Lower Bounds for Testing Triangle-freeness in Boolean Functions*

Arnab Bhattacharyya† Ning Xie‡

Abstract

Let $f_1, f_2, f_3 : \mathbb{F}_2^n \to \{0,1\}$ be three Boolean functions. We say a triple $(x, y, x+y)$ is a *triangle* in the function-triple (f_1, f_2, f_3) if $f_1(x) = f_2(y) = f_3(x+y) = 1$. (f_1, f_2, f_3) is said to be *triangle-free* if there is no triangle in the triple. The distance between a function-triple and triangle-freeness is the minimum fraction of function values one needs to modify in order to make the function-triple triangle-free. A *canonical tester* for triangle-freeness repeatedly picks x and y uniformly and independently at random and checks if $f_1(x) = f_2(y) = f_3(x+y) = 1$. Based on an algebraic regularity lemma, Green showed that the number of queries for the canonical testing algorithm is upper-bounded by a tower of 2's whose height is polynomial in $1/\epsilon$. A trivial query complexity lower bound of $\Omega(1/\epsilon)$ is straightforward to show. In this paper, we give the first non-trivial query complexity lower bound for testing triangle-freeness in Boolean functions. We show that, for every small enough ϵ there exists an integer $n_0(\epsilon)$ such that for all $n \geq n_0$ there exists a function-triple $f_1, f_2, f_3 : \mathbb{F}_2^n \to \{0,1\}$ depending on all the n variables which is ϵ-far from being triangle-free and requires $(\frac{1}{\epsilon})^{4.847\cdots}$ queries for the canonical tester. For the single function case that $f_1 = f_2 = f_3$, we obtain a weaker lower bound of $(\frac{1}{\epsilon})^{3.409\cdots}$. We also show that the query complexity of any general (possibly adaptive) one-sided tester for triangle-freeness is at least square-root of the query complexity of the corresponding canonical tester. Consequently, this yields $(1/\epsilon)^{2.423\cdots}$ and $(1/\epsilon)^{1.704\cdots}$ query complexity lower bounds for multi-function and single-function triangle-freeness respectively, with respect to general one-sided testers.

1 Introduction

Roughly speaking, property testing is concerned with the existence of an efficient algorithm that queries an input object a small number of times and decides correctly with high probability whether the object has a given property or whether it is "far away" from having the property. Formally, let D be a finite domain and R be a finite range. Letting $\{D \to R\}$ denote the set of all functions from D to R, a *property* is specified by a family $\mathcal{F} \subseteq \{D \to R\}$ of functions. A *tester* is a randomized algorithm which is given a distance parameter ϵ and has oracle access to an input function $f : D \to R$, and accepts with probability at least 2/3 if $f \in \mathcal{F}$ and rejects (also with probability at least 2/3) if the function is ϵ-*far* from $\mathcal{F}$. Distance between functions $f, g : D \to R$, denoted $\mathrm{dist}(f, g)$, is simply the fraction of the domain where f and g disagree, and $\mathrm{dist}(f, \mathcal{F}) = \min_{g \in \mathcal{F}}\{\mathrm{dist}(f, g)\}$. For $\epsilon \in (0, 1)$, we say f is ϵ-far from $\mathcal{F}$ if $\mathrm{dist}(f, \mathcal{F}) \geq \epsilon$ and ϵ-close otherwise. A tester is *one-sided* if whenever $f \in \mathcal{F}$, the tester accepts with probability 1. The central parameter associated with a tester is the number of oracle queries it makes to the function f being tested. In particular, a property is called *strongly testable* if, for every fixed ϵ, there is a tester with query complexity that depends only on the distance parameter ϵ and is independent of the size of the domain. Property testing was formally defined by Rubinfeld and Sudan [29], and the systematic exploration of property testing for combinatorial properties was initiated by Goldreich, Goldwasser, and Ron [16]. Subsequently, a rich collection of properties have been shown to be strongly testable [8, 7, 3, 13, 27, 5, 4, 22, 21].

A central quest of research in property testing has been to characterize properties according to their query complexity. One can ask, for example, whether a large class of properties are all strongly testable, and how the query complexity of a strongly testable property depends on the distance parameter ϵ. Such broad understanding of testability has been achieved for graph and hypergraph properties. For graph properties, it is known exactly ([3, 13]) which properties are strongly testable in the dense graph model. Furthermore, for an important class of properties, H-freeness for fixed subgraphs H, it is known exactly for which H, testing H-freeness requires the query complexity to be super-polynomial in $1/\epsilon$ (ϵ being the distance parameter) and for which only a polynomial number of queries suffice: This was proved by Alon [1] for one-sided testers and

*A full version of this paper is available as a technical report at http://eccc.hpi-web.de/report/2009/066/.

†CSAIL, MIT. Research supported in part by a DOE Computational Science Graduate Fellowship and NSF Awards 0514771, 0728645 and 0732334.

‡CSAIL, MIT. Research supported by NSF Awards 0514771, 0728645 and 0732334. Part of the work done while visiting ITCS, Tsinghua University and supported by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

by Alon and Shapira that [6] for general (two-sided) testers. Progress toward similar understanding has also been made for hypergraph properties [28, 9, 7].

Somewhat ironically, algebraic properties, the main objects of study in the seminal work of Rubinfeld and Sudan [29], are not as well understood as (hyper)graph properties from a high-level perspective. On the one hand, there has been a lot of work in constructing low-query testers for specific algebraic properties, such as linearity and membership in various error-correcting codes. However, the systematic study of the query complexity of algebraic properties began only recently with the work of Kaufman and Sudan [23]. Formally, the class of properties under consideration here are linear-invariant properties. In this setting [1], the domain $D = \mathbb{F}_2^n$ and range $R = \{0,1\}$, where $\mathbb{F}_2$ is the finite field with two elements. A property $\mathcal{F}$ is said to be *linear-invariant* if for every $f \in \mathcal{F}$ and linear map $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$, it holds that $f \circ L \in \mathcal{F}$. Roughly speaking, Kaufman and Sudan showed strong testability of any locally-characterized linear-invariant and *linear*[2] property. Moreover, the query complexity of all such properties is only $poly(1/\epsilon)$. Nonlinear linear-invariant properties were studied formally in [12] where the authors isolated a particular class of nonlinear linear-invariant properties, $\mathcal{M}$-freeness for some fixed binary matroids $\mathcal{M}$, and showed an infinitely large set of strongly testable $\mathcal{M}$-freeness properties. Subsequently, Shapira [30] and Král *et al* [24] independently showed that, in fact for any fixed binary matroid $\mathcal{M}$, $\mathcal{M}$-freeness is strongly testable, mirroring the analogous result of subgraph-freeness testing. However, unlike the case of graphs where it is known exactly which subgraph-freeness properties can be tested in time $poly(1/\epsilon)$ and which cannot, there are no similar results known for matroid-freeness properties. Indeed, to the best of our knowledge, prior to our work, there were no non-trivial lower bounds known for the query complexity (in terms of ϵ) for any natural linear-invariant algebraic property.

1.1 Our Results We are interested in the property of triangle-freeness for Boolean functions. Let $f_1, f_2, f_3 : \mathbb{F}_2^n \to \{0,1\}$ be three Boolean functions. We say a triple $(x, y, x+y)$ is a *triangle* in the function-triple (f_1, f_2, f_3) if $f_1(x) = f_2(y) = f_3(x+y) = 1$. (f_1, f_2, f_3) is said to be *triangle-free* if there is no triangle among the three functions. The *canonical tester* for triangle-freeness repeatedly picks x and y uniformly and independently at random and checks if $f_1(x) = f_2(y) = f_3(x+y) = 1$. Note that the canonical tester is a one-sided tester. Moreover, if the number of triangles is N_Δ, then to reject a function-triple that is ϵ-far from being triangle-free with constant probability, the canonical tester needs to make $\Omega(\frac{2^{2n}}{N_\Delta})$ number of queries. Green [19] showed that the canonical tester for the property of triangle-freeness does indeed work correctly, though the analysis is quite different from that of typical algebraic tests and is more reminiscent of the analysis for tests of graph properties. In particular, Green developed an algebraic regularity lemma for the Boolean cube (his result is much more general, in fact, it works for any abelian group). The query complexity upper bound proved by Green has a huge dependency on ϵ: it is a tower of 2's with height polynomial in $1/\epsilon$. A more combinatorial version of Green's result is that, for any function-triple ϵ-far from being triangle-free, there are at least $\delta(\epsilon)2^{2n}$ triangles in the function-triple, though this $\delta(\epsilon)$ is only proved to be super tiny. A trivial lower bound of $\Omega(1/\epsilon)$ is straightforward to show. But, to the best of our knowledge, there is no non-trivial lower bound for testing triangle-freeness in Boolean functions. This question was left open in [19].

