An Upper Bound on Checking Test Complexity for Almost All Cographs

Oleg V. Zubkov
Department of Computer Science
East-Siberian State Academy of Education
Irkutsk, Russia
Email: oleg.zubkov@mail.ru

Dmitry V. Chistikov and Andrey A. Voronenko
Faculty of Computational Mathematics and Cybernetics
Lomonosov Moscow State University
Moscow, Russia
Email: dch@cs.msu.ru, dm6@cs.msu.ru

***Abstract*—The concept of a checking test is of prime interest to the study of a variant of exact identification problem, in which the learner is given a hint about the unknown object. A graph F is said to be a checking test for a cograph G iff for any other cograph H there exists an edge in F distinguishing G and H, that is, contained in exactly one of the graphs G and H. It is known that for any cograph G there exists a unique irredundant checking test, the number of edges in which is called the checking test complexity of G. We show that almost all cographs on n vertices have relatively small checking test complexity of $O(n \log n)$. Using this result, we obtain an upper bound on the checking test complexity of almost all read-once Boolean functions over the basis of disjunction and parity functions.**

***Keywords*-cograph, checking test, complexity, equivalence query, teaching, read-once Boolean function.**

I. Introduction

A general setup of a checking test problem can be described as follows. Consider a set $\mathcal{C}$ and imagine a black box containing a single object $c \in \mathcal{C}$. One knows for sure that the object in the box comes from $\mathcal{C}$, and can ask questions about it. One is also given a hint that the object in the box is c_0. The checking test problem for c_0 lies in verifying the hint, that is, in checking whether the box indeed contains c_0 or not.

In this paper, we study checking test complexity for a certain class of graphs, namely, for cographs [11]. A graph is called *complement-reducible*, or a *cograph*, if it can be reduced to distinct vertices by recursively complementing its connected components (or, equivalently, if it does not contain an induced subgraph isomorphic to a simple path P_4 on four vertices). Cographs are closely related to a broad class of rooted trees. They have arisen in many areas of mathematics and computer science and have been independently rediscovered several times.

Let G be a cograph on vertices X. A graph F on X is said to be a *checking test* for G iff for any cograph H on X such that $H \neq G$ there exists an edge e in F contained in exactly one of the graphs G and H. A *length* of the test is the number of edges contained in it. The smallest length of a checking test for G is called the *checking test complexity* of G.

A checking test for a cograph G can be regarded as a certificate for G in the following sense. If from all edges e contained in a checking test F for G, a cograph H on the same set of vertices contains exactly those that are present in G, then $H = G$. Checking test complexity is therefore a measure of how hard the task of proving that $H = G$ is, if H is known a priori to be a cograph.

One can easily see that a complete graph on X is a checking test for all cographs on X by definition, so the checking test complexity for any cograph on n vertices is less or equal to $\binom{n}{2}$, which is quadratic in n. It must be stressed that there exist cographs requiring all edges to be present in their checking test and thus having checking test complexity of exactly $\binom{n}{2}$ (a complete graph itself is one of these cographs). For some cographs, however, checking test complexity is as low as $2n - 3$, which is linear in n.

The checking test problem for cographs was studied in [25]. It was shown that all *irredundant* checking tests (i. e., ones containing no edges that can be eliminated without losing the property of being a checking test) for any fixed cograph have equal length. A simple formula giving the exact value of this length (checking test complexity) was deduced. It was also demonstrated that checking tests for cographs can be used to construct individual checking tests for Boolean read-once functions over the basis of all two-variable functions. Exact statements of these results are given below.

This paper is devoted to establishing an upper bound on checking test complexity for *almost all* cographs. We show that the fraction of cographs on n vertices allowing a checking test of length $O(n \log n)$ tends to 1 as $n \to \infty$. More precisely, we prove the following theorem:

Theorem 1: Checking test complexity of almost all cographs on n vertices is less or equal to

$$4n \log_2 n \cdot (1 + o(1)).$$

The statement of the theorem means that almost all cographs have relatively small checking test complexity, as compared to the maximum of $\binom{n}{2} = \Omega(n^2)$. This result, apart from having independent value, proves useful for estimating checking test complexity for other classes of objects. Exploiting cographs' relation to trees and applying

a result from [25], we establish an analogous bound on checking test complexity for almost all read-once Boolean functions over the basis of disjunction and sum modulo 2 (parity).

A Boolean function f is called *read-once* over a basis B iff it can be expressed by a formula over B where no variable appears more than once. The structure of such formulae can be represented by rooted trees, where leaves stand for variables, and internal nodes are labeled with functions from B. A checking test for a read-once function $f(x_1, \ldots, x_n)$ over B depending essentially on all its variables is defined as a set of input vectors distinguishing f from all other read-once functions over B of variables $x_1, \ldots, x_n$. Checking test complexity of a read-once function f over B is the smallest possible number of input vectors in a checking test for f.

