ON P VERSUS NP∩CO-NP FOR DECISION TREES AND READ-ONCE BRANCHING PROGRAMS

S. Jukna, A. Razborov, P. Savický
and I. Wegener

Abstract. It is known that if a Boolean function f in n variables has a DNF and a CNF of size $\leq N$ then f also has a (deterministic) decision tree of size $\exp\left(O(\log n \log^2 N)\right)$. We show that this simulation *cannot* be made polynomial: we exhibit explicit Boolean functions f that require deterministic trees of size $\exp\left(\Omega(\log^2 N)\right)$ where N is the total number of monomials in minimal DNFs for f and $\neg f$. Moreover, we exhibit new examples of explicit Boolean functions that require deterministic read-once branching programs of exponential size whereas both the functions and their negations have small nondeterministic read-once branching programs. One example results from the Bruen-Blokhuis bound on the size of nontrivial blocking sets in projective planes: it is remarkably simple and combinatorially clear. Whereas other examples have the additional property that f is in AC^0.
Key words. Computational complexity, Boolean functions, decision trees, branching programs, P versus NP∩co-NP
Subject classifications. 68Q15, 94C10

1. Introduction

The following general question has been widely studied for various computational models:

- *Suppose that both a computational problem f and its complement $\neg f$ possess an efficient nondeterministic computation in some model. Does this imply that f can also be computed efficiently* (*typically, with at most polynomial blow-up*) *and* **deterministically** *in the same model*?

We use for this question somewhat imprecise but very expressive the abbreviation *the* P *versus* NP∩co-NP *question*, and, since we study the models in

Boolean (non-uniform) complexity, we assume in this notation polynomial size instead of polynomial time. More specifically, we study the P versus NP$\cap$co-NP question for decision trees and read-once branching programs (see e.g. Wegener 1987, Razborov 1991).

A (deterministic) *branching program* is a directed acyclic graph with one source whose sinks have labels from $\{0,1\}$ and whose other nodes called internal nodes have labels from $\{1,\ldots,n\}$. If an internal node has label i then one of the outgoing edges is labelled by the test $x_i = 0$ and the other by $x_i = 1$. Every branching program computes a Boolean function f in a natural way: for $a \in \{0,1\}^n$ we simply follow the unique path p consistent with the input a (i.e., all edge labels have the form $x_i = a(i)$), and output the label of the leaf finally reached by p.

A *decision tree* is a branching program whose underlying graph is a tree. The size of a decision tree is defined as the number of its leaves. By $dt(f)$ we denote the minimum size of a decision tree for f.

A *read-once branching program* (1-b.p.) is a branching program such that each path contains at most one internal node labelled by i, $1 \le i \le n$. The size of 1-b.p. is the number of its nodes.

A nondeterministic branching program additionally may contain unlabelled nodes. In order to accept an input it is necessary and sufficient that at least one path consistent with the input reaches a 1-sink.

In the context of decision trees Ehrenfeucht & Haussler (1989) have proved that every boolean function f in n variables has a (deterministic) decision tree of size

$$\exp(O(\log n \log^2 N))$$

where N is the total number of monomials in the minimal DNFs for f and $\neg f$. This result is not stated explicitly in Ehrenfeucht & Haussler (1989), so we describe it more precisely in Theorem 2.1 below. For the analogous result about the *depth* of decision trees see Blum & Impagliazzo (1987), Hartmanis & Hemachandra (1987), Tardos (1989). Since nondeterministic decision trees are essentially equivalent to DNFs, this upper bound states that for decision trees we have NP$\cap$co-NP $\subseteq \tilde{\mathrm{P}}$ where $\tilde{\mathrm{P}}$ stands for *quasipolynomial* time. Ehrenfeucht & Haussler (1989) asked whether their bound can be improved. We prove that it is optimal up to a factor of $\log n$ in the exponent. In particular, NP$\cap$co-NP $\not\subseteq$ P in the context of decision trees. For this purpose we show that the ITERATED MAJORITY function and the ITERATED NAND function considered for a related purpose already in Saks & Wigderson (1986) require deterministic decision trees of sizes $\exp\left(\Omega(\log^{1.58} N)\right)$ and $\exp\left(\Omega(\log^2 N)\right)$, respectively (Theorems 2.3, 2.4).

In the rest of the paper, we deal with read-once branching programs. The first example of a function showing that in this context NP∩co-NP is not even in subexponential time was given in Jukna (1988). In the present paper, we provide new examples of this kind. One of them is remarkably simple and combinatorially clear, whereas the other two have the additional property that the separating function is in AC^0. More specifically:

1. We show that the characteristic function f of blocking sets of appropriately bounded cardinality in a finite projective plane requires 1-b.p. of exponential size whereas both f and $\neg f$ have polynomial size 1-n.b.p. of very transparent structure (Theorem 3.2).