It is interesting to compare the testability of algebraic triangle-freeness and graphic triangle-freeness. Using Szemerédi's regularity lemma, triangle-freeness in graphs is known to be testable with a tower-type query complexity upper bound. Alon [1] gave a super-polynomial query complexity lower bound and it is the strongest query lower bound for a natural strongly testable property known to date. However, the proof technique in [1] does not seem to directly apply to the algebraic setting due to the inherent additive structures of the Boolean cubes. More generally, it seems to us that proving lower bounds for the Boolean function case is more challenging than that of the graphic case. For example, the lower bound given in [19] for regularity partitioning of the Boolean cube, though being of tower-type, is much weaker than its graphic counterpart shown in [18] (and is also weaker than the upper bound proved in the same paper).

In this paper we give the first non-trivial query lower bounds for testing triangle-freeness in Boolean functions. In particular, we show that, for every small enough ϵ there exists an integer $n_0(\epsilon)$ such that for all $n \geq n_0$ there exists a function-triple $f_1, f_2, f_3 : \mathbb{F}_2^n \to \{0,1\}$ depending on all the n variables which is ϵ far from being triangle-free and requires $(\frac{1}{\epsilon})^{4.847\cdots}$ queries for the canonical tester. For the single function case that $f_1 = f_2 = f_3$, we obtain a weaker lower bound of $(\frac{1}{\epsilon})^{3.409\cdots}$ for the canonical tester.

The goal of this research should be to understand

[1] [23] considers linear invariance over general fields. In this paper, we restrict ourselves to $\mathbb{F}_2^n$ for simplicity.

[2] A property $\mathcal{F}$ is linear if for any f and g that are in $\mathcal{F}$ necessarily implies that $f + g$ is in $\mathcal{F}$.

the query complexity with respect to *general testers*. To this end, we show that if there is a one-sided, possibly adaptive tester for triangle-freeness with query complexity q, then one can transform that tester into a canonical one with query complexity at most $O(q^2)$. Combining with our results for canonical testers, this implies a query complexity lower bound of $(\frac{1}{\epsilon})^{2.423\cdots}$ for the multi-function triangle-freeness problem and of $(\frac{1}{\epsilon})^{1.704\cdots}$ for single-function triangle-freeness, with respect to one-sided testers. In fact our result is a bit more general: we prove a polynomial relationship between the query complexity of the canonical tester and arbitrary one-sided testers, for any matroid-freeness property. This is analogous to a result in [2] for one-sided testers of subgraph-freeness in graphs [3]. Another related result is that of Ben-Sasson, Harsha and Raskhodnikova [11] who showed that there is no gap between the query complexities of adaptive testers and non-adaptive ones for testing linear properties.

1.2 Techniques From a combinatorial point of view, proving a lower bound for the query complexity of the canonical tester for triangle-freeness amounts to constructing functions or function-triples which are far from being triangle-free but contain only a small number of triangles.

Our lower bound for function-triples is based on constructing a *vertex-disjoint* function-triple, meaning that all the triangles in the triple are *pairwise disjoint*. The property of being vertex-disjoint makes it simple to calculate the function-triple's distance from triangle-freeness as well as counting the number of triangles within the function-triple. We start our construction of a vertex-disjoint function-triple from three sets, each of cardinality m, of k-bit binary vectors, $\{a_i\}_{i=1}^m$, $\{b_j\}_{j=1}^m$ and$\{c_\ell\}_{\ell=1}^m$, where k and m are fixed integers. Next we define three sets, $\{A_I\}$, $\{B_J\}$ and$\{C_L\}$, of mk-bit vectors, each consisting of the vectors obtained by concatenating $\{a_i\}$, $\{b_j\}$ and$\{c_\ell\}$, respectively, in all possible orders. Finally we define our function-triple (f_A, f_B, f_C) to be the characteristic functions of the three sets $\{A_I\}$, $\{B_J\}$ and$\{C_L\}$. In order to make the triangles in this function-triple pairwise disjoint, we impose the constraint that $\{a_i\}$, $\{b_j\}$ and$\{c_\ell\}$ satisfy the 1-perfect-matching-free (1-PMF for short) property (see Section 4.1 for formal definition). To make this construction work for arbitrarily small ϵ, we concatenate with some $n' \geq 1$ copies of each $\{a_i\}$, $\{b_j\}$ and$\{c_\ell\}$ and require them to satisfy the n'-PMF property for any $n' \geq 1$. It turns out that $\{a_i\}$, $\{b_j\}$ and$\{c_\ell\}$ being PMF is equivalent to a (small) set of homogeneous Diophantine linear equations having no non-trivial solution, which in turn can be checked by linear programming. Our numerical computation indicates the existence of PMF family of vectors for $k = 3, 4$, and 5. Our findings show that larger values of k give stronger lower bounds but unfortunately it was computationally infeasible to search for PMF families of vectors for $k \geq 6$. We conjecture that our approach may lead to super-polynomial query lower bounds for testing multi-function triangle-freeness.

The lower bound for the single function case relies on the notion of *regular functions*. This is a natural generalization of vertex-disjoint function-triples, in the sense that the number of triangle passing through each point is not required to be 1 but can be some uniform constant. We also employ a notion of tensor product between functions, which preserves their "triangle-degree regularity". In analogy to the blow-up operation on graphs [1], we tensor with *bent functions* (see Section 2 for definition) to construct functions on arbitrarily long bits that actually depend on all these bits. [4]

Our result on canonical tester vs. general one-sided tester for triangle-freeness is an adaptation of the proof technique from [17] to the algebraic setting. The proof relies crucially on the fact that both of the testers are one-sided and the property of being triangle-free is invariant under non-singular linear transformations of the underlying domain $\mathbb{F}_2^n$. The latter is used to show that, under a random non-singular linear transformation, all linearly independent 2-tuples are basically equivalent. Therefore, in order to have guaranteed performance for *every* isomorphic copy of the input function, the best strategy for any tester (even an adaptive one) for triangle-freeness is to pick some *random* points in the domain to query and check for triangles.

1.3 Multi-function vs. Single-function Green in [19] used the term "triangle-freeness" to refer to the case $f_1 = f_2 = f_3$, what we call the "single-function case". Arguably, it is a more natural property to examine than the multi-function case. However, it is easily seen (and this has been explicitly observed previously, for example, in [25]) that Green's analysis extends to the multi-function version. Moreover, any analysis that goes through the route of a regularity lemma in the usual way should extend to the multi-

[3] Goldreich and Trevisan in [17] prove a polynomial relationship between the query complexity of *two-sided* testers and canonical testers, for any graph property. For the purposes of this paper, our weaker result is sufficient. However it is an interesting question to study 2-sided testing algorithms for matroid-freeness, see also the discussion in Section 6.

[4] As pointed out by an anonymous referee, taking tensor products between Boolean functions in our setting is also similar to taking tensor products in graphs.

function case. Thus, to determine whether a different notion of regularity is needed for which the number of partitions can be polynomial in $1/\epsilon$, it is reasonable to examine the (easier) multi-function case first, as we are doing in this paper. Additionally, testability of properties of a collection of functions is an interesting, but largely unexplored, question by itself.

1.4 Organization After some necessary definitions in Section 2, the the query complexity lower bounds for testing triangle-freeness in single functions and in function-triples are presented in Section 3 and Section 4, respectively. In Section 5, we study the relationship between the query complexities of the canonical tester and of a general one-sided tester for a broad class of algebraic properties.