For computational learning theory, the study of checking tests is interesting for two reasons. First, checking tests can be regarded as *teaching sequences* described by Goldman and Kearns [14]. For a Boolean function f from a certain class $\mathcal{C}$, a teaching sequence is a list of pairs of form $\langle \alpha, f(\alpha) \rangle$ such that f is the only function in $\mathcal{C}$ consistent with all given pairs. The prime difference between the results on checking tests presented here and previously known results for Goldman and Kearns' model is that we study the problem for individual objects, whereas Goldman and Kearns' work is focused on estimating the *teaching dimension*, which is the complexity of teaching the "hardest" object from the whole concept class. In test theory, maximum complexity of an object in the class with a bounded size is usually called *Shannon function*. For cographs on n vertices, this function is equal to $\binom{n}{2}$.

A second point of view on checking tests in the context of learning is related to the notion of *equivalence queries* for exact identification problem, the goal of which is to determine exactly what object is hidden in the black box by asking questions (performing queries) [2]. For Boolean concepts regarded as functions, an equivalence query allows the learner to check whether the (unknown) target function f can be expressed by a given formula (in some settings, more general representations are allowed). The answer is either "yes" or a counterexample α such that $f(\alpha) \neq g(\alpha)$, where g is a function expressed by the given formula. Consequently, a checking test can be regarded as an implementation of an equivalence query with more primitive *membership queries*, which simply ask the value of the function in the black box on a given input. In order to obtain the answer to the equivalence query, it is sufficient to query the value of the target function on all vectors in a checking test.

It is worth remarking, however, that minor details in the definitions of a checking test for a specific concept class and an equivalence query can create an exponential complexity gap between these two notions. A discussion of this issue can be found at the end of our paper.

II. Preliminaries and related results

First of all, we need some facts concerning cographs [11]. Suppose that T is a rooted tree with the set of leaves X and no nodes with exactly one child. Also suppose that *internal nodes* (non-leaf nodes) of T are properly coloured with 0 and 1 (no two adjacent nodes have the same colour). Any tree satisfying these conditions is called a *cotree*. Denote by $\phi(T)$ a graph on vertices X such that an edge $\{x_i, x_j\}$ is contained in $\phi(T)$ iff the lowest common ancestor of x_i and x_j in T is coloured with 1. Any graph $\phi(T)$ constructed in this manner is a cograph. The mapping ϕ is known to be a bijection between the set of all cotrees with leaves X and the set of all cographs on vertices X. A cotree T represents the process of reducing the cograph $\phi(T)$ to distinct vertices by repeatedly complementing its connected components.

Now, suppose that G is a cograph on $X = \{x_1, \ldots, x_n\}$ and $T = \phi^{-1}(G)$. According to [25], [8], the checking test complexity of G is equal to

$$\nu(T) = \sum_{\{u,w\} \in \operatorname{int} \mathrm{E}(T)} b(u)\, b(w) + \frac{1}{2} \sum_{v \in \operatorname{int} \mathrm{V}(T)} b^2(v) + \frac{n-1}{2}, \quad (1)$$

where $\operatorname{int} \mathrm{V}(T)$ is the set of all internal nodes of T, $\operatorname{int} \mathrm{E}(T)$ is the set of all edges incident only to internal nodes, $b(v) = \deg v - 1$ and $\deg v$ is the number of children of v in T. Furthermore, if F is a checking test for G of length greater than $\nu(T)$, then some of its edges can be removed without losing the property of being a checking test for G. Finally, for each cograph G there exists a unique checking test F containing exactly $\nu(T)$ edges [25].

It is also known that, for $n \geq 2$, the smallest possible value of $\nu(T)$ is $2n-3$, for binary cotrees and, if $n = 3$, for a three-leaf star graph (the latter corresponds to a complete cograph on three vertices). The largest possible value is $\binom{n}{2}$, for cotrees with exactly one or two internal nodes. It is also known that for every integer m between $2n-3$ and $\binom{n}{2}$, there exists a cotree T with n leaves such that $\nu(T) = m$. Hence, for cographs on $n \geq 2$ vertices the set of all possible values of checking test complexity is exactly $[2n-3; \binom{n}{2}] \cap \mathbb{N}$. These results are developed in [8].

The study of checking test complexity for cographs was motivated by a related research on read-once Boolean functions. A nontrivial checking test problem for read-once functions described above was suggested by Voronenko in [21]. This problem is referred to as *testing with respect to read-once alternatives*. For all finite bases this problem is believed to be polynomial in the number of variables, whereas one cannot unconditionally identify an unknown read-once conjunction of n literals with less than $2^n - 1$ queries of type $(\alpha_1, \ldots, \alpha_n) \mapsto f(\alpha_1, \ldots, \alpha_n)$ (membership queries in query learning).

For the basis of all two-variable functions, the following results are known. An exact universal upper bound (Shannon function) of $\binom{n}{2}+n+1$ on checking test complexity for read-

once functions over this basis was obtained by Ryabets [19]. Individual upper bounds can be as low as $3n-2$ (see, e. g., [25]). An n-ary disjunction requires at least $\binom{n}{2}+n+1$ vectors in its test.