2. We exhibit an explicit Boolean function f such that both f and $\neg f$ not only have 1-n.b.p. of polynomial size, but can also be computed by polynomial size depth-3 circuits. Whereas any 1-b.p. computing f still must have exponential size (Theorem 3.3).

3. We exhibit an explicit Boolean function f such that both f and $\neg f$ have extremely small 1-n.b.p., *both* can be computed by Σ_3^p-circuits whereas the minimal 1-b.p. computing f has quasipolynomial size (Theorem 3.4).

2. Decision Trees

In this section we establish the announced bounds for decision trees.

Let $\mathrm{dnf}(f)$ denote the minimum number of monomials in a DNF for f. As we already noted in the introduction, this is essentially the minimum size of a *nondeterministic* decision tree for f, and we do not distinguish between these two complexity measures. The sum $\|f\| \rightleftharpoons \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$ will be called the *weight* of f.

The following result is due to Ehrenfeucht & Haussler (1989). Although not stated explicitly the result follows directly from Lemma 1 and Lemma 6 of that paper.

THEOREM 2.1. *Let f be a nonconstant Boolean function in n variables, $s \rightleftharpoons \mathrm{dnf}(f)$, $t \rightleftharpoons \mathrm{dnf}(\neg f)$, and*

$$r(f) \rightleftharpoons \lfloor \log_2(s+t)\ln(st) \rfloor + 1$$

Then f has a (deterministic) decision tree of a size bounded by

$$\sum_{i=0}^{r(f)} \binom{n}{i} = \exp\Big(O(\log n \log^2 N)\Big)$$

where $N \rightleftharpoons s + t$ is the weight of f.

In the following we will show that the simulation given by Theorem 2.1 cannot be made polynomial. We demonstrate this by proving lower bounds on the decision tree size for two explicit functions introduced in Saks & Wigderson (1986).

Our argument is based on spectral methods, so it will be convenient to switch to the $(-1, +1)$-notation, i.e., to consider Boolean functions as mappings from $\{-1, 1\}^n$ to $\{-1, 1\}$, where the correspondence $true = -1$ and $false = 1$ is assumed. These functions are treated as elements of a 2^n-dimensional vector space with an inner product defined by $\langle f, g\rangle \rightleftharpoons 2^{-n} \sum_x f(x)g(x)$. The set of all monomials $X_S \rightleftharpoons \prod_{i\in S} x_i$ forms an orthonormal basis for this space. Hence, any function f on the cube $\{-1, 1\}^n$ can be uniquely expressed as $f = \sum_S \hat{f}(S)X_S$, where

$$\hat{f}(S) \rightleftharpoons \langle f, X_S\rangle = 2^{-n} \sum_x f(x)X_S(x)$$

is the Sth Fourier coefficient of f. The unique expression of f as a linear combination of X_S may be obtained from any real polynomial expressing f using identities $x_i^2 = 1$.

The following lemma combines Linial *et al.* (1989) (Lemma 4) and Kushilevitz & Mansour (1991) (Lemma 5.1).

LEMMA 2.2. *For every Boolean function f in n variables and every $S \subseteq [n]$ we have the bound*

$$\mathrm{dt}(f) \geq 2^{|S|} \cdot \sum_{T\supseteq S} |\hat{f}(T)|. \tag{2.1}$$

PROOF. Take a decision tree for f of size $\mathrm{dt}(f)$. For a leaf ℓ let $\mathrm{val}(\ell) \in \{-1, 1\}$ be its label (recall that we are in the $(-1, +1)$-notation), and let I_ℓ be the set of indices of those variables, which are tested on the path to ℓ. Let $B_\ell \subseteq \{-1, 1\}^n$ be the set of all the inputs that reach leaf ℓ. We think of B_l as of the corresponding ((0,1)-valued!) function; in particular, we let

$$\hat{B}_\ell(T) \rightleftharpoons 2^{-n} \sum_{x\in B_\ell} X_T(x).$$

Since each input reaches a unique leaf the sets B_ℓ are mutually disjoint, and, hence, $\hat{f}(T) = \sum_\ell \mathrm{val}(\ell) \cdot \hat{B}_\ell(T)$ for every $T \subseteq [n]$. Now, if $T \not\subseteq I_\ell$ then $X_T(x) = +1$ for exactly half of the inputs $x \in B_\ell$, and, hence, $\hat{B}_\ell(T) = 0$. If $T \subseteq I_\ell$ then the value of X_T is fixed on B_ℓ to either $+1$ or -1, and hence, $|\hat{B}_\ell(T)| = 2^{-n} \cdot |B_\ell| = 2^{-|I_\ell|}$.