2 Preliminaries

All logarithms in this paper are base 2. Let $\mathbb{N} = \{0, 1, \ldots\}$ denote the set of natural numbers. Let $n \geq 1$ be a natural number. We use $[n]$ to denote the set $\{1, \ldots, n\}$. The $n \times n$ identity matrix is denoted by $\mathbf{I}_n$. We view elements of $\mathbb{F}_2^n$ as n-bit strings, that is elements of $\{0,1\}^n$, alternatively. If x and y are two n-bit strings, then $x + y$ denotes bitwise addition (i.e. XOR) of x and y. We use (x, y) to denote the concatenation of two bit strings x and y.

Definition 2.1. (Tensor Product of Boolean Functions) *Let $f_1 : \mathbb{F}_2^{n_1} \to \{0,1\}$ and $f_2 : \mathbb{F}_2^{n_2} \to \{0,1\}$. Then the tensor product of f_1 and f_2, denoted by $f_1 \otimes f_2$, is a Boolean function on $\mathbb{F}_2^{n_1+n_2}$ such that $f_1 \otimes f_2(x_1, x_2) = f_1(x_1) \cdot f_2(x_2)$ for all $x_1 \in \mathbb{F}_2^{n_1}$ and $x_2 \in \mathbb{F}_2^{n_2}$.*

Note that if f_1 depends on all the n_1 variables and f_2 depends on all the n_2 variables, then $f_1 \otimes f_2$ depends on all the $n_1 + n_2$ input bits.

In order to define and study some properties of bent functions, first we recall the notion of Fourier transform.

Definition 2.2. (Fourier Transform) *Let $f : \mathbb{F}_2^n \to \mathbb{R}$. The* Fourier transform $\widehat{f} : \mathbb{F}_2^n \to \mathbb{R}$ *of f is defined to be $\widehat{f}(\alpha) = \mathbb{E}_x[f(x)\chi_\alpha(x)]$, where $\chi_\alpha(x) = (-1)^{\sum_{i\in[n]} \alpha_i x_i}$. $\widehat{f}(\alpha)$ is called the* Fourier coefficient *of f at α, and the $\{\chi_\alpha\}_\alpha$ are called characters.*

For $\alpha, \beta \in \mathbb{F}_2^n$, the inner product between α and β: $\langle \chi_\alpha, \chi_\beta \rangle \stackrel{\text{def}}{=} \mathbb{E}_{x\in\mathbb{F}_2^n}[\chi_\alpha(x)\chi_\beta(x)]$ is 1 if $\alpha = \beta$ and 0 otherwise. Therefore the characters form an orthonormal basis for $\mathbb{F}_2^n$, and we thus have the Fourier inversion formula $f(x) = \sum_{\alpha\in\mathbb{F}_2^n} \widehat{f}(\alpha)\chi_\alpha(x)$ and Parseval's equality $\sum_{\alpha\in\mathbb{F}_2^n} \widehat{f}(\alpha)^2 = \mathbb{E}_x[f(x)^2]$. For two functions $f, g : \mathbb{F}_2^n \to \mathbb{R}$, we define their *convolution* as $(f * g)(x) \stackrel{\text{def}}{=} \frac{1}{2^n} \sum_{y\in\mathbb{F}_2^n} f(y)g(x-y)$. By the convolution theorem, $\widehat{f \cdot g} = \widehat{f} * \widehat{g}$ and $\widehat{f * g} = \widehat{f} \cdot \widehat{g}$.

Definition 2.3. (Bent Functions) *Let $\phi : \mathbb{F}_2^n \to \{0,1\}$ be a Boolean function and let $\psi(x) = (-1)^{\phi(x)}$. ϕ is called a* bent function *if the Fourier coefficients of ψ satisfy that $|\widehat{\psi}(\alpha)| = \frac{1}{2^{n/2}}$ for every $\alpha \in \mathbb{F}_2^n$.*

Bent functions have many applications in cryptographic constructions. For more properties of bent functions, we refer interested readers to [26]. It is well known that bent functions exist when the number of variables is an even integer. For example, the inner-product function $\phi(x) = x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n$ is a bent function for every even n.

Let $f_1, f_2, f_3 : \mathbb{F}_2^n \to \{0,1\}$ be a function-triple. We say (f_1, f_2, f_3) is *triangle-free* if there is no x and y such that $f_1(x) = f_2(y) = f_3(x + y)$. We use T-free to denote the set of triangle-free function-triples.

Let $f, g : \mathbb{F}_2^n \to \{0,1\}$. The (relative) distance between f and g is defined to be the fraction of points at which they disagree: $\text{dist}(f, g) \stackrel{\text{def}}{=} \Pr_{x\in\mathbb{F}_2^n}[f(x) \neq g(x)]$. The distance between (f_1, f_2, f_3) and T-free is the minimum fraction of function values one needs to modify (f_1, f_2, f_3) to make it triangle-free, i.e.,

$$\text{dist}((f_1, f_2, f_3), \text{T-free}) \stackrel{\text{def}}{=} \min_{(g_1,g_2,g_3)\in\text{T-free}} (\text{dist}(f_1, g_1) + \text{dist}(f_2, g_2) + \text{dist}(f_3, g_3)) .$$

Let f_1, f_2, f_3 be a Boolean function-triple. The number of triangles passing through f_1 at x is $D_{f_1}(x) \stackrel{\text{def}}{=} |\{y \in \mathbb{F}_2^n : f_1(x) = f_2(y) = f_3(x+y) = 1\}|$. We define the *triangle degree* of f_1 at x, denoted by $d_{f_1}(x)$, to be $d_{f_1}(x) \stackrel{\text{def}}{=} D_{f_1}(x)/2^n$. Note that if $f_1(x) = 0$ then $d_{f_1}(x) = 0$, however the converse may not be true. Triangle degrees of f_2 and f_3 are defined identically. The triangle degree of a single Boolean function f at point x is defined in a similar way: $d_f(x) \stackrel{\text{def}}{=} \frac{1}{2^n} |\{y \in \mathbb{F}_2^n : f(x) = f(y) = f(x + y) = 1\}|$. When the function f is clear from context, we drop the subscript f and simply write the triangle degree as $d(x)$.

3 Lower Bound for Triangle-freeness in Single Functions

Let $f : \mathbb{F}_2^n \to \{0,1\}$ be a Boolean function. We define the *density* of f to be $\rho_f \stackrel{\text{def}}{=} \Pr_x[f(x) = 1]$. We say f is (ρ, d)-*regular* if $\rho_f = \rho$ and $d_f(x) = d$ for all x with $f(x) = 1$.

The reason that we are interested in (ρ, d)-regular functions is because there is an easy lower bound on the distance between regular functions and T-FREE.

PROPOSITION 3.1. *Let f be a (ρ, d)-regular function on n variables. Then there are exactly $\frac{\rho d 2^{2n}}{6}$ triangles of f and f is $\rho/3$-far from being triangle-free.*

Next we observe that tensor product preserves the triangle-degree regularity of Boolean functions.

LEMMA 3.1. *Let $f_1 : \{0,1\}^{n_1} \to \{0,1\}$ and $f_2 : \{0,1\}^{n_2} \to \{0,1\}$ such that f_1 is (ρ_1, d_1)-regular and f_2 is (ρ_2, d_2)-regular. Then $f_1 \otimes f_2$ is $(\rho_1 \cdot \rho_2, d_1 \cdot d_2)$-regular.*

We will use two simple functions defined below as the basic building blocks of our constructions.

FACT 3.1. *The function G on two variables, defined by $G(00) = 0, G(01) = G(10) = G(11) = 1$, is $(3/4, 1/2)$-regular. Also, the function H on three variables, defined by $H(000) = H(111) = 0$ and $H(x) = 1$ otherwise, is $(3/4, 1/2)$-regular.*

The next lemma shows that, by tensoring G with itself appropriate number of times, we obtain a Boolean function which is far from triangle-free yet does not contain too many triangles.

