A $5n - o(n)$ Lower Bound on the Circuit Size over U_2 of a Linear Boolean Function ⋆

Alexander S. Kulikov[1,2], Olga Melanich[1], and Ivan Mihajlin[3]

[1] St. Petersburg Department of Steklov Institute of Mathematics
[2] Algorithmic Biology Laboratory, St. Petersburg Academic University
[3] St. Petersburg State Polytechnical University

Abstract. We give a simple proof of a $5n - o(n)$ lower bound on the circuit size over U_2 of a linear function $f(x) = Ax$ where $A \in \{0,1\}^{\log n \times n}$ (here, U_2 is the set of all Boolean binary functions except for parity and its complement).

1 Introduction

Proving lower bounds on the circuit complexity of explicit Boolean functions is a central problem of theoretical computer science. Despite of many efforts currently we can only prove small linear lower bounds: $3n - o(n)$ for circuits over the full binary basis B_2 [1, 2] and $5n - o(n)$ for the basis $U_2 = B_2 \setminus \{\oplus, \equiv\}$ [3]. These lower bounds are proved for single-output Boolean functions. However, even less is known for multi-output functions. Though intuitively it seems that computing several functions must be harder than computing just one of them no stronger lower bounds are known for multi-output functions. E.g., if instead of one output we consider $o(n)$ outputs then the strongest lower bounds over B_2 and U_2 are still $3n - o(n)$ and $5n - o(n)$, respectively.

In this note, we prove a $5n - o(n)$ lower bound on the circuit size over U_2 of a linear Boolean function $f(x) = Ax$ where all the columns of the matrix $A \in \{0,1\}^{\log n \times n}$ are pairwise different and non-zero. In fact, we prove a wider result:

$$C_{U_2}(Ax \oplus b) \geq 5(n - m)\,,$$

where $A \in \{0,1\}^{m \times n}$ is a matrix consisting of n different columns and $b \in \{0,1\}^m$ is any vector.

The advantage of the proof is that it contains almost no case analysis though is based on the standard gate elimination method. First, we show that an optimal circuit for such a function does not contain out-degree 1 variables. For out-degree 2 variables, we show that either the considered circuit is not optimal or by appropriate substitution at least 5 gates can be eliminated. Finally, if a circuit contains a variable of degree 3 then it is straightforward to eliminate 5 gates.

⋆ Research is partially supported by Russian Foundation for Basic Research (11-01-00760-a and 11-01-12135-ofi-m-2011), RAS Program for Fundamental Research, Grant of the President of Russian Federation (NSh-3229.2012.1), and Computer Science Club scholarship.

2 Definitions

By $B_{n,m}$ we denote the set of all Boolean functions $f\colon \{0,1\}^n \to \{0,1\}^m$. $B_{n,1}$ is denoted just by B_n. For $f \in B_{n,m}$, by f_i we denote the i-th component of f (thus, $f_i \in B_n$).

A circuit over a basis $\Omega \subseteq B_2$ is a DAG where each vertex has in-degree either 0 or 2 and unbounded out-degree. Vertices of in-degree 0 are called input gates and are labeled by variables $x_1, \ldots, x_n$. All other vertices are called gates and are labeled by binary functions from Ω. Some of the gates are also marked as output gates. Edges connecting gates are usually called wires. The out-degree of a variable is the number of gates fed by this variable.

Such a circuit computes in a natural way a function from $B_{n,m}$ where m is the number of output gates. The size of a circuit is its number of gates not including the input gates. The two widely studied bases Ω are B_2 and $U_2 = B_2 \setminus \{\oplus, \equiv\}$.

The 16 binary functions $f(x_1, x_2)$ from B_2 are usually classified as follows:

- two constants: 0, 1;
- four degenerate functions: x_1, $x_1 \oplus 1$, x_2, $x_2 \oplus 1$;
- eight AND-type functions: $(x_1 \oplus a)(x_2 \oplus b) \oplus c$, where $a, b, c \in \{0,1\}$;
- two XOR-type functions: $x_1 \oplus x_2 \oplus a$, where $a \in \{0,1\}$.