For any $S \subseteq [n]$ an any ℓ, there are at most $2^{|I_\ell|-|S|}$ sets T satisfying $S \subseteq T \subseteq I_\ell$. Hence,

$$\sum_{T \supseteq S} |\hat{f}(T)| \leq \sum_{T \supseteq S} \sum_\ell \left|\hat{B}_\ell(T)\right| = \sum_\ell \sum_{T : I_\ell \supseteq T \supseteq S} 2^{-|I_\ell|} = \sum_\ell 2^{-|S|} = 2^{-|S|} \cdot \mathrm{dt}(f),$$

and the desired bound (2.1) follows. □

We are going to apply Lemma 2.2 for $S = [n]$ to the function ITERATED MAJORITY function and for $S = \emptyset$ to the ITERATED NAND function.

Consider the monotone function in $n = 3^h$ variables which is computed by the balanced read-once formula of height h in which every gate is MAJ_3, the majority of 3 variables. Let us denote this function by F_h.

Theorem 2.3. $\mathrm{dt}(F_h) \geq \exp\left(\Omega(\log^\gamma N_h)\right)$, *where* $N_h \rightleftharpoons \|F_h\|$ *and* $\gamma \rightleftharpoons \log_2 3 \geq 1.58$.

Proof. It is easy to see that $\mathrm{dnf}(F_h) = 3 \cdot \mathrm{dnf}(F_{h-1})^2$ and $\mathrm{dnf}(F_0) = 1$. Moreover, the function is self-dual and hence the DNF size of $\neg F_h$ coincides with that of F_h. Thus, $N_h = 2 \cdot 3^{2^h - 1}$ and $n = 3^h = \theta(\log^\gamma N_h)$. So, our goal is to prove the exponential bound

$$\mathrm{dt}(F_h) \geq \exp\left(\Omega(n)\right).$$

By Lemma 2.2 with $S = [n]$, it is sufficient to prove an appropriate lower bound on the leading Fourier coefficient $\hat{F}_h([n])$ of the polynomial $F_h(x_1, x_2, \ldots, x_n)$. We denote this coefficient by a_h and proceed by induction on h.

Clearly, $a_0 = 1$, since F_0 is a variable.

For the inductive step note that $MAJ_3(x_1, x_2, x_3) = (x_1 + x_2 + x_3 - x_1 x_2 x_3)/2$ in the $(-1, +1)$-representation. Since $F_h = MAJ_3(F_{h-1}^{(1)}, F_{h-1}^{(2)}, F_{h-1}^{(3)})$, where $F_{h-1}^{(\nu)}$ are three copies of F_{h-1} with disjoint sets of variables, we have

$$F_h = \frac{1}{2}\left(\sum_{\nu=1}^{3} F_{h-1}^{(\nu)} - \prod_{\nu=1}^{3} F_{h-1}^{(\nu)}\right).$$

Since F_{h-1} depends on less than $n = 3^h$ variables, the first summand does not contribute to the leading coefficient in F_h. Thus, we have the recursion

$$a_h = -\frac{1}{2}a_{h-1}^3.$$

This resolves to

$$a_h = (-1)^h \cdot \left(\frac{1}{2}\right)^{(3^h-1)/2} = (-1)^h \cdot \left(\frac{1}{2}\right)^{(n-1)/2}$$

and gives us (from Lemma 2.2) the bound $\mathrm{dt}(F_h) \geq 2^{(n+1)/2}$, as desired. □

Consider the function in $n = 2^h$ variables which is computed by the balanced read-once formula of height h in which every gate is NAND, the negated AND operation $\mathrm{NAND}(x, y) = \neg x \vee \neg y$. (up to complementation of the inputs this is equivalent to a monotone read-once formula with alternating levels of AND and OR gates). Let us denote this function by G_h.

THEOREM 2.4. $\mathrm{dt}(G_h) \geq \exp\left(\Omega(\log^2 N_h)\right)$, *where* $N_h \rightleftharpoons \|G_h\|$.