LEMMA 3.2. *For all small enough ϵ, there is a Boolean function which is ϵ-far from being triangle-free and the query complexity of the canonical triangle-freeness tester is $\Omega((1/\epsilon)^{3.409\cdots})$.*

Proof. Let $\ell = \lfloor \frac{\log(\frac{1}{3\epsilon})}{\log(4/3)} \rfloor$ and let $f_\epsilon = \underbrace{G \otimes \cdots \otimes G}_{\ell \text{ times}}$. Then f_ϵ is a Boolean function on $n = 2\ell$ variables. By Lemma 3.1, f_ϵ is a (ρ, d)-regular function with $\rho = (\frac{3}{4})^\ell \geq 3\epsilon$ (and also $\rho = O(\epsilon)$) and $d = (\frac{1}{2})^\ell = O(\epsilon^{\frac{\log 2}{\log(4/3)}}) = O(\epsilon^{2.409\cdots})$. By Proposition 3.1, f_ϵ is at least ϵ-far from being triangle-free and the number of triangles in f_ϵ is $\frac{\rho d 2^{2n}}{6} = O(\epsilon^{\frac{\log 4}{\log(4/3)}}) 2^{2n} = O(\epsilon^{3.409\cdots}) 2^{2n}$. □

In order to construct Boolean functions on arbitrarily large Boolean domains, we utilize bent functions to "stretch" the input bits. We show next that there are many bent functions which are regular and satisfy $\rho \approx 1/2$ and $d \approx 1/4$. Moreover, these regular bent functions on $\mathbb{F}_2^m$ exist for every even number $m \geq 2$.

LEMMA 3.3. *For every even number $m \geq 2$, if $\phi : \mathbb{F}_2^m \to \{0,1\}$ is a bent function with $\phi(0) = 0$, then ϕ is $\left(\frac{1}{2} \pm O(2^{-m/2}), \frac{1}{4} \pm O(2^{-m/2})\right)$-regular.*

Combining Lemma 3.1 and Lemma 3.3 gives the following single function triangle-freeness lower bound.

THEOREM 3.1. *For every small enough ϵ there is an integer $n_0(\epsilon)$ such that for all $n \geq n_0$, there is a Boolean function f on n variables with the following properties. f is ϵ-far from being triangle-free and the query complexity of the canonical triangle-freeness tester for f is $\Omega((1/\epsilon)^{3.409\cdots})$. Furthermore, f depends on all n input variables.*

Proof. Given $\epsilon > 0$, let $\ell = \lfloor \frac{\log(\frac{1}{3\epsilon})}{\log(4/3)} \rfloor$ and let f_ϵ be the function constructed in Lemma 3.2 on 2ℓ bits.

If n is even, then $m = n - 2\ell$ is even. Then we let ϕ_m be the bent function on m bits constructed in Lemma 3.3, and define a Boolean function on n variables by $f(x) = f_\epsilon \otimes \phi_m$. If n is odd, then $m = n - 2\ell - 3$ is even, we instead define $f(x) = f_\epsilon \otimes \phi_m \otimes H$. Now setting $n_0 = 2\ell + 3$ makes our construction works for all $n \geq n_0$. Since f_ϵ is regular and by Lemma 3.3 ϕ_m (or $\phi_m \otimes H$)is also regular, therefore following Lemma 3.1 f is a regular function. Moreover, as proved in Lemma 3.3, $\rho(\phi_m) = \Theta(1)$ and $d(\phi_m) = \Theta(1)$. Therefore the triangle degree d and density ρ of the regular function f still satisfy that $d = O(\rho^{2.409\cdots})$ as in Lemma 3.2, hence the same lower bound follows. □

4 Lower Bound for Triangle-freeness in Function-triples

4.1 Perfect-matching-free Families of Vectors

Our goal in this section is to construct function-triples such that all the triangles in a function-triple are disjoint. In other words, for each of the three functions, we want the triangle degree at each point to be either 0 or 2. We will build such function-triples using constructions of *perfect-matching free* families of vectors.

DEFINITION 4.1. (PERFECT-MATCHING-FREE FAMILIES OF VECTORS) *Let k and m be integers such that $0 < k < m < 2^k$. Let $\{a_i\}_{i=1}^m$ and $\{b_i\}_{i=1}^m$ be two families of vectors, with $a_i, b_i \in \{0,1\}^k$ for every $1 \leq i \leq m$. Let $c_i = a_i + b_i$.*

1. *Let $\{A_I\}_I$ be the set of (mk)-bit vectors formed by concatenating the m vectors in $\{a_i\}$ in all possible orders (there are $m!$ such vectors), where $I = (i_1, i_2, \ldots, i_m)$ is a permutation of $[m]$. Similarly define $\{B_J\}_J$ and $\{C_L\}_L$ as the concatenations of vectors in $\{b_i\}$ and $\{c_i\}$ with $J = (j_1, j_2, \ldots, j_m)$ and $L = (\ell_1, \ell_2, \ldots, \ell_m)$, respectively. We say the set of vectors $\{a_i, b_i, c_i\}$ is a (k, m)* 1-perfect-matching-free *(abbreviated as 1-PMF) family of vectors if $A_I + B_J = C_L$ necessarily implies that $I = J = L$ (i.e., $i_s = j_s = \ell_s$ for every $1 \leq s \leq m$).*

2. *Let $n' \geq 1$ be an integer and now let $\{A_I\}_I$, $\{B_J\}_J$ and $\{C_L\}_L$ be the sets of $n'mk$-bit vectors by concatenating n' copies of $\{a_i\}$, $\{b_i\}$ and $\{c_i\}$, respectively, in all possible orders (two concatenations are regarded the same if they give rise to two identical strings in $\{0,1\}^{n'mk}$). We say the set of vectors $\{a_i, b_i, c_i\}$ is a (k,m) n'-*PMF *family of vectors if $A_I + B_J = C_L$ necessarily implies that $I = J = L$.*

3. *Finally we say $\{a_i, b_i, c_i\}$ is a (k,m)-*PMF *family of vectors if it is n'-PMF for all $n' \geq 1$.*

In other words, suppose we color all the $3m$ vectors in $\{a_i, b_i, c_i\}$ with m different colors so that a_i, b_i and c_i are assigned the same color. Suppose further we are given equal number of copies of $\{a_1, b_1, c_1; \ldots; a_m, b_m, c_m\}$ and we wish to arrange them in three aligned rows such that all the a_i's are in the first row, all the b_i's are in the second row and all the c_i's are in the third row. Then the only way of making every column summing to 0^k is to take the trivial arrangement in which every column is monochromatic.

4.2 Construction Based on PMF Families of Vectors Let $\{a_i, b_i, c_i\}$ be a (k,m)-PMF family of vectors. Let n be an integer such that $mk|n$ and let $n' = \frac{n}{mk}$. let $\{A_I\}_I$, $\{B_J\}_J$ and $\{C_L\}_L$ be the sets of n-bit vectors by concatenating n' copies of $\{a_i\}$, $\{b_i\}$ and $\{c_i\}$ respectively. Note that $|\{A_I\}| = |\{B_J\}| = |\{C_L\}| = \frac{(n'm)!}{(n'!)^m}$. Now let $f_A, f_B, f_C : \mathbb{F}_2^n \to \{0,1\}$ be three Boolean functions which are the characteristic functions of sets $\{A_I\}_I$, $\{B_J\}_J$ and $\{C_L\}_L$ respectively. That is, $f_A(x) = 1$ iff $x \in \{A_I\}$, $f_B(x) = 1$ iff $x \in \{B_J\}$ and $f_C(x) = 1$ iff $x \in \{C_L\}$.

PROPOSITION 4.1. *All the triangles in the function-triple (f_A, f_B, f_C) are pairwise disjoint.*

THEOREM 4.1. *If (k,m)-PMF family of vectors exists, then there exists $\epsilon_0 = \epsilon_0(k,m)$ such that for all $\epsilon < \epsilon_0$, there is a $n_0 = n_0(\epsilon)$ and functions $f_A, f_B, f_C : \mathbb{F}_2^{n_0} \to \{0,1\}$ such that (f_A, f_B, f_C) is ϵ-far from being triangle-free and testing triangle-freeness in (f_A, f_B, f_C) requires the canonical tester to query the functions $\Omega((\frac{1}{\epsilon})^{\alpha - o(1)})$ times, where $\alpha = \frac{2 - \frac{\log m}{k}}{1 - \frac{\log m}{k}}$.*

Proof. Given a small enough $\epsilon > 0$, let n' be the largest integer such that $\epsilon \leq \frac{\frac{(n'm)!}{(n'!)^m}}{2^{n'mk}}$. Let f_A, f_B and f_C be the characteristic functions of $\{A_I\}_I$, $\{B_J\}_J$ and $\{C_L\}_L$ respectively defined above. Set $n_0 = n'mk$ and then f_A, f_B and f_C are Boolean functions on n_0 variables. Let N_Δ be the number of triangles in (f_A, f_B, f_C). Then by Stirling's formula, for all small enough ϵ (therefore large enough n' since we assume that m and k are fixed constants),

$$\begin{aligned} N_\Delta &= \frac{(n'm)!}{(n'!)^m} \\ &= \frac{\sqrt{2\pi mn'}(\frac{mn'}{e})^{mn'}(1 + O(\frac{1}{n'}))}{\left(\sqrt{2\pi n'}(\frac{n'}{e})^{n'}(1 + O(\frac{1}{n'}))\right)^m} \\ &= \Theta\left(\frac{m^{mn'}}{n'^{\frac{m-1}{2}}}\right) \\ &= 2^{(m \log m)n' - \frac{m-1}{2} \log n' - o(1)} \\ &= 2^{(\beta - o(1))n_0}, \end{aligned}$$

where $\beta = \frac{\log m}{k}$.