It is easy to see that gates labeled by functions from the first two classes can be easily removed from a circuit. Also, circuits over U_2 do not contain gates computing XOR-functions. In the following, we assume without loss of generality that a circuit over U_2 consists of AND-type gates only.

For a function $f \in B_{n,m}$, by $C_\Omega(f)$ we denote the minimal size of a circuit over Ω computing f.

3 Known Lower Bounds

3.1 Single-output Functions

By a counting argument, Shannon [4] showed that the circuit complexity (over both B_2 and U_2) of almost all functions from B_n is $\Theta(2^n/n)$. For explicit functions, the following lower bounds are known.

For the basis B_2, Schnorr [5] proved a $2n - \Theta(1)$ lower bound for a wide class of functions satisfying some natural property. Paul [6] proved a $2n - o(n)$ lower bound for the storage access function and a $2.5n - o(n)$ lower bound for a modification of the storage access function. Stockmeyer [7] proved a $2.5n - \Theta(1)$ lower bound for many symmetric functions. Blum [1] generalized Paul's proof to get a $3n - o(n)$ lower bound. Kojevnikov and Kulikov [8] proved a $7n/3 - \Theta(1)$ lower bound for functions of high degree. Demenkov and Kulikov [2] proved a $3n - o(n)$ lower bound for affine dispersers.

For the basis U_2, Schnorr [5] proved that the circuit complexity of parity is $3n-3$. Zwick [9] proved a $4n-\Theta(1)$ lower bound for certain symmetric functions. Lachish and Raz [10] proved a $4.5n-o(n)$ lower bound for strongly two-dependent functions. Iwama and Morizumi [3] improved the bound to $5n - o(n)$.

3.2 Multi-output Functions

By counting one can show that almost all functions from $B_{n,n}$ have circuit complexity $\Theta(2^n)$.

Several lower bounds on complexity of multi-output functions are discussed in Hiltgen's thesis [11] (section 4.3, "Lower Bounds on the Complexity of Vector Functions").

Lamagne and Savage [12] proved the following result: if, for $f \in B_{n,m}$, all f_i's are different, then

$$C_\Omega(f) \geq \min_{1 \leq i \leq m} C_\Omega(f_i) + m - 1\,.$$

In other words, the result says that if one needs to compute m different functions instead of just one then at least $m-1$ additional gates are needed. Using this simple method one can prove $4n - o(n)$ and $6n - o(n)$ lower bounds on the circuit complexity over B_2 and U_2, respectively, for a function from $B_{n,n}$. E.g., we can take a function $g \in B_n$ with $C_{B_2}(g) \geq 3n - o(n)$ and define $f = (f_1, \ldots, f_n)$ where $f_i(x_1, \ldots, x_n) = g(x_1, \ldots, x_n) \oplus x_i$.

Blum and Seysen [13] proved that $C_{B_2}(\mathrm{AND}, \mathrm{NAND}) = 2n - 2$ and moreover any optimal circuit for (AND, NAND) consists of two independent trees. Red'kin [14] proved that the circuit complexity over B_2 of a binary adder is $2.5n - 3$. The binary adder is a function from $B_{n,n/2+1}$ that outputs the sum of two input $n/2$-bit numbers. Interlando et al. [15] studied the circuit complexity over $\{\oplus\}$ of the same function studied in this note. They proved that $C_{\{\oplus\}}(Ax) = 2n - o(n)$. Chashkin [16] proved that $C_{B_2}(Ax) = 2n - o(n)$ and also constructed a function from $B_{n,n}$ with circuit complexity (over B_2) $3n - o(n)$. Hiltgen [11] studied so-called feebly one-way functions, i.e., permutations from $B_{n,n}$ such that $C(f) < C(f^{-1})$. He constructed examples of such f's with $C_{B_2}(f) = n - \Theta(1)$ and $C_{B_2}(f^{-1}) = 2n - \Theta(1)$.