Checking techniques for this basis enabling one to obtain individual bounds on checking test complexity for any read-once function f were suggested in [25]. These techniques exploit a connection between read-once functions over this basis and cographs. First, suppose that the target function is read-once over the basis $\{\vee, \oplus\}$ containing only disjunction and sum modulo 2 (parity) functions. Let T_f be a cotree representing the structure of a formula for f where no variable appears more than once, and symbols $\oplus$ and $\vee$ are replaced with 0 and 1, respectively. It turns out that checking test complexity of f is always less or equal to $\nu(T_f)+n+1$, even if arbitrary read-once functions over the basis of all two-variable functions are allowed as alternatives for f. If f itself is read-once only over this basis, the upper bound of $4\,\nu(\ddot{T}_f)$ holds, where $\ddot{T}_f$ is a cotree representing the structure of a formula for f, in which all non-linear basis functions are not distinguished from one another.

A generalization of the suggested techniques to the case of an arbitrary Boolean basis B was described in [23], which is an English translation of a paper in Russian. A proof of a universal upper bound for the basis of all five-variable Boolean functions can be found in [26]. A recent paper [9] (also an English translation) is devoted to individual checking test complexity for arbitrary bases and describes read-once functions whose checking test complexity is strictly less than that of their projections.

Read-once Boolean functions have been studied in computational learning theory for more than two decades. In 1984 Valiant suggested a polynomial algorithm using three types of queries to solve the problem of exact identification of an unknown read-once function over the basis of conjunction, disjunction and negation [20]. Further major results are due to Angluin, Hellerstein and Karpinski, who conducted a thorough study of the problem for the bases $\{\wedge, \vee, \neg\}$ and $\{\wedge, \vee\}$ [3]. More powerful algorithms for generalized versions of this problem to the case of wider bases have been developed by Bshouty, Hancock and Hellerstein [5]. Recent research due to Golumbic, Mintz and Rotics has been devoted to efficient factoring and recognition of read-once functions over the basis $\{\wedge, \vee, \neg\}$ [15]. This work also involves cographs, and makes use of the concept of normality, in the flavour of early results by Gurvich [16].

Test theory, initially developed for switching circuits, was suggested by Yablonsky and Chegis in 1958 [6], though the authors paid more attention to a diagnostic test problem — in terms of modern learning theory, that of exact identification. A similar problem for automata was studied by Moore [17]. Research in this field seems to have been fueled by development of electrical circuits, but the researchers' attention was subsequently drawn to other object regions. A somewhat extensive survey of Soviet results in the area of tests is contained in [27]. Several checking test problems for graphs (under the name of graph verification problems) were considered by Reyzin and Srivastava [18]. Diagnostic testing (learning) algorithms for graphs can also be found in [18], as well as in Aigner's book [1] and Bouvel, Grebinski and Kucherov's survey [4]. Unconditional diagnostic tests for graphs were studied by Debrev (see, e. g., [13]).

III. A COMBINATORIAL ARGUMENT ON COTREES

Our plan is as follows. We first establish an upper bound on the fraction of cotrees with n leaves having an internal node v with at least k children (i. e., such that $\deg v \geq k$). We show that if $k = k(n)$ is chosen sufficiently large, then the fraction tends to zero as $n \to \infty$. This means that almost all cographs on n vertices do not have internal nodes with more than $k-1$ children, for the chosen sequence $k = k(n)$. We then demonstrate that the characteristic $\nu(T)$ of a cotree T with n leaves cannot be greater than $C \cdot nd$, where d is the maximum number of children of internal nodes in T and C is a universal positive constant. Put together, these results give an upper bound on the checking test complexity for almost all cographs.

Remark 1: All cographs in this paper are assumed to have a specific set of vertices $X = \{x_1, \ldots, x_n\}$, where $n \to \infty$. Therefore, for each $n \geq 2$, the distribution underlying our *almost-all* statements assigns equal non-zero probabilities to all cographs on these n vertices. For example, since all graphs on three vertices are cographs, for $n = 3$ there exist three distinct cographs having exactly one edge, and only one cograph having no edges at all.

Remark 2: We say that a property holds for *almost all* cographs (cotrees, read-once functions) if the fraction ϵ_n of cographs on vertices $X = \{x_1, \ldots, x_n\}$ (cotrees with leaves X, read-once functions of variables X) not having the property tends to 0 as $n \to \infty$. However, our main bounds can be proved with a constant factor relaxation under conditions of form $\epsilon_n = O(n^{-c})$ for arbitrary $c > 0$ (details are provided in Section IV).

Let us establish an upper bound on the fraction of cotrees with n leaves having an internal node v with at least k children. To each cotree we shall assign a rooted and ordered binary tree (each internal node of the latter tree has exactly two children, called *left* and *right*, respectively). This assignment will constitute a bijection between the set of cotrees and the set of rooted and ordered binary trees.