By Proposition 4.1, all the triangles in (f_A, f_B, f_C) are pairwise disjoint, therefore modifying the function-triple at one point in the domain can remove at most one triangle. Hence $\mathrm{dist}((f_A, f_B, f_C), \text{T-FREE}) \geq \frac{N_\Delta}{2^{n_0}} \geq \epsilon$. Consequently, the query complexity of the canonical tester is at least $\Omega(\frac{2^{2n_0}}{N_\Delta}) = \Omega(2^{(2-\beta+o(1))n_0}) = \Omega((\frac{1}{\epsilon})^{\alpha - o(1)})$. □

One can construct f_A, f_B, f_C to be Boolean functions on $\mathbb{F}_2^n$ for any $n \geq n_0$, by simply making the functions ignore the last $n - n_0$ bits and behave as defined above on the first n_0 bits. In Theorem 4.5, we give a construction by tensoring with bent functions so that the resulting functions depend on all n bits.

We conjecture the following to be true.

CONJECTURE 4.1. *There are infinitely many (k,m)-PMF families of vectors with $m \geq 2^{k(1-o_k(1))}$ as k (and hence m as well) tends to infinity.*

By Theorem 4.1, Conjecture 4.1 would imply a super-polynomial query lower bound for testing triangle-freeness in function-triples using the canonical tester. To be more specific, if there exists a (k,m)-PMF family of vectors with $m \geq 2^{k(1-o_k(1))}$, then query complexity is at least $\Omega((\frac{1}{\epsilon})^{\frac{1}{o_k(1)}})$. Moreover, when composed with Theorem 5.1 it would also give a super-polynomial lower bound for *any* one-sided triangle-freeness tester.

4.3 Existence of PMF Families of Vectors In this section we present an efficient algorithm which, given a family of vectors $\{a_i, b_i, c_i\}_{i=1}^m$, checks if it is PMF. Let $\{a_i, b_i, c_i\}_{i=1}^m$ be a family of vectors such that $a_i, b_i, c_i \in \mathbb{F}_2^k$ and $c_i = a_i + b_i$ for every $1 \leq i \leq m$. First we observe that if $\{a_i, b_i, c_i\}$ is PMF, then all the vectors in $\{a_i\}$ must be distinct. The same distinctness condition holds for vectors in $\{b_i\}$ and $\{c_i\}$. From now on, we assume these to be true. Next we define a set of "collision blocks".

Definition 4.2. (Collision Blocks) *Let $\{a_i, b_i, c_i\}_{i=1}^m$ be a family of vectors satisfying the distinctness condition. We say (i, j, ℓ) is a* collision block *if $a_i + b_j = c_\ell$, and for simplicity will just call it a block. We denote the set of all blocks by $\mathcal{B}$. We will call a block* trivial *if $i = j = \ell$ and* non-trivial *otherwise.*

Since $\{a_i, b_i, c_i\}$ satisfies the distinctness condition, clearly $|\mathcal{B}| < m^2$. Let r be the number of non-trivial blocks, and let $\{\mathsf{bl}_1, \ldots, \mathsf{bl}_r\}$ be the set of non-trivial blocks. For a collision block bl_s, we use $\mathsf{bl}_s^a, \mathsf{bl}_s^b$ and bl_s^c to denote the three indices of the colliding vectors. That is, if $\mathsf{bl}_s = (i, j, \ell)$ is a block, then $\mathsf{bl}_s^a = i$, $\mathsf{bl}_s^b = j$ and $\mathsf{bl}_s^c = \ell$.

Now suppose $\{a_i, b_i, c_i\}_{i=1}^m$ is not PMF. Then by the definition of PMF, there exists an integer n' such that $A_I, B_J, C_L \in \{0,1\}^{n'mk}$, $A_I + B_J = C_L$ and I, J, and L are not the same sequence of indices. We consider the equation $A_I + B_J = C_L$ as a tiling of $3 \times (n'm)$ k-bit vectors: the first row consists of the $n'm$ vectors from $\{a_i\}$ with each a_i appearing exactly n' times and the ordering is consistent with that of A_I. Similarly we arrange the second row with vectors from $\{b_i\}$ according to B_J and the third row with vectors from $\{c_i\}$ according to C_L. Observe that when we look at the columns of the tiling, each column corresponds to a block in $\mathcal{B}$. Now we remove all the trivial blocks, then because I, J, and L are not identical sequences of indices, there are some non-trivial blocks left in the tiling. Since all the blocks removed are trivial blocks, the remaining tiling still has equal number of a_i, b_i and c_i for every $1 \le i \le m$. We denote these numbers by $y_1, \ldots, y_m$. Note that y_i's are non-negative integers and not all of them are zero. Let the number of blocks bl_i left in the tiling be x_i, $1 \le i \le r$. Again x_i's are non-negative integers and not all zero. Moreover, we have the following constraints when counting the number of a_i, b_i and c_i vectors, respectively, left in the tiling:

(4.1)
$$\begin{cases} \sum_{j\in[r]:\mathsf{bl}_j^a=i} x_j - y_i = 0 \\ \sum_{j\in[r]:\mathsf{bl}_j^b=i} x_j - y_i = 0 \\ \sum_{j\in[r]:\mathsf{bl}_j^c=i} x_j - y_i = 0 \end{cases} \qquad \text{(for every } 1 \le i \le m\text{)}$$

where

$$x_j = \text{number of type } j \text{ blocks left after removing the trivial blocks}$$

and

$$y_i = \text{number of vectors } a_i \text{ (equiv. } b_i \text{ or } c_i\text{) left after removing the trivial blocks.}$$

Lemma 4.1. *$\{a_i, b_i, c_i\}_{i=1}^m$ is not PMF if and only there is a non-zero integral solution to the system of linear equations (4.1).*

Writing equations (4.1) in matrix form, we have

$$\mathbf{M}\vec{Z} = \vec{0},$$

where $\mathbf{M}$ is a $3m \times (r + m)$ integer-valued matrix (actually all entries are in the set $\{-1, 0, 1\}$) and

$$\vec{Z} = [x_1, \ldots, x_r, y_1, \ldots, y_m]^T$$

is an $(r + m) \times 1$ non-negative integer-valued column vector.

The following observation of Domenjoud [15], which essentially follows from Carathéodory's theorem, gives an exact characterization of when the system of equations (4.1) has a non-zero integral solution.

Theorem 4.2. ([15]) *Let $\mathbf{M}$ be an $s \times t$ integer matrix, then the Diophantine linear system of equations $\mathbf{M}\vec{Z} = \vec{0}$ with $\vec{Z} \in \mathbb{N}^t$ has a non-zero solution if and only if $\vec{0} \in Conv(M_1, \ldots, M_t)$, where M_i's are the column vectors of $\mathbf{M}$ and $Conv(M_1, \ldots, M_t)$ denotes the convex hull of vectors $M_1, \ldots, M_t$.*

It is well known that checking point-inclusion in a convex hull can be solved by Linear Programming, see e.g. [10]. Put everything together, there is an efficient procedure to check if a family of vectors $\{a_i, b_i, c_i\}_{i=1}^m$ is PMF or not. This enables us to find the following (k, m)-PMF families of vectors.

