cdot d^{2k}$. Hence, we need to take the union bound over $d^{k \cdot 2^{\ell+2} \cdot d^{2k}}$ possible (prefixes of) σ and the lemma follows. ∎

Corollary 9 implies that in order to prove Theorem 5 we need to show that algorithm PROOFOFSATISFIABILITYTESTER with sample size $r = \lceil \log(1/\varepsilon) \rceil r_{\text{sub}}$ rejects an arbitrary input that is ε-far from satisfiable with probability at least $3/4$. Therefore, let us now assume that our input set E is ε-far from satisfiable. Using the union bound we obtain that with probability at least $9/10$ the statement of Lemma 14 is true for all stages of the algorithm. We will show that conditioned on the event that the statement of Lemma 14 is true for all stages of the algorithm, it will reject. Indeed, assume for the sake of contradiction that the algorithm accepts. Then there exists a sequence σ such that $\Phi_{U_\sigma^{(r)}}$ is a satisfying assignment. This implies that at the end of each stage ℓ , $1 \leq \ell \leq \lceil \log(1/\varepsilon) \rceil$, the assignment $\Phi_{U_\sigma^{(\ell \cdot r_{\text{sub}})}}$ was satisfying and so statement (b) was true at the end of stage ℓ. By the monotonicity of the potential, we can conclude that for every value of ℓ, $1 \leq \ell \leq \lceil \log(1/\varepsilon) \rceil$, the same statement is still true at the end of the last stage. This implies that after the last stage, there are fewer than $2 \cdot \varepsilon \cdot n/(8d^k \lceil \log(1/\varepsilon) \rceil)$ variables with a potential of at most $d^k \cdot n^k$. Furthermore, by monotonicity for every ℓ, $2 \leq \ell \leq \lceil \log(1/\varepsilon) \rceil$, there are fewer than $\varepsilon \cdot 2^\ell \cdot n/(8d^k \lceil \log(1/\varepsilon) \rceil)$ variables with a potential between $n^k/2^\ell - 1$ and $n^k/2^{\ell+1}$. For the at most n remaining we know that their potential is at most $n^k/2^{\lceil \log(1/\varepsilon) \rceil + 1}$.

Thus,

$$\begin{aligned}
\text{avgpot}(S^{(r)}, \sigma) &\leq \frac{1}{n} \cdot 2 \cdot \varepsilon \cdot \frac{n}{8d^k \lceil \log(1/\varepsilon) \rceil} \cdot d^k \cdot n^k \\
&+ \frac{1}{n} \cdot \sum_{\ell=2}^{\lceil \log(1/\varepsilon) \rceil} \frac{n^k}{2^\ell} \cdot \varepsilon \cdot 2^\ell \cdot \frac{n}{8d^k \lceil \log\ (1/\varepsilon) \rceil} \\
&+ \frac{1}{n} \cdot n \cdot \frac{n^k}{2^{\lceil \log(1/\varepsilon) \rceil + 1}} \\
&\leq \varepsilon n^k.
\end{aligned}$$

Now Lemma 13 implies that the input is not ε-far from satisfiable, which is a contradiction. Hence, with probability at least $9/10$ the canonical tester rejects and is therefore a property tester. The sample complexity of the tester is $r_{\text{sub}} \cdot \lceil \log(1/\varepsilon) \rceil = O(\log\log(1/\varepsilon) \cdot \log^2(1/\varepsilon)/\varepsilon)$. This finishes the proof of Theorem 5.

VI. PROOF OF LEMMA 13

Proof: The proof is by contradiction. We assume that our input set of constraints (V, E) is ε-far from satisfiable but we have a sequence of variables $S^{(r)}$ and a sequence of values σ such that $\Phi_{U_\sigma^{(r)}}$ is a satisfying assignment for $U_\sigma^{(r)}$ and $\text{avgpot}(S^{(r)}, \sigma) \leq \varepsilon n^k$. We will show that in this case we can construct an assignment $\chi : V \to \{1, \ldots, d\}$ that violates at most εn^k constraints from E. This will contradict the assumption that the input set is ε-far from satisfiable.