4 A $5n - o(n)$ Lower Bound

In this section, we consider functions $f \in B_{n,m}$ of the form $f(x) = Ax \oplus b$, where $A \in \{0,1\}^{m \times n}$ is a matrix with n different non-zero columns and $b \in \{0,1\}^m$ is any vector. First, note that after assigning a Boolean value to any variable one gets a function of the same type (the corresponding column of A is eliminated and some bits of b are changed; all the columns of A are still different and non-zero). This allows us to prove a lower bound by induction. Next, we prove a few elementary lemmas for circuits computing such functions.

Recall that in this section we consider only circuits over U_2. One can assume without loss of generality that such a circuit consists of AND-type gates only. The main property of such a gate is the following: if a variable feeds an AND-type gate then one can assign a constant to this variable so that the gate becomes constant.

Lemma 1. *If some input variable x_i of a circuit computing f feeds exactly one gate then this circuit is not optimal.*

Proof. Assume for the sake of contradiction that x_i has out-degree 1.

If all outputs either depend only on x_i or does not depend on x_i at all then x_i does not need to feed any gate. In this case we can feed a constant instead of x_i into the gate that is fed by x_i.

Otherwise at least one of the outputs depends on x_i and at least one other variable. Let G be the other input of the gate that is fed by x_i. Note that G does not depend on x_i. The considered gate computes a function of the form $(x_i \oplus a)(g \oplus b) \oplus c$. Assume that there is an assignment to all variables but x_i under which G is equal to b. Then the considered gate trivializes to the constant c and the resulting circuit does not depend on x_i. This contradicts the fact that even under this substitution the considered output still depends on x_i. This means that under all possible substitutions G is equal to $b \oplus 1$, i.e., G is constant. This, in turn, means that the circuit is not optimal. □

Lemma 2. *Let P and Q be gates fed by a variable x_i and let Q be a direct successor of P. Then one can reconstruct the circuit without increasing its size so that P and Q are not connected by a wire.*

Proof. Let G be the other successor of P. Then Q depends on G and x_i. Note that Q cannot compute a XOR-type function of x_i and G since to compute the XOR of two variables in U_2 one needs three gates. Thus, we can change the binary function computed in Q and put a wire to Q not from P, but directly from G. The transformation is illustrated in Fig. 1. □

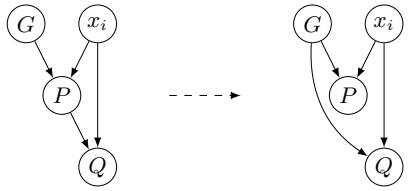


Fig. 1. A transformation described in Lemma 2.

Theorem 1.

$$C_{U_2}(f) \geq 5(n - m)\,.$$

Proof. The proof is by induction on $n - m$. Without loss of generality we may assume that A does not contain all-zero columns.

There are two cases when the statement holds by trivial reasons: $n \leq m$ and $m = 1$ (if $m = 1$ then $n = 1$ since all the columns must be different).

Assume now that $n > m$ and $m > 1$. If there is a row containing exactly one 1 (say, at i-th column), then the corresponding output depends only on x_i. We may then assign $x_i = 0$. This removes the i-th column from the matrix as well as

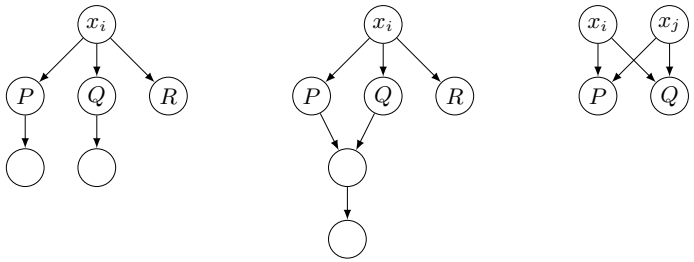
the corresponding row. It is easy to see that the resulting circuit still computes the resulting function. It is also easy to see that all the columns in the resulting matrix are different.

Let now all the rows A contain at least two 1's and consider an optimal circuit computing f. None of the input variables is an output gate. Note also the following property of the function f: even if we assign all variables but x_i there is at least one output that still depends on x_i. Also, by Lemma 1 all variables feed at least two gates.

Consider a top gate P, i.e., a gate fed by two variables x_i and x_j. As the circuit is optimal these variables are different and each of them feeds at least one other gate.