Suppose R is a rooted and ordered binary tree. A sequence $v_1, \ldots, v_h$ of internal nodes in R is called a *right interval of length* h iff for all $i = 2, \ldots, h$ the node v_i is the right child of v_{i-1}. The node v_1 is called the *source* of the interval. If the interval cannot be extended, i. e., v_1 is either the root of R or the left child of an internal node in R, and the right child of v_h is a leaf, then the interval is called *complete*. Note that the left child of any internal node in R is the

source of exactly one complete right interval in R, and so is the root of R.

In a similar fashion *left interval* and *complete left interval* are defined. As above, we observe that the right child of any internal node in R is the source of exactly one complete left interval in R, an so is the root of R.

For the sake of simplicity, a subtree of T containing an internal node v and all its descendants is identified with v. In a binary tree, the left child subtree and the right child subtree are distinguished for every v.

We now describe the bijection announced above. Suppose T is a cotree with leaves $X = \{x_1, \ldots, x_n\}$. We obtain a binary tree R by taking T and modifying it as follows:

1) Replace each internal node v of T having k children with a right interval of length $k-1$. Each node of the interval is labeled with the same symbol as v.
2) Suppose v is an internal node in T having k children, and $T_1, T_2, \ldots, T_k$ are its child subtrees. Then all T_i become the descendants of the nodes of the right interval replacing v. In order for the transformation to be bijective, we demand that the following condition be satisfied. Whenever T_i and T_j are child subtrees of a node u in a binary tree R, the left child subtree of u shall be the one containing a leaf with the smallest variable index among all the leaves of T_i and T_j.

For convenience, the obtained tree shall be called an *ordered binary tree*.

Lemma 1: In an ordered binary tree, the following statements hold true:

- if an internal node u is the left child of v, then u and v have different labels;
- if u and v are adjacent internal nodes with the same label, then one of them is the right child of another;
- labels of nodes in any complete left interval are uniquely determined by the label of any single node in the interval (e. g., the interval's source).

The proof is trivial. We remark that labels of any two internal nodes, one of which is the right child of another, can be considered independent. One can observe that the described procedure indeed gives a bijection between the set of all cotrees and the set of all ordered binary trees. Furthermore, the obtained ordered binary tree R contains a right interval of length $k-1$, all nodes of which are labeled with identical symbols, iff the initial cotree T contains an internal node v such that $\deg v \geq k$.

Define the *index-saving structure* of an ordered binary tree R as a tree obtained from R by removing labels of all its internal nodes.

Lemma 2: Suppose S is the index-saving structure of an ordered binary tree R. Also suppose that S contains exactly m internal nodes which are right children of nodes in S. Then there exist exactly 2^{m+1} ordered binary trees having index-saving structure S.

Proof: Since S contains exactly m internal nodes which are right children of nodes in S, it can be unambiguously split into $m+1$ complete left intervals. One of these intervals has the root of S as its source, and other intervals' sources are those m right children. By Lemma 1, each of these intervals can be labeled in two different ways. Since labels of two distinct left intervals are independent, we obtain the desired. ■

The previous definition splits the set of all ordered binary trees into classes of trees having identical index-saving structures. We now obtain an upper bound on the fraction of ordered binary trees containing an interval of length $k-1$ of nodes with identical labels. This bound will lead us to the bound on the fraction of cotrees containing a node with k or more children.

Lemma 3: The fraction of cotrees with n leaves containing a node with k or more children is less or equal to

$$\frac{n-1}{2^{k-2}}.$$

Proof: We recall that a node with k or more children in a cotree is transformed into an interval of length $k-1$ or greater, all nodes of which are labeled with identical symbols. Let S be an index-saving structure with n leaves. Since a right interval of length $k-1$ is uniquely determined by its source, and a binary tree with n leaves has exactly $n-1$ internal nodes, it follows that the number of right intervals of length $k-1$ in S cannot be greater than $n-1$.

Now suppose that S contains m internal nodes which are right children. Then S can be split into $m+1$ complete left intervals. Let us choose an arbitrary right interval of length $k-1$ in S (there are no more than $n-1$ ways of doing this) and label all its nodes with identical symbols (here the number of possibilities is 2). The labeled nodes uniquely determine the labels of $k-1$ complete left intervals containing the nodes of the chosen right interval.

All remaining $m+1-(k-1)$ complete left intervals are then labeled arbitrarily; the number of possibilities is 2^{m-k+2}. Therefore, in the set of all ordered binary trees having index-saving structure S, no more than $(n-1)\cdot 2\cdot 2^{m-k+2}$ contain a right interval of length $k-1$, all labels in which are identical. Note that whenever a tree contains a right interval of length greater than $k-1$, or more than one right interval of length $k-1$ (or greater) such that all the labels inside each said interval are identical, then this tree is "counted" more than once, which does not render our upper bound invalid.