PROOF. $\mathrm{dnf}(G_0) = \mathrm{dnf}(\neg G_0) = 1$ (since G_0 is a single variable), and it is easy to see that for every $h \geq 1$ we have $\mathrm{dnf}(G_h) \leq 2 \cdot \mathrm{dnf}(\neg G_{h-1})$ and $\mathrm{dnf}(\neg G_h) \leq \mathrm{dnf}(G_{h-1})^2$. By induction on h one obtains $\mathrm{dnf}(G_h) \leq 2^{2^{(h+1)/2}-1}$ and $\mathrm{dnf}(\neg G_h) \leq 2^{2^{(h/2)+1}-2}$. Hence, we have $N_h \leq 2^{2^{(h/2)+1}}$. Since $n = 2^h$, our statement boils down to showing

$$\mathrm{dt}(G_h) \geq \exp\left(\Omega(n)\right).$$

Let us say that a Fourier coefficient $\hat{G}_h(S)$ is *dense* if for every subtree of height 2, S contains the index of at least one of the four variables in that subtree. We are going to calculate exactly the sum of absolute values of dense coefficients. Denote this sum by c_h. Note that in the $(-1, +1)$-representation, we have $\mathrm{NAND}(x, y) = (xy - x - y - 1)/2$. Hence,

$$G_h = \frac{1}{2}\left(G_{h-1}^{(1)} \cdot G_{h-1}^{(2)} - G_{h-1}^{(1)} - G_{h-1}^{(2)} - 1\right), \tag{2.2}$$

where $G_{h-1}^{(1)}$, $G_{h-1}^{(2)}$ are two copies of G_{h-1} with disjoint sets of variables.

In order to compute c_2, we use the following transformation. Let $f_1 = G_1^{(1)} + 1/2$ and $f_2 = G_1^{(2)} + 1/2$. Then it follows from (2.2) that

$$G_2 = \frac{1}{2}f_1 f_2 - \frac{3}{4}f_1 - \frac{3}{4}f_2 + \frac{1}{8}.$$

Since each monomial in f_1 and f_2 contains at least one variable and the sets of variables of f_1 and f_2 are disjoint, there are no common monomials in the four terms in the above expression of G_2. Hence, it is easy to calculate the sum of the absolute values of the coefficients in the nonconstant monomials, which is $c_2 = 1/2 \cdot r_1 \cdot r_2 + 3/4 \cdot (r_1 + r_2) = 27/8 = 3.375$, where $r_1 = r_2 = 3/2$ is the sum of the absolute values of the coefficients in f_1 and f_2.

In order to compute c_h for $h > 2$, we use (2.2) directly. Only the first term $G^{(1)}_{h-1} \cdot G^{(2)}_{h-1}$ in this equation can contribute to dense coefficients, and its individual contributions do not cancel each other. Hence, we have the recursion

$$c_h = \frac{1}{2}\, c^2_{h-1}.$$

This resolves to $c_h = 2(c_2/2)^{2^{h-2}}$ which is $\exp(\Omega(n))$ since $c_2 > 2$. The proof is now completed by applying Lemma 2.2 (this time with $S = \emptyset$). □

3. Read-Once Branching Programs

In this section we investigate functions which separate P from NP∩co-NP for read-once branching programs. For simplicity, let us call such functions *separating.* The first example of a separating function was presented in Jukna (1988) (Example 6.14) and another separating function of a similar flavour was discovered in Savický & Žák (1996).

We first present a separating function of quite a different nature, which has a surprisingly compact combinatorial definition. It is the characteristic function of a certain system of point-sets in a projective plane $\mathrm{PG}(2, q)$.

Then we present examples of separating functions in AC^0. Let Σ^p_d, Π^p_d be the classes of functions computable by polynomial size depth-d circuits over the de Morgan basis {AND, OR, NOT} (negations are allowed only at the input variables and do not contribute to the depth) that have OR (respectively, AND) as output gate. Our first example is in Σ^p_3 and demonstrates an exponential separation. The second example belongs to the smaller class $\Sigma^p_3 \cap \Pi^p_3$, but the separation is only quasipolynomial.

All the lower bounds for 1-b.p. in this section are based on the following “folklore observation” (see Jukna 1988, or Gál 1997 for a simple proof). Let us say that a Boolean function f in n variables is *k-mixed* if for any $I \subseteq [n]$ with $|I| = k$, and any two different assignments $a, b : I \to \{0,1\}$, there is an assignment $c : [n] \setminus I \to \{0,1\}$ for which $f(a, c) \neq f(b, c)$.

Lemma 3.1. *If f is k-mixed then any 1-b.p. computing f must have size at least $2^k - 1$.*

3.1. 1-b.p. versus 1-n.b.p.. The size of DNF (or CNF) and the size of 1-b.p. are in general incomparable:

- the *parity* function has a small 1-b.p. but requires exponential size AC^0 circuits Furst *et al.* (1981), Ajtai (1983), Yao (1985), Håstad (1989), and
- the *exact-perfect-matching* function has a CNF of size $O(n^{3/2})$ but requires exponential size 1-b.p. Jukna (1988).

Gál (1997) has proved that the characteristic function of all blocking sets in finite projective planes has even a CNF with a *linear* number of clauses, but still requires 1-b.p. of exponential size. We describe the function of Gál in more detail, since our first example is its modification.