We now consider several cases. In each of the cases we show that it is possible to assign some variable a constant so that at least 5 gates are eliminated. Note that all the gates that become constant are not output gates as all output gates depend on at least two variables. This means that each such gate has at least one successor. Note also that Lemma 2 allows us to assume that if two gates are fed by the same variable then neither of them is fed by the other one.

– **Case 1.** One of x_i and x_j (say, x_i) feeds at least three gates (call them P, Q, R). Then there exists a constant $c \in \{0, 1\}$ such that the substitution $x_i = c$ makes at least two of these gates constant (say, P and Q). Hence, this substitution eliminates P, Q, R and all the successors of P and Q (Fig. 2).



Case 1.1 Case 1.2 Case 2.1

Fig. 2. Cases 1.1, 1.2, and 2.1.

- **Case 1.1.** If the total number of successors of P and Q is at least 2, then at least five gates are eliminated.
- **Case 1.2.** If P and Q have a single successor then it also becomes constant and its successor is also eliminated (and if R happens to be this last successor then R becomes constant and also its successor is eliminated).

– **Case 2.** Both x_i and x_j feed exactly two gates. Denote the other successors of x_i and x_j by R and Q, respectively.

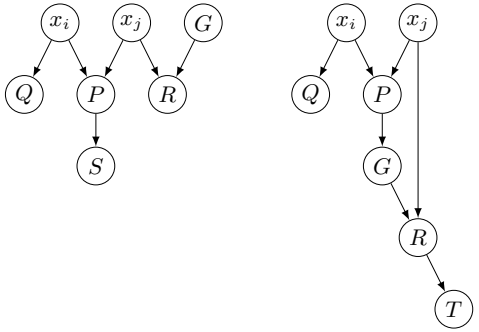
- **Case 2.1.** $Q = R$ (Fig. 2). We show that this case is just impossible. Indeed, since P and Q are AND-type gates,

$$P = (x_i \oplus a_1)(x_j \oplus b_1) \oplus c_1\,,$$
$$Q = (x_i \oplus a_2)(x_j \oplus b_2) \oplus c_2\,,$$

for some constants $a_1, b_1, c_1, a_2, b_2, c_2 \in \{0, 1\}$. Note that $a_1 \neq a_2$. Otherwise by a substitution $x_i = a_1$ we would make the circuit independent of x_j. By exactly the same reason $b_1 \neq b_2$. But then the circuit does not distinguish between the assignments $\{x_i = a_1, x_j = b_1 \oplus 1\}$ and $\{x_i = a_1 \oplus 1, x_j = b_1\}$ (under both these substitutions $P = c_1$ and $Q = c_2$). This cannot be the case since all the columns of A are different and hence there is at least one output that either depends on x_i and does not depend on x_j or vice versa.

- **Case 2.2.** $Q \neq R$. Since the circuit is a DAG the gates can be topologically sorted. Fix some topological order and assume that R precedes Q in it. This, in particular, means that R does not depend on Q.
We would like to show that there is a constant $c \in \{0, 1\}$ such that the substitution $x_j = c$ makes both gates P and R constant. Let G be the other input of R (Fig. 3). As P and R are AND-type gates there exist constants $a_1, b_1, c_1, a_2, b_2, c_2 \in \{0, 1\}$ such that

$$P = (x_j \oplus a_1)(x_i \oplus b_1) \oplus c_1\,,$$
$$R = (x_j \oplus a_2)(g \oplus b_2) \oplus c_2\,.$$



Case 2.2.1 Case 2.2.2

Fig. 3. All subcases of case 2.2.

Note that under the substitution $x_i = b_1$ the gate P trivializes to the constant c_1. Then if for some substitution to all other variables except

for x_j the gate G has value b_2, then the gate R also trivializes to the constant c_2 and the circuit becomes independent of the variable x_j while this cannot be the case. Thus, $x_i = b_1$ implies $G = b_2 \oplus 1$. A crucial observation is that then $x_j = a_1$ also implies $G = b_2 \oplus 1$. Indeed there is no path in the circuit from Q to G (by the assumption that R precedes Q), hence G can only depend on x_i through P. And we know that if $P = c_1$ then $G = b_2 \oplus 1$.