According to Lemma 2, the number of all ordered binary trees with index-saving structure S is equal to 2^{m+1}. Hence, the fraction of trees containing a right interval of length $k-1$ with identical node labels in the set of all ordered binary trees having index-saving structure S is less or equal to $(n-1)/2^{k-2}$. Since this bound does not depend on m or S, it holds true for the set of all ordered binary trees with

n leaves. If we substitute cotrees for ordered binary trees using the bijection described above, we obtain the desired. ■

Theorem 2: For any function

$$k(n) = \log_2 n + r(n)$$

such that $r(n) \to +\infty$ as $n \to \infty$, almost all cotrees with n leaves contain no nodes having k or more children.

Proof: Under the conditions of the theorem, we have

$$\frac{n-1}{2^{k-2}} = 2^{\log_2 n - k + 2} \cdot \frac{n-1}{n} = 2^{-r(n)+2} \cdot (1 - o(1)) \to 0.$$

■

IV. Upper bounds on checking test complexity

Our goal now is to establish a tight upper bound on the characteristic $\nu(T)$ of a cotree T containing no internal nodes with more than, say, k children. In such a tree, for all internal nodes v the inequality $b(v) \le k-1$ holds. First, suppose that $b(v) = k-1$ for all v. From (1), it follows that

$$\nu(T) = |\operatorname{int} \mathrm{E}(T)| \cdot (k-1)^2 + \frac{1}{2} \cdot |\operatorname{int} \mathrm{V}(T)| \cdot (k-1)^2 + \frac{n-1}{2}.$$

Since $|\operatorname{int} \mathrm{V}(T)| \cdot k = |\operatorname{int} \mathrm{V}(T)| + n - 1$ and $|\operatorname{int} \mathrm{E}(T)| = |\operatorname{int} \mathrm{V}(T)| - 1$, we have

$$\begin{aligned}\nu(T) &= (n-1) \cdot (k-1) - (k-1)^2 \\ &\quad + \frac{1}{2} \cdot (n-1) \cdot (k-1) + \frac{n-1}{2} \\ &= \frac{3}{2} \cdot (n-1) \cdot \left(k - \frac{2}{3}\right) - (k-1)^2,\end{aligned}$$

and

$$\nu(T) \le \frac{3nk}{2}.$$

However, if some internal nodes have less than k children, then the cardinality $|\operatorname{int} \mathrm{V}(T)|$ cannot be bounded by $C \cdot (n-1)/(k-1)$. In fact, it may be as big as $n-1$, which gives us only

$$\nu(T) \le \frac{3nk^2}{2}.$$

While this yields an $O(n \log^2 n)$ upper bound on $\nu(T)$ for almost all cotrees T, we show how to tighten the bound on $\nu(T)$. We need an auxiliary lemma on edge contraction in cotrees.

Lemma 4: If a cotree T' is obtained from a cotree T by contracting several edges between internal nodes, then

$$\nu(T) \le \nu(T').$$

Proof: Suppose that T' is obtained from T by contraction of exactly one edge $\{u, v\}$. Let u be the parent of v. By $u_1, \ldots, u_{b(u)}$ denote the rest of the children of u, and by $v_0, v_1, \ldots, v_{b(v)}$ denote the children of v. We have

$$\begin{aligned}\nu(T') - \nu(T) &= \frac{1}{2}\left(b(u) + b(v)\right)^2 \\ &\quad - \frac{1}{2}\left(b^2(u) + b^2(v)\right) - b(u)b(v) \\ &\quad + b(v) \sum_{i=1}^{b(u)} b(u_i) + b(u) \sum_{i=0}^{b(v)} b(v_i) \\ &\quad + b(v)\, bp(u),\end{aligned}$$

where $bp(u) = 0$ if u is the root of T and $bp(u) = b(w)$ if T contains a parent w of u. It follows that

$$\nu(T') = \nu(T) + b(v) \sum_{i=1}^{b(u)} b(u_i) + b(u) \sum_{i=0}^{b(v)} b(v_i) + b(v)\, bp(u),$$

which proves the desired for the considered special case. One can easily see that for any number of edge contractions the inequality $\nu(T) \le \nu(T')$ holds. ■

We can now prove an upper bound on $\nu(T)$ which is linear in $n \cdot k$.

Lemma 5: If all internal nodes in a cotree T with n leaves have no more than k children, then

$$\nu(T) \le 4nk.$$

Proof: Take T and contract all internal edges $\{u, v\}$ such that $b(u) + b(v) \le k - 1$ into single nodes (all edges incident to leaves are left untouched). Repeat the procedure until for all pairs of adjacent internal nodes u and v it holds that $b(u) + b(v) > k - 1$. Denote the obtained tree by T'. By Lemma 4, $\nu(T) \le \nu(T')$.