Let $P \rightleftharpoons \{1, \ldots, n\}$ be the set of points of a projective plane $\mathrm{PG}(2, q)$ of order q, and let $L_1, \ldots, L_n$ be the lines viewed as subsets of P; hence $n = q^2 + q + 1$. Recall that each line has exactly $q+1$ points, every two lines intersect in exactly one point, and exactly $q+1$ lines meet in one point. A *blocking set* is a set of points which intersects every line. The smallest blocking sets are just the lines. Gál (1997) proved that the characteristic function

$$G(x_1, \ldots, x_n) \rightleftharpoons \bigwedge_{i=1}^{n} \bigvee_{j \in L_i} x_j$$

of all blocking sets has no 1-b.p. of size smaller than $2^{\sqrt{n}}$. It appears that, using known lower bounds on the size of non-trivial blocking sets due to Bruen (1970) and Blokhuis (1994), the argument of Gál (1997) can be easily modified to get another result, namely that $\mathrm{P} \neq \mathrm{NP} \cap \text{co-NP}$ in the context of read-once branching programs.

Blocking sets containing a line are called *trivial*. Bruen (1970) proved that any non-trivial blocking set in a projective plane of order q must have at least $q + \sqrt{q} + 1$ points, and this lower bound is known to be tight when q is a square. For the prime order q, Blokhuis (1994) improved Bruen's bound to $3(q+1)/2$ (which is also optimal). These results motivate the investigation of the following Boolean function:

$$B(x_1, \ldots, x_n) \rightleftharpoons \left(\bigwedge_{i=1}^{n} \bigvee_{j \in L_i} x_j \right) \wedge \neg T^n_{q+k+1}(x_1, \ldots, x_n),$$

where $k = (q+1)/2$ if q is a prime, and $k = \lceil \sqrt{q} \rceil$ otherwise; $T^n_s(x_1, \ldots, x_n)$ is the usual threshold function which outputs 1 if and only if $x_1 + \cdots + x_n \geq s$.

Thus, for any input $a : P \to \{0,1\}$, $f(a) = 1$ if and only if the set $a^{-1}(1)$ is blocking and has at most $q+k$ points. This modified function has the required property:

THEOREM 3.2. *Both B and $\neg B$ have 1-n.b.p. of size $O(n^{5/2})$ whereas any 1-b.p. computing B must have size at least $2^k - 1$.*

PROOF. *Upper bound.* Associate with each of the n lines L_i ($i = 1, \ldots, n$) the following two Boolean functions: $\varphi_i \rightleftharpoons \bigwedge_{j \in L_i} x_j \wedge \neg T_k^{n-q-1}(\{x_j \mid j \notin L_i\})$ and $\psi_i \rightleftharpoons \bigwedge_{j \in L_i} (\neg x_j)$.

By the Bruen-Blokhuis bounds we have that $B(a) = 1$ iff a has at most $q+k$ ones and contains some line L, i.e., $a(i) = 1$ for all points $i \in L$. Thus, B is an OR of n functions $\varphi_1, \ldots, \varphi_n$, each of which has a 1-b.p. of size $O(n^{3/2})$. Hence, B has a 1-n.b.p. of size $O(n^{5/2})$. On the other hand, $\neg B(a) = 1$ if and only if either a has more than $q+k+1$ ones or a avoids some line (or both). Thus, $\neg B$ is also an OR of the threshold function T^n_{q+k+1} and n functions $\psi_1, \ldots, \psi_n$, each of which has a 1-b.p of size $O(nq)$. Hence, $\neg B$ has a 1-n.b.p. of size $O(n^{3/2})$.

Lower bound. To this end, we use an argument similar to that of Gál (1997). By Lemma 3.1, it is enough to verify that the function B is k-mixed. To show this, let $I \subseteq P$, $|I| = k$, and $a, b : I \to \{0,1\}$ be two different assignments. Take an $i \in I$ where a and b differ: W.l.o.g. we may assume that $a(i) = 1$ and $b(i) = 0$. There are $q+1$ lines containing the point i. Since $|I - \{i\}| = k - 1 \leq q - 1$ (the number of lines containing i, minus two) we can find among them two lines L_1 and L_2 such that $L_1 \cap I = L_2 \cap I = \{i\}$. Define the assignment $c : P \setminus I \to \{0,1\}$ by letting $c(j) = 1$ if and only if $j \in L_1$. Then the inputs (a,c) and (b,c) both have at most $|I \cup L_1| \leq q+k$ ones. Thus, $f(a,c) = 1$, since (a,c) contains the line L_1, which in turn intersects all other lines. On the other hand, $f(b,c) = 0$ since (b,c) does not intersect the line L_2 in view of $L_2 \cap (I \cup L_1) = \{i\}$ and $b(i) = 0$. □

3.2. 1-b.p. versus 1-n.b.p. and AC^0. In this subsection, we describe two separating functions in AC^0.