In order to construct χ we will define the *time stamp* of a variable X_j to be the smallest number $i \in \{1, \ldots, r\}$ such that $\text{weight}(U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}, X_j) = \text{pot}(S^{(r)}, \sigma, X_j)$, i.e. the first step i when the weight of the variable X_j is smallest. Let us now start with the construction of χ by defining $\chi(X) = \Phi_{U_\sigma^{(r)}}$ for all $X \in U_\sigma^{(r)}$. In order to construct the assignment for the remaining variables, we proceed in order of increasing time stamps. We assign to every variable X_j with time stamp i the value that minimizes the weighted number of new constraints with respect to $(U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}})$, i.e. we get $\chi(X_j) = c$, where $c \in T(U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}, X_j)$ is a value that minimizes $\sum_{\ell=1}^{k-1} n^{k-\ell} \cdot \left|\{C \mid C \in E[U \cup X_j, \Phi_{U \cup X_j, c}] \setminus E[U, \Phi_U] \text{ and } |C| = \ell\}\right|$. Note that the minimum in the above expression is equal to $\text{weight}(U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}, X_j)$.

We will now remove a set of at most $\text{avgpot}(S^{(r)}, \sigma)$ many constraints from E and show that χ is a satisfying assignment for the resulting set, which we call E'. For each variable X_j with time stamp i and $\chi(X_j) = c$ we remove all constraints in E that lead to reduced clauses in $E[U_\sigma^{(i)} \cup X_j, \Phi_{U_\sigma^{(i)} \cup X_j, c}] \setminus E[U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}]$. For a fixed reduced constraint $C \in E[U_\sigma^{(i)} \cup X_j, \Phi_{U_\sigma^{(i)} \cup X_j, c}] \setminus E[U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}]$ the number of constraints that we have to remove from E is at most $d^k \cdot n^{k-|C|-1}$ since each constraint from E has to include X_j and the variables of C and there are at most d^k constraints on the same set of k variables. Thus, the overall number of constraints that we have to

remove is at most $\sum_{\ell=1}^{k-1} d^k \cdot n^{k-\ell-1} \cdot \left|\{C \mid C \in E[U \cup X_j, \Phi_{U \cup X_j, c}] \setminus E[U, \Phi_U] \text{ and } |C| = \ell\}\right|$, which is $\frac{1}{n} \cdot \text{weight}(U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}, X_j)$. By the choice of the time stamp we also have $\frac{1}{n} \cdot \text{weight}(U_\sigma^{(i)}, \Phi_{U_\sigma^{i)}}, X_j) = \frac{1}{n} \cdot \text{pot}(S^{(r)}, \sigma, X_j)$. Summing up over all variables yields that the number of constraints removed from E is at most $\text{avgpot}(S^{(r)}, \sigma)$, which is at most εn^k by our assumption. Thus, if we can prove that the assignment χ satisfies all constraints in E' then we have arrived at the desired contradiction. For this purpose let us consider an arbitrary $i, 1 \leq i \leq r$. Let us use $V[i]$ to denote the set of variables with time stamp at most i. Our first claim is that the set of reduced constraints that can be derived from assigning values to all variables with time stamp at most i is a subset of $E[U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}]$.

Claim 17. *For any $i, 1 \leq i \leq r$, we have $E'[V[i], \chi_{V|V[i]}] \subseteq E[U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}]$.*

Proof: Assume the claim is not true and let i be the minimum value for which the statement of the lemma is false. Then there is a reduced clause $C \in E'[V[i], \chi_{|V[i]}]$ that is not in $E[U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}]$. This means that there must be a variable X_j with time stamp i involved in creating C, because $E'[V[i-1], \chi_{|V[i-1]}] \subseteq E[U_\sigma^{(i-1)}, \Phi_{U_\sigma^{(i-1)}}] \subseteq E[U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}]$. However, when X_j is assigned its value c we remove all constraints from E that lead to reduced clauses in $E[U_\sigma^{(i)} \cup X_j, \Phi_{U_\sigma^{(i)} \cup X_j, c}] \setminus E[U_\sigma^{(i)}, \Phi_{U_\sigma^{(i)}}]$. Hence, C was deleted in this step, which is a contradiction. ■

Once we have established the above claim, it is easy to show that χ is a satisfying assignment for E'. Let us consider an arbitrary constraint $C \in E'$ and let X_j be the last variable in C that is assigned a value. By the previous claim and the choice of the time stamp, we know that we can assign X_j the value $\chi(X_j)$ as otherwise the weight of X_j would be infinite, which contradicts the choice of the time stamp as the potential is always finite. Furthermore, since X_j is the last variable of C that is assigned a value, we know that χ satisfies C. ■

