Theoretical Computer Science 408 (2008) 208–212

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Reductions for monotone Boolean circuits

Kazuo Iwama [a,*], Hiroki Morizumi [a,1], Jun Tarui [b]

[a] Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

[b] Department of Information and Communication Engineering, University of Electro-Communications, Chofu, Tokyo 182-8585, Japan

ARTICLE INFO

Keywords:
Circuit complexity
Monotone circuit
Negation-limited circuit
Majority function
Merging function

ABSTRACT

The large class, say *NLOG*, of Boolean functions, including 0-1 Sort and 0-1 Merge, have an upper bound of $O(n \log n)$ for their monotone circuit size, i.e., they have circuits with $O(n \log n)$ AND/OR gates of fan-in two. Suppose that we can use, besides such normal AND/OR gates, any number of more powerful "F-gates" which realize a monotone Boolean function F with $r(\geq 2)$ inputs and $r'(\geq 1)$ outputs. Note that the cost of each AND/OR gate is one and we assume that the cost of each F-gate is r. Now we define: A Boolean function f in NLOG is said to be F-Easy if f can be constructed by a circuit with AND/OR/F gates whose total cost is $o(n \log n)$. In this paper we show that 0-1 Merge is not F-Easy for an arbitrary monotone function F such that $r' \leq r/\log r$.

1. Introduction

Suppose that we wish to construct a Boolean monotone circuit for 0-1 Merge by using ordinary AND and OR gates of fan-in two and (any number of) Majority gates of any number of inputs at any places. It is well known that, if we are allowed to use only AND/OR gates, then the circuit size must be $\Theta(n \log n)$. Here a size means the number of gates and each AND/OR gate has a unit cost. When we use a Majority gate of r inputs, we assume that such a gate has a cost of r (the reason is given later). It should be noted that a Majority gate of r inputs is realized by using $O(r \log r)$ AND/OR gates, and therefore, if its cost is $r \log r$, then Majority gates are obviously useless. Our setting of cost r is thus subtle, and our primary question in this paper is whether such Majority gates are substantially useful, or whether we can construct a circuit of size $o(n \log n)$ for 0-1 Merge by adding Majority gates. The motivation is as follows:

Our knowledge about the traditional (bounded-fan-in AND, OR and Negation gates are allowed) circuit complexity is summarized as follows: (i) If we cannot use Negation, i.e., for monotone circuits, exponential lower bounds are known (e.g., [17,4,8]). (ii) If we can use few Negations, i.e., at most $(1/6) \log \log n$ Negations, then there are also a superpolynomial lower bounds [2]. (iii) If we can use $\lceil \log(n+1) \rceil$ Negations, then all n inputs can be negated by using $O(n \log n)$ gates [5]. In other words, if we can obtain an $\omega(n \log n)$ lower bound for circuits with $\lceil \log(n+1) \rceil$ Negations, the same lower bound applies for general circuits. The best lower bound for this type of circuits, however, is $6n - \log(n+1) - c$ for a two-output function [10] (see below for details). (iv) For general circuits (i.e., without any restriction for the number of Negations), the best lower bound is still $5n - o(n)$ [11,9] in spite of the long history of research.

Looking at this series of facts, the challenging and somewhat realistic goal is to attain nonlinear lower bounds for circuits of type (iii), for example, for the circuit that negates all n inputs as mentioned above (which we call *Inverter*). Towards this goal, this paper proposes an approach based on reduction which is quite popular in many different contexts of complexity theory.

* Corresponding author.
E-mail addresses: iwama@kuis.kyoto-u.ac.jp (K. Iwama), morizumi@kuis.kyoto-u.ac.jp (H. Morizumi), tarui@ice.uec.ac.jp (J. Tarui).
[1] Present Affiliation: Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan.

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.08.009

Our contribution. In [5], Beals, Nishino and Tanaka studied Inverter. Reading this paper carefully it turns out that if we can use only $\log(n+1)$ Negations, then the way of using them is very restricted, i.e., such circuit includes *monotone* subcircuits which compute $\log(n+1)$ different threshold values with respect to the number of input one's, in particular, one of them must be a majority of n. This means, if we can prove some (e.g., nonlinear) lower bound for monotone Majority, then that bound also applies for Inverter with $\log(n+1)$ Negations. Thus, proving lower bounds for Inverter can be reduced to proving (similar) lower bounds for *monotone* circuits, which appears to be much easier.