Theorem 4.3. *There are $(3, 4)$-PMF, $(4, 7)$-PMF and $(5, 13)$-PMF families of vectors.*

Proof. See Appendix A. □

We were unable to check the cases $k \ge 6$ since they are too large to do numerical calculations. However, our best findings for $k = 3, 4, 5$ indicates that the exponent α defined in Theorem 4.1 increases as k increases, which we view as a supporting evidence for Conjecture 4.1.

Now using the $(5, 13)$-PMF family of vectors as the building block, Theorem 4.1 implies the following.

Theorem 4.4. *For all small enough ϵ, there is an $n_0 = n_0(\epsilon)$ and functions $f_A, f_B, f_C : \mathbb{F}_2^{n_0} \to \{0, 1\}$ such that (f_A, f_B, f_C) is ϵ-far from being triangle-free and testing triangle-freeness of (f_A, f_B, f_C) requires the canonical tester to query the functions $\Omega((\frac{1}{\epsilon})^{4.847\cdots})$ times.*

Tensoring regular bent functions on appropriate number of bits with the function-triples constructed in Theorem 4.4 yields the following Theorem.

THEOREM 4.5. *For all small enough ϵ there is an integer $n_0(\epsilon)$ such that the following holds. For all integers $n \geq n_0$, there is a function-triple (f'_A, f'_B, f'_C) such that (f'_A, f'_B, f'_C) is ϵ-far from being triangle-free and testing triangle-freeness in (f'_A, f'_B, f'_C) requires the canonical tester to query the functions $\Omega((\frac{1}{\epsilon})^{4.847\cdots})$ times. Moreover, (f'_A, f'_B, f'_C) depends on all n input variables.*

5 Query Complexities of the Canonical Tester and General One-sided Testers

In this section, we prove a general result between the query complexities of an arbitrary one-sided tester and the canonical tester, for a large class of algebraic properties. A property in our class is specified [5] by k vectors $v_1, \ldots, v_k$ in the vector space $\mathbb{F}_2^r$. Following the notation in [12], we call this set of vectors a rank-r *matroid* $\mathcal{M}$ [6]. An alternative, equivalent notation based on solutions of systems of linear equations is adopted in [30].

DEFINITION 5.1. ($\mathcal{M}^*$-FREE) *Given a rank-r matroid $\mathcal{M} = (v_1, \ldots, v_k)$ with each $v_i \in \mathbb{F}_2^r$, a k-tuple of Boolean functions $f_1, \ldots, f_k : \mathbb{F}_2^n \to \{0,1\}$ is said to be* $\mathcal{M}^*$-free *if there is no full-rank linear transformation $L : \mathbb{F}_2^r \to \mathbb{F}_2^n$ such that $f_i(L(v_i)) = 1$ for every $i \in [k]$. Otherwise, if such an L exists, $f_1, \ldots, f_k$ is said to* contain $\mathcal{M}$ at L, *or equivalently, L is called a* violating linear transformation *of $\mathcal{M}$.*

REMARK 5.1. *Let $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_r)$ be a set of basis vectors in $\mathbb{F}_2^r$. Each linear map L in the above definition is then specified by r vectors $z_1, \ldots, z_r$ in $\mathbb{F}_2^n$ such that $L(\boldsymbol{e}_i) = z_i$ for every $1 \leq i \leq r$. The linear map L is* full rank *if $(z_1, \ldots, z_r)$ are linearly independent.*

To see that this generalizes the triangle-freeness property, let $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ be the two unit vectors in $\mathbb{F}_2^2$ and consider the matroid $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_1 + \boldsymbol{e}_2)$. Then the three elements of the matroid will be mapped to all triples of the form $(x, y, x+y)$ by the set of full-rank linear transformations, where x and y are two distinct non-zero elements in $\mathbb{F}_2^n$. Also note that in this case, $r = 2$ and $k = 3$.

The property of being $\mathcal{M}^*$-free is not linear-invariant. The original notion of $\mathcal{M}$-freeness, as defined in [12], allows L in the above definition be arbitrary linear transformations, not restricted to full-rank ones, and is hence truly linear-invariant. However, from a conceptual level, for a fixed matroid $\mathcal{M}$, the property of being $\mathcal{M}$-free and being $\mathcal{M}^*$-free are very similar. It is analogous to the distinction between a graph being free of H as a subgraph and being free of homomorphic images of H, for a fixed graph H.

In terms of testability, we have some evidence that the distinction is unimportant, although we are unable to prove a formal statement at this time. For the case when $\mathcal{M} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_1 + \boldsymbol{e}_2)$, we can show that a tester for triangle-freeness can be converted to one for triangle*-freeness. Consider a function-triple (f_1, f_2, f_3) that is promised to be either triangle*-free or ϵ-far from being triangle*-free, where the distance parameter ϵ is a constant. Define a new function-triple (f'_1, f'_2, f'_3) by setting, for $i = 1, 2, 3$, $f'_i(0) = 0$ and $f'_i(x) = f_i(x)$ for all $x \neq 0$. Observe that if (f_1, f_2, f_3) is triangle*-free, then (f'_1, f'_2, f'_3) is triangle-free because setting $f'_i(0) = 0$ removes all degenerate triangles. On the other hand, if (f_1, f_2, f_3) is ϵ-far from triangle*-free, then (f'_1, f'_2, f'_3) is still $\epsilon' \geq \epsilon - 3/2^n$ far from triangle*-free and, hence, also from triangle-free. Since ϵ' approaches ϵ as n goes to infinity, assuming the continuity of the query complexity as a function of the distance parameter, the query complexity of triangle-freeness is therefore lower-bounded [7] by the query-complexity of triangle*-freeness.

For general binary matroids $\mathcal{M} = (v_1, \ldots, v_k)$ with each $v_i \in \mathbb{F}_2^r$, observe that if a function tuple is far from being $\mathcal{M}$-free, then almost all the linear maps where $\mathcal{M}$ is contained are full-rank. This is because the main theorems of [30] and [24] show that if a function tuple is $\Omega(1)$-far from $\mathcal{M}$-free, then $\mathcal{M}$ is contained at $\Omega(2^{nr})$ many linear maps, while there are only $o(2^{nr})$ many linear maps $L : \mathbb{F}_2^r \to \mathbb{F}_2^n$ of rank less than r. Therefore, in fact, any $\mathcal{M}^*$-free function tuple is $o(1)$-close to $\mathcal{M}$-free. If there were a more query efficient one-sided tester for $\mathcal{M}$-freeness than for $\mathcal{M}^*$-freeness, it must be the case that the few linear maps with rank less than r where $\mathcal{M}$ is contained can somehow be discovered more efficiently than the full-rank maps. But on the other hand, we know of a large class of matroids $\mathcal{M}$ for which there exist functions that are far from $\mathcal{M}$-free but do not contain $\mathcal{M}$ at *any* non-full-rank linear map. More precisely, letting $C_k = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{k-1}, \boldsymbol{e}_1 + \cdots + \boldsymbol{e}_{k-1})$ be the graphic matroid of the k-cycle, Theorem 1.3 in [12] proves that for any odd $k \geq 5$, there exist functions which are far from C_k-free but contain C_k only at full-rank linear maps (by showing a separation between the classes C_k-free and C_{k-2}-free). So, for these reasons, it seems unlikely that the query complexities of testing

[5] We assume that r is the minimal dimension of the vector space which preserves the linear dependencies between $v_1, \ldots, v_k$. That is, r is the rank of the matrix with $v_1, \ldots, v_k$ as its columns.

[6] In this paper we do not employ any property of matroids. Here matroid is simply a synonym for a collection of binary vectors.

[7] The other direction is easy to show in general: for any binary matroid $\mathcal{M}$ and constant ϵ, an ϵ-tester for $\mathcal{M}^*$-freeness can be used to ϵ-test $\mathcal{M}$-freeness (again assuming continuity of the query complexity function).

$\mathcal{M}^*$-freeness properties are very different from those of testing $\mathcal{M}$-freeness properties. We conjecture that the query complexities of testing $\mathcal{M}$-freeness and $\mathcal{M}^*$-freeness properties are the same [8] and leave this as an open problem.

We first observe a simple fact about the behavior of any *one-sided* tester for $\mathcal{M}^*$-freeness.