Altogether, if we assign $x_j = a_1$ then P and G trivialize to constant, R also becomes a constant as it is now fed by two constants. All the successors of P, G, R are also eliminated. Also, in the resulting circuit x_i has out-degree 1 hence by Lemma 1 at least one additional gate can be eliminated. To show that in all possible cases at least 5 gates are eliminated we consider two subcases depending on the successors of P (the two subcases are shown at Fig. 3). Denote by S some successor of P. Note that Lemma 2 guarantees that $S \neq Q$ and $S \neq R$.

* **Case 2.2.1.** $S \neq G$. Then the substitution $x_j = a_1$ kills the gates P, R, G, S. After that at least one additional gate can be eliminated due to Lemma 1.
* **Case 2.2.2.** $S = G$. Note that if any of P, G, R has out-degree more than 1 we again remove at leats 5 gates. Assume now that all of them have out-degree exactly 1. Denote the only successor of R by T and consider the other input of T. It is not a constant and it does not depend on x_j. Thus, by an appropriate substitution to all variables but x_j we can trivialize the gate T and make the circuit independent of x_j, a contradiction.

□

References

1. Blum, N.: A Boolean function requiring $3n$ network size. Theoretical Computer Science **28** (1984) 337–345
2. Demenkov, E., Kulikov, A.S.: An elementary proof of a $3n - o(n)$ lower bound on the circuit complexity of affine dispersers. In: Proceedings of 36th International Symposium on Mathematical Foundations of Computer Science (MFCS). Volume 6907 of Lecture Notes in Computer Science., Springer (2011) 256–265
3. Iwama, K., Morizumi, H.: An explicit lower bound of $5n - o(n)$ for boolean circuits. In: Proceedings of International Symposium on Mathematical Foundations of Computer Science (MFCS). Volume 2420 of Lecture Notes in Computer Science., Springer (2002) 353–364
4. Shannon, C.E.: The synthesis of two-terminal switching circuits. Bell System Technical Journal **28** (1949) 59–98
5. Schnorr, C.P.: Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen. Computing **13** (1974) 155–171
6. Paul, W.J.: A $2.5n$-lower bound on the combinational complexity of Boolean functions. SIAM Journal of Computing **6**(3) (1977) 427–433
7. Stockmeyer, L.J.: On the combinational complexity of certain symmetric Boolean functions. Mathematical Systems Theory **10** (1977) 323–336

8. Kojevnikov, A., Kulikov, A.S.: Circuit Complexity and Multiplicative Complexity of Boolean Functions. In: Proceedings of Computability in Europe (CiE). Volume 6158 of Lecture Notes in Computer Science., Springer (2010) 239–245
9. Zwick, U.: A $4n$ lower bound on the combinational complexity of certain symmetric boolean functions over the basis of unate dyadic Boolean functions. SIAM Journal on Computing **20** (1991) 499–505
10. Lachish, O., Raz, R.: Explicit lower bound of $4.5n - o(n)$ for boolean circuits. In: Proceedings of the Annual Symposium on Theory of Computing (STOC). (2001) 399–408
11. Hiltgen, A.P.L.: Cryptographically Relevant Contributions to Combinational Complexity Theory. ETH Series in Information Processing **3** (1994)
12. Lamagna, E.A., Savage, J.E.: On the logical complexity of symmeric switching functions in monotone and and complete bases. Technical report, Brown University (1973)
13. Blum, N., Seysen, M.: Characterization of all optimal networks for a simultaneous computation of AND and NOR. Acta Informatica **21**(2) (1984) 171–181
14. Red'kin, N.P.: Minimal realization of a binary adder. Problemy kibernetiki **38** (1981) 181–216 In Russian.
15. Interlando, J.C., Byrne, E., Rosenthal, J.: The Gate Complexity of Syndrome Decoding of Hamming Codes. Applications of Computer Algebra (ACA) (2004) 1–5
16. Chashkin, A.V.: On complexity of Boolean matrices, graphs and corresponding Boolean matrices. Diskretnaya matematika **6**(2) (1994) 43–73 In Russian.