We further extend this reduction approach. Let NLOG be the class of (possibly multi-output) Boolean functions which have a monotone upper bound of $O(n \log n)$. Then f in NLOG is said to be *Maj-Easy* if f has a circuit C of size $o(n \log n)$ with AND/OR gates of fan-in two and Majority gates of any fan-in. (Recall that the cost of each AND/OR gate is one and the cost of each Majority gate of fan-in r is r.) Now one can easily see that a conventional (without Majority gates) nonlinear lower bound for monotone Majority (and therefore the same lower bound for Inverter, too) is attained if we can find a monotone Boolean function f such that f is Maj-Easy and f has a conventional monotone lower bound of $\Omega(n \log n)$. A Boolean function f is monotone if $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$ whenever $x_i \leq y_i$ for all i.

Note that we already know that several Boolean functions do have a conventional monotone lower bound of $\Omega(n \log n)$, including Merge. So, we are done if we could prove that Merge is Maj-Easy, which is exactly the question raised at the beginning of this paper. Unfortunately, we can show that Merge is *not* Maj-Easy. In fact we can prove a stronger result, Merge is not F-Easy for any monotone Boolean function F of r inputs and up to $r/\log r$ outputs (under the similar definitions, see Section 2 for details). In other words, even if such an F-gate is arbitrarily powerful, it does not help to reduce the complexity of Merge if its cost is r and its output size is at most $r/\log r$.

Thus, we have to seek other candidates. (Sort has also an $\Omega(n \log n)$ lower bound but it obviously does not help since Sort is more powerful than Merge.) One candidate we show in this paper is what we call an approximated Sort that is Maj-Easy (but is not known if it has an $\Omega(n \log n)$ lower bound).

Previous work. Proving a superpolynomial lower bound for AND/OR/Negation circuits that compute a particular NP language implies P≠NP. However, research towards this ultimate goal has not been very successful. Schnorr first gave a nontrivial lower bounds of $2n$ using base B_2 [18]. After a number of improvements, Zwick gave a $4n$ lower bound for base U_2 [24]. Some ten years later, Lachish and Raz succeeded in improving this bound to $4.5n$ by using the new Strongly-Two-Dependent function [11]. Iwama and Morizumi [9] raised this to $5n$ by deeper analysis of [11], which is currently best. All those results are based on the so-called gate-elimination method, which many people believe has a clear limit of power for further improvement.

For monotone circuits, Razborov first proved a superpolynomial lower bound for a circuit computing Clique [17], which was later improved to an exponential lower bound by Andreev [4]. By combining their proof technique and other techniques, Amano and Maruoka obtained a superpolynomial lower bound for circuits which can use at most $(1/6)\log\log n$ Negation gates [2].

As mentioned before, Beals, Nishino and Tanaka studied the least amount of Negations to compute all Boolean functions. Especially they showed, for Inverter, a lower bound of $5n + 3\log(n+1) - c$ and an upper bound of $O(n \log n)$ which is conjectured optimal by them [5]. Their lower bound was improved to $(7+1/3)n + \log(n+1)/3 - c$ in [19] and improved to $8n - \log(n+1) - c$ in [10]. For the case that a function has two outputs, the best lower bound for circuits with $\log(n+1)$ Negations is $6n - \log(n+1) - c$. This lower bound is obtained by applying the lower bound on a circuit with $\log(n+1) - 1$ Negations for Parity in [10] to the lower bound on a circuit with $\log(n+1)$ Negations for (Parity, ¬Parity). Interestingly, if the number of available Negations is less than $\log(n+1)$, even by one, the lower bound jumps. Sung and Tanaka proved an exponential lower bound for $(\log(n+1) - 1)$-Negation circuits [20].

0-1 Sort, 0-1 Merge and Majority are all practically important and have a large literature for their circuit realization. Majority can be constructed by $O(n)$ AND/OR/Negation gates (see e.g., Chap. 3.4 in [23]) and by $O(n \log n)$ AND/OR gates by using the famous sorting network [1]. (Valiant gave a completely different construction based on a probabilistic method [22].) For its monotone lower bound, however, we have only linear ones. In 84, Dunne proved a $3.5n$ lower bound [7], and in 86, Long proved a $4n$ lower bound [14], but we did not have any further progress in the last two decades. By contrast, we have tight lower bounds for Sort and Merge. Lamagna and Savage [13] proved an $\Omega(n \log n)$ lower bound for Sort. Pippenger and Valiant [16] and Lamagna [12] proved independently $\Omega(n \log n)$ lower bound for Merge. Amano, Maruoka and Tarui [3] proved $\Theta(2^a n)$ for Merge in negation-limited circuits with $\log\log n - a$ Negations.