We now prove an upper bound on $\nu(T')$. Define

$$\begin{aligned}V' &= \{v \in \operatorname{int} \mathrm{V}(T') \mid b(v) \le h\}, \\ V'' &= \{v \in \operatorname{int} \mathrm{V}(T') \mid b(v) > h\},\end{aligned}$$

where $h = (k-1)/2$. By construction of T', it contains no edges between nodes in V'. Furthermore, since for all $v \in V''$ it holds that $b(v) > h$, it then follows that $n - 1 > h \cdot |V''|$, since the sum of $b(v)$ over all internal nodes v is equal to $n - 1$.

Now recall that

$$\nu(T') = \sum_{\{u,w\} \in \operatorname{int} \mathrm{E}(T)'} b(u)\, b(w) + \frac{1}{2} \sum_{v \in \operatorname{int} \mathrm{V}(T)'} b^2(v) + \frac{n-1}{2},$$

Consider the first sum in this formula. Denote by v_0 the root of T' and by $\operatorname{anc}(v)$ the parent of a non-root internal

node v. We have

$$\begin{aligned}\sum_{\{u,v\}\in \operatorname{int} \mathrm{E}(T')} b(u)\, b(v) &= \sum_{v\in V'\setminus\{v_0\}} b(v)\, b(\operatorname{anc}(v)) \\ &\quad + \sum_{v\in V''\setminus\{v_0\}} b(v)\, b(\operatorname{anc}(v)) \\ &\le \sum_{v\in V'} b(v)(k-1) + \sum_{v\in V''} (k-1)^2 \\ &= (k-1)\cdot \sum_{v\in V'} b(v) + (k-1)^2\cdot |V''| \\ &\le (k-1)\,(n-1) + 2\,(k-1)\,(n-1) \\ &= 3\,(k-1)\,(n-1).\end{aligned}$$

Now take the second sum. Clearly, the sum of $b^2(v)$ over all $v \in \operatorname{int} \mathrm{V}(T')$ is less or equal to

$$M = \max \sum_{i=1}^{n-1} y_i^2,$$

where $0 \le y_i \le k-1$ for $1 \le i \le n-1$ and $\sum_{i=1}^{n-1} y_i = n-1$. If $0 < y_i \le y_j < k-1$, then

$$\begin{aligned}\left((y_i-1)^2 + (y_j+1)^2\right) - \left(y_i^2 + y_j^2\right) &= 2\,(y_j - y_i + 1) \\ &\ge 2\cdot 1 > 0\end{aligned}$$

and $(y_1, \ldots, y_{n-1})$ does not bring the maximum. Hence, the maximizing vector consists only of 0's, $(k-1)$'s and at most one number from $(0; k-1)$. Suppose that $p\cdot(k-1)+q = n-1$, where $p, q \ge 0$ and $q < k-1$. We obtain

$$\begin{aligned}\sum_{v\in \operatorname{int} \mathrm{V}(T')} b^2(v) &\le M = p\cdot(k-1)^2 + q^2 \\ &\le (p+1)\cdot(k-1)^2 \\ &\le 2p\cdot(k-1)^2 \\ &\le 2\,(n-1)\,(k-1).\end{aligned}$$

Combining the bounds yields

$$\nu(T') \le 3\,(n-1)\,(k-1) + \frac{1}{2}\cdot 2\,(n-1)\,(k-1) + \frac{n-1}{2} \le 4nk,$$

which concludes the proof. ∎

Remark 3: The bound of Lemma 5 cannot be improved by more than a constant factor. Indeed, if all internal nodes of a cotree T have exactly k children, then, as shown above,

$$\begin{aligned}\nu(T) &= \frac{3}{2}\cdot(n-1)\cdot\left(k - \frac{2}{3}\right) - (k-1)^2 \\ &= \frac{3}{2}\cdot nk - k^2 - n + \frac{k}{2}.\end{aligned}$$

First suppose that $k \le C$ for some fixed C as $n \to \infty$, then

$$\nu(T) = \left(\frac{3}{2}\cdot k - 1\right)\cdot n\cdot(1 - o(1)).$$

Now suppose that $k \to \infty$. In this case, we have

$$\begin{aligned}\nu(T) &= \frac{nk}{2}\cdot\left(3 - 2\cdot\frac{k}{n}\right) - n + \frac{k}{2} \\ &\ge \frac{nk}{2} - n + \frac{k}{2} \\ &= \frac{nk}{2}\cdot(1 - o(1)).\end{aligned}$$

We observe now that the statement of Theorem 1 follows from Theorem 2 and Lemma 5, for $r(n) = o(\log n) \to +\infty$ as $n \to \infty$.

Remark 4: Choosing $k(n) = (c+1)\log_2 n$, $c > 0$, in Theorem 2 leads to the bound

$$\nu(T) \le 4(c+1)n\log_2 n,$$

which is still $O(n\log n)$ and holds for all but the fraction of

$$2^{-c\log_2 n + 2}\cdot\left(1 - \frac{1}{n}\right) = 4\cdot\left(\frac{1}{n}\right)^c\cdot\left(1 - \frac{1}{n}\right) = O(n^{-c})$$

cographs.