LEMMA 5.1. *Let $\mathcal{M}$ be a matroid of k vectors. Then any one-sided tester T for $\mathcal{M}^*$-freeness rejects if and only if it detects a violating full-rank linear transformation L of $\mathcal{M}$.*

Next, we define the canonical tester for $\mathcal{M}^*$-freeness, which naturally extends the previously described canonical tester for triangle-freeness.

DEFINITION 5.2. (CANONICAL TESTER) *Let $\mathcal{M} = (v_1, \ldots, v_k)$, with each $v_i \in \mathbb{F}_2^r$, be a rank-r matroid of k vectors. A tester $\mathcal{T}$ for $\mathcal{M}^*$-freeness is* canonical *if $\mathcal{T}$ operates as follows. Given as input a distance parameter ϵ and oracle access to k-tuple of Boolean functions $f_1, \ldots, f_k : \mathbb{F}_2^n \to \{0,1\}$, the tester $\mathcal{T}$ repeats the following process independently $\ell(\epsilon)$ times: select uniformly at random a rank-r linear transformation $L : \mathbb{F}_2^r \to \mathbb{F}_2^n$ and check if f contains $\mathcal{M}$ at L. If so, $\mathcal{T}$ rejects and halts. If $\mathcal{T}$ does not reject after $\ell(\epsilon)$ iterations, then $\mathcal{T}$ accepts. The query complexity of the canonical tester is therefore at most $\ell(\epsilon) \cdot k$.*

Our main theorem in this section is the following.

THEOREM 5.1. *For a given rank-r matroid $\mathcal{M} = (v_1, \ldots, v_k)$ with each $v_i \in \mathbb{F}_2^r$, suppose there is a one-sided tester for $\mathcal{M}^*$-freeness with query complexity $q(\mathcal{M}, \epsilon)$. Then the canonical tester for $\mathcal{M}^*$-freeness has query complexity at most $O(k \cdot q(\mathcal{M}, \epsilon)^r)$.*

Proof. Since the rank of $\mathcal{M}$ is r, without loss of generality, we assume that $v_1, \ldots, v_r$ are the r basis vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_r$. Thus, any linear transformation $L : \mathbb{F}_2^r \to \mathbb{F}_2^n$ is uniquely determined by $L(v_1), \ldots, L(v_r)$.

Suppose we have a one-sided, possibly adaptive, tester T for $\mathcal{M}$-freeness with query complexity $q(\mathcal{M}, \epsilon)$. We say T operates in *steps*, where at each step $i \in [q(\mathcal{M}, \epsilon)]$, T selects an element y_i from $\mathbb{F}_2^n$ (based on a distribution that depends arbitrarily on internal coin tosses and oracle answers in previous steps) and then queries the oracle for the value of $f_j(y_i)$, for some $1 \le j \le k$.

We convert the tester T into another tester T' that operates as follows. Given oracle access to a function tuple $f_1, \ldots, f_k : \mathbb{F}_2^n \to \{0,1\}$, T' first selects, uniformly at random, a *non-singular* linear map $\Pi : \mathbb{F}_2^n \to \mathbb{F}_2^n$, and then invokes the tester T, providing it with $f_j(\Pi(y))$ whenever it queries for $f_j(y)$. For convenience the linear map may be generated on-the-fly in the following sense. Suppose in the first $i-1$ queries, T queries $(y_1, \ldots, y_{i-1})$ and T' queries $(x_1, \ldots, x_{i-1})$. Now if T chooses a new point y_i to query, tester T' picks a Π uniformly at random from all non-singular maps that are consistent with all the points queried previously, that is, maps satisfying $\Pi(y_1) = x_1, \ldots, \Pi(y_{i-1}) = x_{i-1}$, and feeds the query result at $\Pi(y_i)$ to the original tester T.

CLAIM 5.1. *T' is also a tester of $(f_1, \ldots, f_k)$ for $\mathcal{M}^*$-freeness with the same query complexity as T.*

For convenience, let us fix the following notation. At a step $i \in [q(\mathcal{M}, \epsilon)]$, the element whose value is requested by T is denoted y_i, and the element of $\mathbb{F}_2^n$ queried by T' (and whose value is supplied to T) is denoted x_i. Both x_i and y_i are of course random variables, and also $x_i = \Pi(y_i)$. We now make the simple observation that at each step, no matter how cleverly T selects the y_i's, each x_i is either uniformly distributed outside or lies inside the span of elements selected at previous steps. More precisely:

LEMMA 5.2. *Fix an integer $i \in [q(\mathcal{M}, \epsilon)]$. Let $y_1, \ldots, y_i$ be the elements in $\mathbb{F}_2^n$ requested by T in the first i stages, and elements $x_1, \ldots, x_{i-1}$ be the points queried by T' in the first $i-1$ steps. Then, x_i, the element queried by T' at the i^{th} step is either an element in $\mathrm{span}(x_1, \ldots, x_{i-1})$ or is uniformly distributed in $\mathbb{F}_2^n - \mathrm{span}(x_1, \ldots, x_{i-1})$.*

Due to Lemma 5.2, we may divide the queries of T into two types: *staying query* if the newly queried point is in the span of the previously queried points, and *expanding query* if the newly queried point is a random point outside the span of previously queried points. Let the number of expanding queries of T' be t, $t \le q(\mathcal{M}, \epsilon)$ and let the subspace spanned by $(x_1, \ldots, x_{q(\mathcal{M},\epsilon)})$ be $V_{T'}$, then clearly $\dim(V_{T'}) = t$ and the expanding query points generate $V_{T'}$ (i.e., the set of expanding queries $(x_{i_1}, \ldots, x_{i_t})$ form a basis for $V_{T'}$). Therefore, as a corollary to Lemma 5.2, we have the following property of $V_{T'}$.

COROLLARY 5.1. *The subspace $V_{T'}$ spanned by the query points of tester T' is a random subspace of dimension t in $\mathbb{F}_2^n$.*

[8] It seems possible that some functions may have quite different query complexities for these two properties. However, the query complexities in our conjecture are measured as (non-increasing) functions of the distance parameter ϵ, which are *worst-case* query complexities among all input functions that are ϵ-far from the corresponding properties.

Since each non-singular linear transformation is determined by the images of the first r vectors in the matroid, it suffices to study the distribution of these r-tuples. The next Lemma shows, when we look at any fixed linearly independent r-tuple inspected by T, the corresponding r-tuple queried by T' after a random non-singular transformation of the space $\mathbb{F}_2^n$, distributes uniformly over all linearly independent r-tuples.

LEMMA 5.3. *Let $V_{T'}$ be a random subspace in $\mathbb{F}_2^n$ of dimension $t < n$ generated by picking uniformly at random a set of t linearly independent vectors $(b_1, \ldots, b_t)$* [9] *in $\mathbb{F}_2^n$ as basis. Let $x = (x_1, \ldots, x_r)$ be any* fixed *linearly independent r-tuple, $r \leq t$, given by a set of linear combinations of the basis vectors $(b_1, \ldots, b_t)$. Then x is uniformly distributed over all linearly independent r-tuples in $(\mathbb{F}_2^n)^r$.*