2. *F*-Easiness and nonlinear lower bounds

In this paper, we mainly deal with the class, denoted by *NLOG*, of monotone Boolean functions that can be computed by circuits consisting of $O(n \log n)$ AND/OR gates of fan-in two. One such function is MERGE(n, m), which is a collection of functions that merges two presorted binary sequence $x_1 \leq x_2 \leq \cdots \leq x_n$ and $x_{n+1} \leq x_{n+2} \leq \cdots \leq x_m$ into the sequence $y_1 \leq y_2 \leq \cdots \leq y_m$. In this paper we discuss only MERGE$(n, 2n)$.

Unless otherwise stated, all circuits in this paper are monotone. *A (monotone) circuit C* is a directed acyclic diagram consisting of *gates* and *links* as shown in Fig. 1. Each gate has *input* and *output terminals*. Each link connects an output terminal to an input terminal, where no two links can go to a single input terminal. Gates are associated with different types: An *AND* (similarly for *OR*) *gate* has two input and one output terminals, an *F-gate* has r input and r' output terminals and computes the Boolean function $F : \{0,1\}^r \to \{0,1\}^{r'}$. Different F-gates in the circuit may have different r/r' values

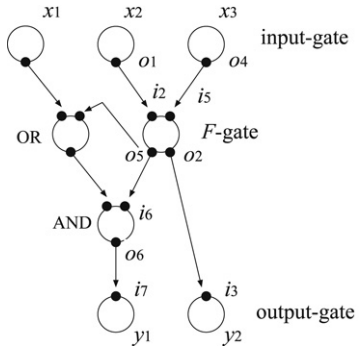

Fig. 1. Circuit.

(i.e., different gate sizes), but must compute the same Boolean function F like Majority, Merge, Sort and so on. An *input gate* has one output and no input terminals and is labeled by a variable in $\{x_1, \ldots, x_n\}$ or a constant 0 or 1, and an *output gate* has one input and no output terminals and is labeled by a variable in $\{y_1, \ldots, y_m\}$.

Each gate has a cost. The cost of AND and OR gate is one, and that of an F-gate is always r regardless of the function F. An input and output gate has cost zero. The *size* of a circuit C is the sum of the costs of all gates in C. The size measure is also used for a Boolean function f: $size_F(f)$ is the minimum size of a circuit with AND/OR/F gates computing f. $size(f)$ is the minimum size of a circuit computing f in which only AND and OR gates can be used (without F-gates). Thus, NLOG can be written as a family of Boolean functions f such that $size(f) = O(n \log n)$. Now we are ready to define F-Easy Boolean functions; intuitively F-Easy functions are those functions for which F-gates are substantially useful:

Definition 1. Let f be in NLOG and F be a monotone function. f is said to be F-Easy if $size_F(f)$ is $o(n \log n)$. In particular, if F is Majority then f is called Maj-Easy.

Now the next Theorem is immediate:

Theorem 1. *Suppose that a function f is Maj-Easy and $size(f)$ is $\Omega(n \log n)$. Then $size(Majority)$ is $\omega(n)$.*

Inverter is a Boolean function from $\{0, 1\}^n$ into $\{0, 1\}^n$ such that $y_i = \neg x_i$ for $1 \leq i \leq n$. A (not monotone) circuit C is said to be *FewNOT* if C uses at most $\log(n + 1)$ Negation gates. $size_{FewNOT}(f)$ is the minimum number of AND/OR/Negation gates that are needed to realize f by a FewNOT circuit.

Theorem 2. *(Implicit in [5]).*

$$size(Majority) \leq size_{FewNOT}(Inverter).$$

Thus, in order to prove a nonlinear lower bound for the FewNOT size of Inverter, it is enough to find a function f in NLOG such that f is Maj-Easy and $size(f) = \Omega(n \log n)$. Although details are omitted, Theorem 2 still holds if Majority is replaced by LogThreshold which is a collection of $\log n$ threshold functions $T^n_{n/2}$, $T^n_{n/4}$(or $T^n_{3n/4}$, controlled by extra input), $T^n_{n/8}$(or $T^n_{3n/8}$, $T^n_{5n/8}$, $T^n_{7n/8}$) and so on (The k-threshold function T^n_k is 1 iff $\Sigma x_i \geq k$.), and therefore Maj-Easy in the above sentence can also be replaced by LogThreshold-Easy. This is the case not only for Inverter but also for many others including (Parity, ¬Parity).