THEOREM 3.3. *There is an explicit Boolean function in Σ_3^p such that both f and $\neg f$ have a 1-n.b.p. of polynomial size, whereas the 1-b.p. size of f is* $\exp\left(\Omega(n/\log n)^{1/2}\right)$.

PROOF. Let $k = \lfloor \log_2 n \rfloor$ and choose arbitrarily n different subsets $\alpha(1), \alpha(2), \ldots, \alpha(n)$ of $\{1, 2, \ldots, 2k\}$ such that $|\alpha(i)| = k$ for all $i \in \{1, \ldots, n\}$.

Split the n variables into $2k$ blocks $X_1, X_2, \ldots, X_{2k}$ of size $\lfloor n/(2k) \rfloor$. For every $j \in \{1, 2, \ldots, 2k\}$ we define a function $g_j(x)$ depending only on variables in X_j. For this purpose, we further split the variables in X_j into s subblocks of size s, where $s = \lfloor (n/2k)^{1/2} \rfloor$, and let g_j be the AND of ORs of variables in these subblocks. Finally, let

$$f(x) \rightleftharpoons \bigvee_{i=1}^{n} \left(x_i \wedge \bigwedge_{j \in \alpha(i)} g_j(x) \right).$$

Obviously, f is in Σ_3^p.

To compute f and $\neg f$ by polynomial size 1-n.b.p., we employ the fact that there are only logarithmically many functions $g_1, \ldots, g_{2k}$ and we can search exhaustively through all possible 2^{2k} outputs of this set. Formally, we re-write f in the form

$$f(x) = \bigvee_{\substack{\epsilon_1, \ldots, \epsilon_{2k} \\ \epsilon_j \in \{0,1\}}} \left(\bigvee_{i \in I(\vec{\epsilon})} x_i \;\wedge\; \bigwedge_{j=1}^{2k} (g_j(x) = \epsilon_j) \right), \tag{3.3}$$

where $I(\vec{\epsilon}) \rightleftharpoons \{ i \in \{1, \ldots, n\} \mid \forall j \in \alpha(i)(\epsilon_j = 1) \}$. The expansion for $\neg f$ is obtained by replacing $\bigvee_{i \in I(\vec{\epsilon})} x_i$ with its negation. Now, $x_i \wedge \bigwedge_{j=1}^{2k}(g_j(x) = \epsilon_j)$ can be easily computed by a 1-b.p. of polynomial size, and taking OR of these programs over all $\vec{\epsilon} \in \{0,1\}^{2k}$ and $i \in I(\vec{\epsilon})$ we get a poly-size 1-n.b.p. for f. In order to apply this argument to $\neg f$, we only need to compute $\bigwedge_{i \in I(\vec{\epsilon})}(\neg x_i) \wedge \bigwedge_{j=1}^{2k}(g_j(x) = \epsilon_j)$ by a poly-size 1-b.p. But this becomes obvious if we note that we can get rid of the double occurrences of x_i ($i \in I(\vec{\epsilon})$) by substituting in $\bigwedge_{j=1}^{2k}(g_j(x) = \epsilon_j)$ zeros for all these variables.

In order to prove the exponential lower bound on the 1-b.p. size of f we show that f is $(s-1)$-mixed. Assume that a and b are two different partial inputs setting some set I of $s-1$ variables to constants. Let $l \in I$ be an index of a variable satisfying $a_l \neq b_l$. We need to prove that there is an assignment $c : [n] \setminus I \to \{0,1\}$ such that $f(a,c) \neq f(b,c)$. To this end, we will construct in the next paragraph c in such a way that for any $j \in \{1, 2, \ldots, 2k\}$ c reduces g_j to the constant 1 if $j \in \alpha(l)$, and to the constant 0 if $j \notin \alpha(l)$. Then it is clear from (3.3) that $f(a,c) = a_l$, $f(b,c) = b_l$, and thus $f(a,c) \neq f(b,c)$.

Assignment c is constructed by considering each block X_j separately. If $j \in \alpha(l)$, select in each subblock of X_j one variable not in I and set this variable to 1. This guarantees $g_j|_c \equiv 1$. If $j \notin \alpha(l)$, find a subblock of X_j such that all its variables are not in I and set them to zero. This guarantees $g_j|_c \equiv 0$. By repeating this for all j and by setting all remaining variables in an arbitrary way, we obtain the required c.

Thus, the function f satisfies all requirements from the theorem. □

Slightly modifying this construction, we get a separating function in $\Sigma_3^p \cap \Pi_3^p$.