VII. Constructing good solutions for satisfiable instances

In this section we will argue that, given a satisfiable instance of (k, d)-Function-SAT, it is possible to efficiently construct an assignment to the variables that violates no more than εn^k constraints. First, we observe that we can apply the same arguments as in the proof of our main result to show that for each sequence σ for which $\Phi_{U_\sigma^{(r)}}$ is a satisfying assignment, in each of our stages only relatively few variables with high potential are left and so the average potential of the assigned variables defined as in the proof is at most εn^k. This implies that an assignment as constructed in the proof will violate at most εn^k constraints. Furthermore, we know that such a σ exists since the input set is satisfiable.

We next describe a simple (and slow) way to compute a solution that violates at most εn^k constraints. For this purpose, we iterate over all subsequences of length $m = O(1/\varepsilon)$ of our sample sequence. For each of them, we check all d^m possible assignments. For each such assignment we then compute a solution as described in the proof. This requires us to calculate the time stamp for each variable, which in turn requires its weight at each point of time. Computing the weight of a variable requires to iterate over the $O(n^{k-1})$ possible new reduced constraints and check in $\tilde{O}(1/\varepsilon^k)$ time whether this constraint is already a reduced constraint at the current point of time. Computing the value of the variable can then be done in $O(m)$ time. After the values of all variables have been determined, we can compute in $O(n^k)$ time the number of violated constraints. We return the assignment that violates the fewest number of constraints. As argued above, this assignment will violate at most εn^k constraints. Overall, this procedure requires $O(n^k \cdot 2^{\tilde{O}(1/\varepsilon)})$ time.

It remains to speed up the algorithm. For this purpose, we will sketch an algorithm that constructs an assignment which violates at most $O(\varepsilon n^k)$ constraints. Then we can use a smaller value of ε to get the same guarantee as above. There are two steps that require more than linear time in the above procedure. We will show that both can be replaced by a 'sampling version' to achieve linear running time. Let us start with the second one. For a given assignment we need to compute the number of violated constraints. Since we aim at an approximation with additive error εn^k, it suffices to sample $O(r/\varepsilon^2)$ constraints to ensure that with sufficiently high probability the number of violated constraints is approximated for all subsequences upto an additive error of εn^k. If we use this approximation instead of the exact count, we will still end up with an additive error of $O(\varepsilon n^k)$. Thus, we can reduce the running time of the second step from $O(n^k)$ to $O(r/\varepsilon^2) = O(1/\varepsilon^{O(1)})$. To reduce the running time of the first step, we observe that for our purposes, it suffices to approximate the weight of a vertex with an additive error of upto εn^k. Again, such an error will still allow us to end up with an overall additive error of $O(\varepsilon n^k)$. To approximate the weight of a vertex, we can estimate the number of new reduced constraints by iterating over the $k-1$ possible arities and for each arity ℓ sampling random subsets of ℓ variables and compute the number of new reduced constraints on them. For each subset this can be done in $O(1/\varepsilon^{O(1)})$ time for constant d and k as checking which constraints are present on the subset requires to look at all constraints that have only variables from the subset and the sample set. Thus, we can also estimate the weights of the variables in $O(1/\varepsilon^{O(1)})$ time per variable. Thus, for each subsequence we can construct an assignment in $O(n/\varepsilon^{O(1)})$ time, which overall gives a running time of $O(n \cdot 2^{\tilde{O}(1/\varepsilon)})$.

To further reduce the running time, we observe that we are only using $O(r/\varepsilon^2)$ constraints to evaluate the quality of the current assignment. Thus, it suffices to only compute the value of these variables. This way, for a fixed subsequence, we can approximate in $O(1/\varepsilon^{O(1)})$ time the quality of the induced assignment upto an additive error of $O(\varepsilon n^k)$. Thus, we can determine in $2^{\tilde{O}(1/\varepsilon)}$ time a subsequence that produces an assignment which violates at most $O(\varepsilon n^k)$ constraints. This assignment can then be computed in $O(n/\varepsilon^{O(1)} + 2^{\tilde{O}(1/\varepsilon)})$ time for constant d and k.