By Lemma 5.1, T' rejects if and only if it detects a violating full-rank linear transformation. Notice that each full-rank linear transformation $L : \mathbb{F}_2^r \to \mathbb{F}_2^n$ corresponds to a linearly independent r-tuple $(z_1, \ldots, z_r) \in (\mathbb{F}_2^n)^r$, where the corresponding linear transformation is given by $L_{z_1,\ldots,z_r}(u_1, \ldots, u_r) = \sum_{i=1}^r u_i z_i$. Thus, T' rejects iff it finds a linearly independent r-tuple $(z_1, \ldots, z_r)$ such that the corresponding linear transformation is violating. Furthermore, because $v_1 = \boldsymbol{e}_1, \ldots, v_r = \boldsymbol{e}_r$, the elements $z_1, \ldots, z_r$ must lie in the set of samples made by T'. Then, since T' makes $q(\mathcal{M}, \epsilon)$ queries, the total number of linearly independent r-tuples T' can check is at most $q(\mathcal{M}, \epsilon) \cdot (q(\mathcal{M}, \epsilon) - 1) \cdots (q(\mathcal{M}, \epsilon) - r + 1) < q(\mathcal{M}, \epsilon)^r$. Let δ be the fraction of violating linearly independent r-tuples $z = (z_1, \ldots, z_r) \in (\mathbb{F}_2^n)^r$. By Lemma 5.3, each linearly independent r-tuple checked by T' is drawn uniformly at random from the set of all linearly independent r-tuples in $(\mathbb{F}_2^n)^r$. That is, the probability that T' rejects after checking any non-singular linear transformation it inspects is exactly δ. By union bound, the probability that T' rejects $(f_1, \ldots, f_k)$ after $q(\mathcal{M}, \epsilon)$ queries is at most $\delta q(\mathcal{M}, \epsilon)^r$. In order to reject with probability at least $2/3$, the query complexity of T' is at least $q(\mathcal{M}, \epsilon) \geq (\frac{2}{3\delta})^{1/r}$. Now consider the canonical tester T'' that runs in ℓ independent stages which, at each stage, selects uniformly at random a linearly independent r-tuple $(z_1, \ldots, z_r)$ and checks for violation of $\mathcal{M}^*$-freeness. How many queries does T'' need to make to achieve the same rejection probability on $(f_1, \ldots, f_k)$ as T' does after $q(\mathcal{M}, \epsilon)$ queries? Clearly the probability that T'' rejects $(f_1, \ldots, f_k)$ after ℓ stages is $1 - (1 - \delta)^\ell \geq 2/3$, for all $\ell \geq \ell_0 = \frac{2}{\delta} = O(q(\mathcal{M}, \epsilon)^r)$. Since T'' makes k queries in each stage, the total number of queries T'' makes is at most $k\ell_0 = O(k \cdot q(\mathcal{M}, \epsilon)^r)$. □

[9] One may think of the basis of $V_{T'}$ as the set of expanding query points $(x_{i_1}, \ldots, x_{i_t})$ of tester T'.

6 Concluding Remarks and Open Problems

We have given polynomial lower bounds on the query complexities of canonical triangle-free tester for both the triple function case and single function case. We strongly believe that there exist super-polynomial lower bounds for both of these problems. One possible approach is try to prove Conjecture 4.1 for the triple function case. It seems that one of the main difficulties in understanding triangle-freeness lower bound is that there is no good characterization of the distance between a Boolean function and the set of triangle-free functions (as opposed to the linearity case, where the distance is exactly characterized by the Fourier coefficients of the function). It is also interesting to study the query complexities of (cycle) C_r-freeness for $r \geq 4$.

Another interesting problem is whether the tower of 2's type query upper bound of testing triangle-freeness can be improved. Is it possible that some two-sided testers can achieve much better upper bound? Finally, is there a separation between multi-function and single-function versions of triangle-freeness or other matroid-freeness properties?

Acknowledgments

We thank Victor Chen and Madhu Sudan for collaboration during the early stages of this research as well as enlightening discussions. We are indebted to Ilan Newman for asking a question that initiated the work presented in Section 5. We thank Avinatan Hassidim, Ronitt Rubinfeld and Andy Yao for helpful discussions and Alex Samorodnitsky and an anonymous referee for valuable comments.

References

[1] Noga Alon. Testing subgraphs in large graphs. *Random Structures and Algorithms*, 21(3-4):359–370, 2002.

[2] Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(6):451–476, 2000.

[3] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 251–260, 2006.

[4] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing low-degree polynomials over GF(2). In *Proceedings of Random 2003*, pages 188–199, 2003.

[5] Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30(6):1842–1862, 2000.

[6] Noga Alon and Asaf Shapira. Testing subgraphs in directed graphs. *Journal of Computer and System Sciences*, 69(3):354–382, 2004.

[7] Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. In *FOCS'05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 429–438. IEEE Computer Society, 2005.

[8] Noga Alon and Asaf Shapira. Every monotone graph property is testable. In Harold N. Gabow and Ronald Fagin, editors, *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 128–137. ACM, 2005.

[9] Tim Austin and Terence Tao. On the testability and repair of hereditary hypergraph properties. `http://arxiv.org/abs/0801.2179`, 2008.

[10] Thomas Bailey and John Cowles. A convex hull inclusion test. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):312–316, 1987.

[11] Eli Ben-Sasson, Prahladh Harsha, and Sofya Raskhodnikova. Some 3CNF properties are hard to test. *SIAM Journal on Computing*, 35(1):1–21, 2005. Early version in STOC'03.

[12] Arnab Bhattacharyya, Victor Chen, Madhu Sudan, and Ning Xie. Testing linear-invariant non-linear properties. In *STACS'09*, pages 135–146, 2009.

[13] Christian Borgs, Jennifer T. Chayes, László Lovász, Vera T. Sós, Balázs Szegedy, and Katalin Vesztergombi. Graph limits and parameter testing. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 261–270, 2006.

[14] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 2000.

[15] Eric Domenjoud. Solving systems of linear diophantine equations: an algebraic approach. In *In Proc. 16th Mathematical Foundations of Computer Science, Warsaw, LNCS 520*, pages 141–150. Springer-Verlag, 1991.

[16] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[17] Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, 23(1):23–57, 2003.

[18] Timothy Gowers. Lower bounds of tower type for Szemerédi's uniformity lemma. *Geom. Funct. Anal.*, 7(2):322–337, 1997.

[19] Ben Green. A Szemerédi-type regularity lemma in abelian groups, with applications. *Geom. Funct. Anal.*, 15(2):340–376, 2005.

[20] Peter Gruber. *Convex and Discrete Geometry*. Springer, New York, 2007.

[21] Charanjit S. Jutla, Anindya C. Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. In *FOCS'04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 423–432, 2004.

[22] Tali Kaufman and Dana Ron. Testing polynomials over general fields. In *FOCS'04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 413–422, 2004.

[23] Tali Kaufman and Madhu Sudan. Algebraic property testing: The role of invariance. In *STOC'08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 403–412, 2008.

[24] Daniel Král', Oriol Serra, and Lluís Vena. A removal lemma for systems of linear equations over finite fields, 2008.

[25] Daniel Král', Oriol Serra, and Lluís Vena. A combinatorial proof of the removal lemma for groups. *Journal of Combinatorial Theory*, 116(4):971–978, May 2009.

[26] Florence J. MacWilliams and Neil J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[27] Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic Boolean formulae. *SIAM Journal on Discrete Mathematics*, 16(1):20–46, 2003.

[28] Vojtěch Rödl and Mathias Schacht. Generalizations of the removal lemma. *Combinatorica*, To appear. Earlier version in STOC'07.

[29] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[30] Asaf Shapira. Green's conjecture and testing linear-invariant properties. In *STOC'09: Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 159–166, 2009.

A Proof of Theorem 4.3

Proof. [Proof of Theorem 4.3] By numerical calculation, the following set of vectors is $(3, 4)$-PMF:

$$\begin{aligned} a_1 &= 110 & b_1 &= 001 \\ a_2 &= 010 & b_2 &= 100 \\ a_3 &= 101 & b_3 &= 111 \\ a_4 &= 011 & b_4 &= 011. \end{aligned}$$

The following set of vectors is $(4, 7)$-PMF:

$$\begin{aligned} a_1 &= 1101 & b_1 &= 0011 \\ a_2 &= 0001 & b_2 &= 1011 \\ a_3 &= 0010 & b_3 &= 0111 \\ a_4 &= 0110 & b_4 &= 1001 \\ a_5 &= 0000 & b_5 &= 0000 \\ a_6 &= 0111 & b_6 &= 0100 \\ a_7 &= 1001 & b_7 &= 0101. \end{aligned}$$

The following set of vectors is $(5, 13)$-PMF:

$$
\begin{array}{ll}
a_1 = 11101 & b_1 = 01101 \\
a_2 = 11001 & b_2 = 11101 \\
a_3 = 11000 & b_3 = 10011 \\
a_4 = 00101 & b_4 = 10001 \\
a_5 = 10010 & b_5 = 00101 \\
a_6 = 11110 & b_6 = 10100 \\
a_7 = 10000 & b_7 = 10000 \\
a_8 = 01000 & b_8 = 01111 \\
a_9 = 00011 & b_9 = 01010 \\
a_{10} = 11100 & b_{10} = 00111 \\
a_{11} = 00010 & b_{11} = 11010 \\
a_{12} = 01100 & b_{12} = 10010 \\
a_{13} = 01010 & b_{13} = 11111. \square
\end{array}
$$