3. Merge is not F-easy

Recall that $\Omega(n \log n)$ lower bounds for $size(f)$ are already known for some functions f in NLOG, in particular for Sort and Merge. Hence, by Theorem 1, our goal would be achieved if we could prove, for example, Merge is Maj-Easy. Unfortunately this is not the case (and neither for Sort since Sort operates exactly as Merge for the presorted inputs). In fact we prove the following stronger result:

Theorem 3. *For any monotone function $F : \{0, 1\}^r \to \{0, 1\}^{r'}$ such that $r' \leq r/\log r$, $MERGE(n, 2n)$ is not F-Easy.*

In the proof of this theorem we don't need the condition that all F-gates compute the same function F. It means that we also prove the theorem for the more general case that F-gates compute different functions.

To prove Theorem 3, we need several new definitions: We define vertex-disjoint paths for both circuits and graphs. For an directed acyclic graph $G = (V, E)$, a *path* is a sequence $v_1 v_2 \cdots v_k$ of vertices such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$. Two paths $v_1 v_2 \cdots v_k$ and $u_1 u_2 \cdots u_l$ are said to be *vertex-disjoint* if $\{v_1, v_2, \ldots, v_k\} \cap \{u_1, u_2, \ldots, u_l\} = \emptyset$. For a circuit C, a path is a sequence of terminals $u_1 v_2 u_2 v_3 \cdots u_k v_{k+1}$ such that $u_1, u_2, \ldots, u_k$ are output terminals, $v_2, v_3, \ldots, v_{k+1}$ are input terminals, v_i and u_i $(2 \leq i \leq k)$ belong to the same gate, and there is an link from u_i to v_{i+1} $(1 \leq i \leq k)$. Two paths are *terminal-disjoint* if their terminals are disjoint. For example, $o_1 i_2 o_2 i_3$ and $o_4 i_5 o_5 i_6 o_6 i_7$ are terminal-disjoint in Fig. 1.

For finite sets X and Y ($|Y| \geq |X|$), let δ be a one-to-one mapping from X into Y. We say that a graph $G = (V, E)$ *implements* a mapping $\delta : X \to Y$ if the following is met: (i) $X \subseteq V$ and $Y \subseteq V$. (ii) $X \cap Y = \emptyset$, and (iii) there are $|X|$ vertex-disjoint

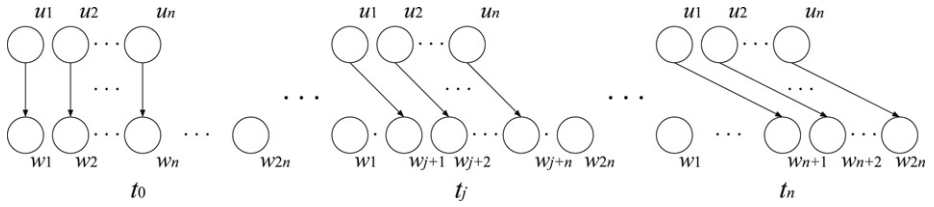

Fig. 2. T_n.

paths from each $x \in X$ to $\delta(x)$. For fixed X and Y, let M be a set of one-to-one mappings from X into Y. Then G *implements* M if G implements every mapping in M. Let $X = \{u_1, \ldots, u_n\}$ and $Y = \{w_1, \ldots, w_{2n}\}$. Then mapping $t_j (0 \le j \le n)$ is defined as $t_j(u_i) = w_{i+j}$. Let $T_n = \{t_0, t_1, \ldots, t_n\}$ (see Fig. 2). Then the following fact is known (see Corollary 2.2.2 in [16]).