Theorem 18. *Given access to a satisfiable instance of (k,d)-Function-SAT, one can construct in $O(n/\varepsilon^{O(1)} + 2^{\tilde{O}(1/\varepsilon)})$ time an assignment that with probability at most $3/4$ violates at most εn^k constraints.*

VIII. Conclusions

In the previous section we showed that one can find a good solution based on sampling, if the input instance is satisfiable. Unfortunately, the techniques we use do not seem to carry over to the case of finding a near optimal solution in the case that the instance is not satisfiable. The problem is that it is unclear how to estimate the number of constraints violated by a solution induced by the sample. In the satisfiable case, we can always use that there exists at least one solution that does not violate any constraints. We nevertheless hope that our technique will be a first step to improving the time complexity of MAX-(k,d)-Function-SAT to something similar to the above running time.

In terms of property testing, an obvious question is to tighten the above bounds. Another interesting question is to use adaptive sampling to improve the bounds (for example, in the case of bipartiteness) or prove a lower bound for adaptive testers. For the case of 3-coloring the author believes that a lower bound of $\Omega(1/\varepsilon^2)$ on the number of entries in the adjacency matrix that needs to be queried is plausible.

IX. Acknowledgements

The author thanks Inge Li Goertz and Artur Czumaj for some preliminary discussion on the problem and the anonymous reviewers of STOC'12 and FOCS'12 for carefully reading the paper and their helpful comments regarding to the presentation.

References

[1] G. Andersson, L. Engebretsen. Property testers for dense constraint satisfaction programs on finite domains. Random Structures & Algorithms, 21(1): 14-32, 2002.

[2] N. Alon, W. Fernandez de la Vega, R. Kannan and M. Karpinski. Random sampling and approximation of MAX-CSPs. Journal of Computer and System Sciences, 67(2):212-243, 2003.

[3] N. Alon and M. Krivelevich. Testing k-colorability. SIAM J. Discrete Math., 15, pp. 211-227, 2002.

[4] N. Alon and A. Shapira. Testing Satisfiability. Journal of Algorithms, 47, pp. 87-103, 2003.

[5] L. Avigad, O. Goldreich. Testing Graph Blow-Up. APPROX-RANDOM, pp. 389-399, 2011.

[6] A. Bogdanov, L. Trevisan. Lower Bounds for Testing Bipartiteness in Dense Graphs. IEEE Conference on Computational Complexity, pp. 75-81, 2004.

[7] B. Bollobás, P. Erdös, M. Simonovits and E. Szemerédi. Extremal graphs without large forbidden subgraphs. Annals of Discrete Mathematics 3 (1978), 2941.

[8] A. Czumaj and C. Sohler. Testing hypergraph colorability. Theor. Comput. Sci., 331(1): 37-52, 2005.

[9] A. Czumaj, C. Sohler. Abstract Combinatorial Programs and Efficient Property Testers. SIAM Journal on Computing, 34(3): 580-615, 2005.

[10] A. Frieze and R. Kannan. Quick Approximation to Matrices and Applications. Combinatorica, 19(2): 175-220, 1999.

[11] O. Goldreich, S. Goldwasser and D. Ron. Property Testing and its Connection to Learning and Approximation. J. ACM, 45(4): 653-750, 1998.

[12] O. Goldreich and Dana Ron. Algorithmic Aspects of Property Testing in the Dense Graphs Model. SIAM Journal on Computing, 40(2): 376-445, 2011.

[13] O. Goldreich and L. Trevisan. Three Theorems Regarding Testing Graph Properties. FOCS, pp. 460-469, 2001.

[14] M. Gonen and D. Ron. On the Benefits of Adaptivity in Property Testing of Dense Graphs. APPROX-RANDOM, pp. 525-539, 2007.

[15] M.Karpinski, W. Schudy. Personal communication with Marek Karpinski, 2011.

[16] V. Rödl and R. Duke. On graphs with small subgraphs of large chromatic number. Graphs and Combinatorics 1, pp. 91-96, 1985.

[17] R. Rubinfeld and A. Shapira. Sublinear Time Algorithms. SIAM Journal on Discrete Math, to appear.

[18] R. Rubinfeld and M. Sudan, Robust characterization of polynomials with applications to program testing. SIAM Journal of Computing, 25: 252–271, 1996.