Recall that for any read-once function $f(x_1, \ldots, x_n)$ over the basis of disjunction and sum modulo 2 (parity) its checking test complexity with respect to read-once alternatives either over the same basis or over the basis of all two-variable functions is less or equal to $\nu(T_f) + n + 1$, where T_f is a tree representing the structure of a read-once formula for f. Combining this with the statement of Theorem 1, we obtain the following corollary:

Corollary 1: Checking test complexity of almost all read-once functions of n variables over the basis of disjunction and sum modulo 2 (parity) with respect to read-once alternatives either over the same basis or over the basis of all two-variable functions is less or equal to

$$4n\log_2 n\cdot(1 + o(1)).$$

Remark 5: Here we employ the fact that our cotree representation of read-once functions over $\{\vee, \oplus\}$ is a one-to-one correspondence (see, e. g., [25]). For each n, we consider functions as mappings from $\{0,1\}^n$ to $\{0,1\}$ and assign equal non-zero probabilities to all read-once functions depending on variables $X = \{x_1, \ldots, x_n\}$. For example, formulae $(x_1 \vee x_2) \oplus x_3$ and $(x_2 \vee x_1) \oplus x_3$ express the same function, different from the one expressed by $(x_1 \vee x_3) \oplus x_2$.

Similarly to the case of cographs, the obtained value is relatively small compared to the maximum possible value of $\binom{n}{2} + n + 1$, which is the exact value of checking test complexity of n-ary disjunction. Arguing as above, one can also obtain an $O(n\log n)$ bound for all but the fraction of $O(n^{-c})$ read-once functions.

V. Discussion

We obtained an $O(n \log n)$ upper bound on checking test complexity of almost all cographs and deduced a corollary on said complexity of almost all read-once functions over the basis of disjunction and sum modulo 2 (parity). These bounds reveal that almost all cographs G on n vertices (or, similarly, almost all read-once functions f of variables $x_1, \ldots, x_n$ over $\{\vee, \oplus\}$) are relatively easy to "certify": if one knows for sure that a given graph is a cograph (or, similarly, that a given Boolean function is read-once over $\{\vee, \oplus\}$), then an $O(n \log n)$ amount of information is sufficient to check whether the given graph (the given function) is equal to G (f). This information can be obtained by checking the presence of edges (computing function's value at input vectors) pointed to by a checking test. It is important to note that cograph recognition can be performed in linear time [12].

We believe that our bounds are optimal in terms of $O(\cdot)$, i. e., they cannot be improved by more than a constant factor. For the problem of testing with respect to read-once alternatives, we expect that analogous bounds hold for wider bases, even though individual lower bounds for some functions are provably as high as n^l for any $l \in \mathbb{N}$ [24]. An approach based on similar techniques for read-once functions over $\{\wedge, \vee\}$ gives not only bounds, but also exact values of checking test complexity, though the results computed so far look quite different [10].

As indicated above, checking tests are closely related to equivalence queries in the context of exact identification problems. This connection should be treated with great care, since minor details in definitions can turn out to be of major importance. In our definition of a checking test for read-once functions, the target function f is assumed to depend essentially on all its variables, though its alternatives are not subject to the same restriction. If this restriction on f is not imposed either, a problem of distinguishing $f(x_1, \ldots, x_n) \equiv 0$ from all conjunctions of n literals emerges. Since all such conjunctions are read-once over the basis $\{\wedge, \vee, \neg\}$, all 2^n vectors must be included in the checking test for this f. At the same time, if all variables of the target function are known to be essential, then for any n-variable read-once function over this basis there exists a checking test containing $O(n)$ vectors [22].

The problem of closing this complexity gap in the context of exact identification can be addressed by introducing auxiliary oracles. In a learning model considered in [7], the learner is allowed, in addition to standard membership queries, to ask questions of the following form: "Can the value of f be determined unambiguously if only some variable assignments, namely $x_{i_1} = \alpha_{i_1}, \ldots, x_{i_k} = \alpha_{i_k}$, are known?" Such questions, on the one hand, seem quite natural for the problem of identifying the unknown function and, on the other hand, enable the use of modeling techniques for checking tests in a simulation of an equivalence query. Large classes of read-once Boolean functions are shown to be exactly identifiable in this model.

Acknowledgment

This work was supported by Russian Foundation for Basic Research, project numbers 09-01-00701 and 09-01-00817, and by Russian Presidential grant MD-757.2011.9.

References

[1] M. Aigner, *Combinatorial search*. John Wiley and Sons, 1988.

[2] D. Angluin, "Queries and concept learning," *Machine Learning*, vol. 2, pp. 319–342, 1987.

[3] D. Angluin, L. Hellerstein, M. Karpinski, "Learning read-once formulas with queries," *Journal of the ACM*, vol. 40, pp. 185–210, 1993.

[4] M. Bouvel, V. Grebinski, G. Kucherov, "Combinatorial search on graphs motivated by bioinformatics applications: a brief survey," in *Graph-Theoretic Concepts in Computer Science 2005*, ser. Lecture Notes in Computer Science, vol. 3787, pp. 16–27.