THEOREM 3.4. *There is an explicit Boolean function in* $\Sigma_3^p \cap \Pi_3^p$ *such that both* f *and* $\neg f$ *have a 1-n.b.p. of polylogarithmic* (!) *size whereas the 1-b.p. size of* f *is*

$$\exp\left(\Omega((\log n/\log\log n)^2)\right).$$

PROOF. We will actually exhibit a Boolean function in m variables such that both f and $\neg f$ have polynomial (in m) size 1-n.b.p. and $\exp\left(O(m\log m)^{1/2}\right)$-sized Σ_3-circuits, whereas every 1-b.p. for f must have size $\exp\left(\Omega(m/\log m)\right)$. This function is transformed into the required form by a standard padding argument: simply introduce $2^{(m\log m)^{1/2}}$ extra dummy variables.

Let $k = \lceil \log_2 m \rceil$, and split the m variables into k blocks $X_1, \ldots, X_k$ of size $\lfloor m/k \rfloor$ each. As in the previous proof, we split the variables of X_j into s subblocks of size s, where s is the even number in $\{\lfloor (m/k)^{1/2} \rfloor, \lfloor (m/k)^{1/2} \rfloor - 1\}$, and let g_j this time be the MAJORITY of MAJORITYs of variables in these subblocks.

f is defined by the following expansion that is analogous to (3.3):

$$f(x) \rightleftharpoons \bigvee_{\substack{\epsilon_1,\ldots,\epsilon_k \\ \epsilon_j \in \{0,1\}}} \left(x_{\overline{\epsilon_1 \ldots \epsilon_k}} \ \wedge \ \bigwedge_{j=1}^{k} (g_j(x) = \epsilon_j) \right), \tag{3.4}$$

where $\overline{\epsilon_1 \ldots \epsilon_k}$ is the integer with the binary representation $\epsilon_1 \ldots \epsilon_k$ (cf. Jukna (1988), Savický & Žák (1996)).

The upper bound $O(m^3/\log m)$ on the size of 1-n.b.p. for f and $\neg f$ is clear (cf. the proof of the previous theorem).

For the lower bound on the size of 1-b.p., note that essentially the same argument as in the proof of Theorem 3.3 shows that f is $\left(\frac{s^2}{4} - 1\right)$-mixed. Indeed, if $|I| < s^2/4$ then for every particular j, there are at least $(s/2)+1$ subblocks of X_j, each of them having at least $(s/2)+1$ variables not in I. Setting all free variables in these subblocks by c to either 0 or 1 forces the value of g_j to the same constant.

Finally, MAJORITY of s variables has a trivial $\exp(O(s))$-sized Σ_2-circuit and Π_2-circuit. Merging these two, we get an $\exp(O(s))$-sized Σ_3-circuit and a Π_3-circuit for $g_j(x)$. Then we can get rid of the fan-in k AND gates in (3.4) at the cost of another factor k in the exponent. This results in Σ_3-circuits for both f and $\neg f$ of size $\exp(O(ks)) = \exp\left(O(m\log m)^{1/2}\right)$. □

Remark 3.5. *Note that in all our examples the nondeterministic 1-b.p. has a very special form: it is a disjunction of polynomially many small 1-b.p. Moreover, in the last example the nondeterminism is further restricted to so-called unique nondeterminism, i.e., at most one computational path may lead to a 1-leaf.*

4. Open Problems

We conclude with several open questions. By Theorems 2.3 and 2.4, the functions F_h and G_h defined in Section 2 do not have a decision tree of size polynomial in $\|F_h\|$ and $\|G_h\|$. On the other hand, both have an oblivious 1-b.p. of size $O(n^2)$.

Open Question 4.1. *Does there exist a Boolean function f which requires 1-b.p. of size super-polynomial in $\|f\|$?*

In the terminology of Section 3, this is equivalent to the question if $\Sigma_2^p \cap \Pi_2^p$ contains a separating function. There are several more interesting questions of the same flavour aimed at bridging the gap between our lower and upper bounds. Does Σ_2^p have a separating function? Does Σ_2^p or $\Sigma_3^p \cap \Pi_3^p$ contain a function separating NP∩co-NP from quasipolynomial time $\tilde{\mathrm{P}}$ in the context of 1-b.p.? Finally, can one get rid of the peculiar $\log n$ factor in the exponent in Theorem 2.1?

Open Question 4.2. *What can be said about the* probabilistic *complexity of the separating functions considered in Section 3?*

Optimistically, what do they separate: NP∩co-NP from BPP or NP∩co-NP ∩ BPP(∩AC^0) from P? A partial step in that direction was made by Ablayev (1997) where he proves that a Boolean function $f_\phi(\vec{x}) = x_{\phi(\vec{x})}$ with a particular "pointer" $\phi : \{0,1\}^n \to [n]$, requires *oblivious* two-sided small error randomized 1-b.p. of exponential size. The pointer ϕ was defined in Savický & Žák (1996), its graph has small 1-b.p., and hence, both f_ϕ and $\neg f_\phi$ have small 1-n.b.p. (cf. Section 3.2). Can this bound be extended to *non-oblivious* random 1-b.p.?

More modest but still (combinatorially) interesting is the following question.

Open Question 4.3. *Does Gál's function (the characteristic function of all blocking sets in $PG(2,q)$) require exponential 1-n.b.p.?*

Note that any *minimal* 1-n.b.p. for this function (as well as for any monotone function) is *monotone*, i.e. edges labelled by $x_i = 0$ can lead only to reject sinks. Thus, the question is in fact about the combinatorial structure of blocking sets: take an acyclic directed graph and label its edges by points of $\mathrm{PG}(2, q)$ so that no point appears in an s–t path twice. A graph is *blocking* if every s–t path corresponds to a blocking set and every such set has at least one s–t path. What is the minimal number of nodes in a blocking graph for $\mathrm{PG}(2, q)$?

Acknowledgements

We want to thank Hans Ulrich Simon for giving us the reference to the paper of Ehrenfeucht and Haussler.

References

F. Ablayev. Randomization and nondeterminism are incomparable for polynomial ordered binary decision diagrams. In *Proc. of ICALP '97*, Lecture Notes in Computer Science 1256, Springer, Berlin, 1997, 195–202.

M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic* **24** (1983), 1–48.

A. Blokhuis. On the size of a blocking set in $\mathrm{PG}(2, p)$. *Combinatorica* **14** (1994), 111–114.

M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proc. of 28th IEEE FOCS*, 1987, 118–126.

A. A. Bruen. Baer subplanes and blocking sets. *Bull. Amer. Math. Soc.* **76** (1970), 342-344.

A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation* **82** (1989), 231-246.

M. Furst, J. Saxe and M. Sipser. Parity, circuits and the polynomial time hierarchy. In *Proc. of 22nd IEEE FOCS*, 1981, 260–270.

A. Gál. A simple function that requires exponential size read-once branching programs. *Information Processing Letters* **62** (1997), 13–16.

J. Hartmanis and L. A. Hemachandra. One-way functions, robustness and non-isomorphism of NP-complete classes. *Tech. Rep. DCS TR86-796*, Cornell University, 1987.

J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation* (*Advances in Computing Research, Vol. 5*), JAI Press, 1989, 143–170.

S. Jukna. Entropy of contact circuits and lower bounds on their complexity. *Theor. Comput. Sci.* **57** (1988), 113–129.

E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. In *Proc. of 23rd ACM STOC*, 1991, 455–464.

N. Linial, Y. Mansour and N. Nisan. Constant depth circuits, Fourier transforms and learnability. In *Proc. of 30th IEEE FOCS*, 1989, 574–579.

A. A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *Proc. of FCT'91*, Lecture Notes in Computer Science 529, Springer, Berlin, 1991, 47-60.

M. Saks and A. Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating games. In *Proc. of 27th IEEE FOCS*, 1986, 29–38.

P. Savický and S. Žák. A large lower bound for 1-branching programs, *TR96-036*, ECCC, Trier, 1996.

G. Tardos. Query complexity, or why is it difficult to separate $\mathrm{NP}^A \cap \text{co-}\mathrm{NP}^A$ from P^A by a random oracle A? *Combinatorica* **9** (1989), 385-392.

I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.

A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. of 26th IEEE FOCS*, 1985, 1–10.

S. Jukna
Dept. of Computer Science,
University of Trier,
D-54286 Trier, Germany and
Institute of Mathematics,
LT-2600 Vilnius, Lithuania
Supported by DFG grant
Me 1077/10–1.
jukna@ti.uni-trier.de

A. Razborov
Steklov Mathematical Institute,
Gubkina 8,
117966, Moscow, Russia
Supported by RBRF grant
#96-01-01222.
razborov@genesis.mi.ras.ru

P. Savický
Institute of Computer Science,
Acad. of Sci. of Czech Republic,
Pod vodárenskou věží 2,
182 07 Praha 8, Czech Republic
Supported by grant of GA
the Czech Republic No. 201/95/0976.
savicky@uivt.cas.cz

I. Wegener
Dept. of Computer Science,
University of Dortmund,
D-44221 Dortmund, Germany
Supported by DFG grant We 1066/8.
wegener@ls2.cs.uni-dortmund.de