[5] N. H. Bshouty, T. R. Hancock, L. Hellerstein, "Learning Boolean read-once formulas over generalized bases," *Journal of Computer and System Sciences*, vol. 50, no. 3, pp. 521–542, 1995.

[6] I. A. Chegis, S. V. Yablonsky, "Logical methods for controlling electrical circuits," in *Trudy matematicheskogo instituta imeni V. A. Steklova* (*in Russian*), 1958, vol. 51, pp. 270–360.

[7] D. V. Chistikov, "On the relationship between diagnostic and checking tests of the read-once functions," *Discrete Mathematics and Applications*, vol. 21, no. 2, pp. 203–208, 2011.

[8] D. V. Chistikov, "On one characteristic of trees related to individual testing of read-once functions," in *Proc. XVIII International Workshop "Synthesis and complexity of control systems"* (*in Russian*), Penza, 2009, pp. 94–99.

[9] D. V. Chistikov, "Read-once functions with hard-to-test projections," *Moscow University Computational Mathematics and Cybernetics*, vol. 34, no. 4, pp. 188–190, 2010.

[10] D. V. Chistikov, "Testing monotone read-once functions," in *Proc. 22nd International Workshop on Combinatorial Algorithms*, ser. Lecture Notes in Computer Science, 2011, vol. 7056, pp. 121–134.

[11] D. G. Corneil, H. Lerchs, L. Stewart Burlingham, "Complement reducible graphs," *Discrete Applied Mathematics*, vol. 3, no. 3, pp. 163–174, 1981.

[12] D. G. Corneil, Y. Perl, L. K. Stewart, "A linear recognition algorithm for cographs," *SIAM Journal of Computing*, vol. 14, no. 4, pp. 926–934, 1985.

[13] E. V. Debrev, "On a combinatorial search problem," *Discrete Mathematics and Applications*, vol. 12, no. 4, pp. 325–335, 2002.

[14] S. A. Goldman, M. J. Kearns, "On the complexity of teaching," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 20–31, 1995.

[15] M. C. Golumbic, A. Mintz, U. Rotics, "Factoring and recognition of read-once functions using cographs and normality and the readability of functions associated with partial k-trees," *Discrete Applied Mathematics*, vol. 154, pp. 1465–1477, 2006.

[16] V. A. Gurvich, "On read-once Boolean functions," *Uspehi matematicheskih nauk* (*in Russian*), vol. 32, no. 1, pp. 183–184, 1977.

[17] E. F. Moore, "Gedanken-experiments on sequential machines," in *Automata Studies*, ser. Annals of Mathematical Studies, Princeton University Press, 1956, vol. 34, pp. 129–153.

[18] L. Reyzin, N. Srivastava, "Learning and verifying graphs using queries with a focus on edge counting," in *Algorithmic Learning Theory 2007*, ser. Lecture Notes in Computer Science, vol. 4754, pp. 285–297.

[19] L. V. Ryabets, *Checking test complexity for read-once Boolean functions* (*in Russian*), ser. Diskretnaya matematika i informatika, Izdatel'stvo Irkutskogo gosudarstvennogo pedagogicheskogo universiteta, 2007, vol. 18.

[20] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, 27, pp. 1134–1142, 1984.

[21] A. A. Voronenko, "On checking tests for read-once functions," in *Matematicheskie voprosy kibernetiki* (*in Russian*). Moscow: Fizmatlit, 2002, vol. 11, pp. 163–176.

[22] A. A. Voronenko, "On the length of checking test for repetition-free functions in the basis $\{0, 1, \&, \vee, \neg\}$," *Discrete Mathematics and Applications*, vol. 15, no. 3, pp. 313–318, 2005.

[23] A. A. Voronenko, "Recognizing the nonrepeating property in an arbitrary basis," *Computational Mathematics and Modeling*, vol. 18, no. 1, pp. 55–65, 2007.

[24] A. A. Voronenko, "Testing disjunction as a read-once function in an arbitrary unrepeated basis," *Moscow University Computational Mathematics and Cybernetics*, vol. 32, no. 4, pp. 239–240, 2008.

[25] A. A. Voronenko, D. V. Chistikov, "Learning read-once functions individually," *Uchenye zapiski Kazanskogo universiteta* (*in Russian*), *ser. Fiziko-matematicheskie nauki*, vol. 151, no. 2, pp. 36–44, 2009.

[26] A. A. Voronenko, D. V. Chistikov, "On testing read-once Boolean functions in the basis B_5," in *Proc. XVII International Workshop "Synthesis and complexity of control systems"* (*in Russian*). Novosibirsk, 2008, pp. 24–30.

[27] S. V. Yablonsky, "Several questions of reliability and control for controlling systems," in *Matematicheskie voprosy kibernetiki* (*in Russian*). Moscow: Nauka. Fizmatlit, 1988, vol. 1, pp. 5–25.