Lemma 1. *Suppose that a graph $G = (V, E)$ implements T_n. Then $|E| = \Omega(n \log n)$.*

Now we consider an arbitrary circuit, denoted by $C_{n,2n}$, with AND/OR/F gates which computes Merge$(n, 2n)$. Recall that an F-gate has r input and r' output terminals and different F-gates may have different r/r' values. Note that $C_{n,2n}$ has $2n$ input gates $x_1, \ldots, x_{2n}$ and $2n$ output gates $y_1, \ldots, y_{2n}$, and let mapping t_j; $\{x_1, \ldots, x_n\} \to \{y_1, \ldots, y_{2n}\}$ be defined exactly as before, i.e., $t_j(x_i) = y_{i+j}$.

Lemma 2. *For any $0 \le j \le n$, $C_{n,2n}$ has a set of n terminal-disjoint paths connecting x_i to $t_j(x_i)$ for $1 \le i \le n$.*

Proof. The following argument is similar to the one in [16]. Suppose that $j = 0$. Then we set $x_1 = \cdots = x_n = 0$ and $x_{n+1} = \cdots = x_{2n} = 1$, which forces $y_1 = \cdots = y_n = 0$ and $y_{n+1} = \cdots = y_{2n} = 1$. Now we change the value of x_n from 0 to 1. Then since $C_{n,2n}$ is monotone, there must be at least one path P_0 from x_n to y_n such that the value of all the (input and output) terminals on P_0 changes from 0 to 1 according to this input change. Note that these values of the terminals on P_0 will never change if we keep $x_n = x_{n+1} = \cdots = x_{2n} = 1$.

We next change the value of x_{n-1} from 0 to 1, by which the value of y_{n-1} changes from 0 to 1. Then there must be at least one new path P_1 from x_{n-1} to y_{n-1} such that all the terminal values on P_1 change from 0 to 1 and that P_0 and P_1 are terminal-disjoint. (Otherwise, i.e., if all the new paths intersect with P_0, then the value of y_{n-1} should have been changed to 1 when x_n was changed from 0 to 1 in the previous step.) Similarly, by changing the value of x_{n-2} from 0 to 1, we can create another new path P_2 which does not intersect P_0 or P_1, and so on. Thus, there are n terminal-disjoint paths $P_0, \ldots, P_{n-1}$ connecting x_n to y_n $(= t_0(x_n))$, $\ldots$, x_1 to y_1 $(= t_0(x_1))$, respectively.

For $j = 1$, we can repeat the same argument by initially setting $x_1 = \cdots = x_{n+1} = 0$ and $x_{n+2} = \cdots = x_{2n} = 1$. We can argue similarly for $j = 2, 3, \ldots, n$. □

Lemma 3. *Suppose that every F-gate of r input and r' output terminals in $C_{n,2n}$ satisfies that $r' \le r/\log r$ and that the size of $C_{n,2n}$ is s. Then there exists a graph $G_{n,2n} = (V, E)$ such that $G_{n,2n}$ implements T_n and that $|E| = c \cdot s$ for some constant c.*

Proof. We consider a graph, denoted by $\Pi_{r,r'}$ $(r' \le r)$, such that the graph has disjoint subsets of r *input vertices* X and r' *output vertices* Y and that it implements every one-to-one mapping from any size-r' subset of X to Y. If $r' \le r/\log r$, we can construct $\Pi_{r,r'}$ with $O(r)$ edges as follows.

The construction depends on results in communication networks [15]. Consider a graph that has n input vertices X' and n output vertices Y'. A graph is an *n-superconcentrator* if for all m $(1 \le m \le n)$ the graph implements at least one one-to-one mapping from any size-m subset of X' to any size-m subset of Y'. (Note that nothing is said about which input vertex is connected to which output vertex.) A graph is an *n-connecter* if the graph implements every one-to-one mapping from X' to Y'. $\Pi_{r,r'}$ is obtained from an r-superconcentrator and an r'-connecter by connecting arbitrary r' output vertices of the r-superconcentrator to the input vertices of the r'-connecter. Beneš [6] has given a construction of an n-connecter with $O(n \log n)$ edges and Valiant [21] has given a construction of an n-superconcentrator with $O(n)$ edges. Since we assume that $r' \le r/\log r$, the constructed $\Pi_{r,r'}$ has $O(r + r' \log r') = O(r)$ edges.

Now we construct the graph $G_{n,2n}$ from the circuit $C_{n,2n}$ as follows. Each AND/OR gate of $C_{n,2n}$ is replaced by a vertex of in-degree two. Each F-gate is replaced by $\Pi_{r,r'}$ with $O(r)$ edges. Therefore, the number of edges in the whole resulting graph $G_{n,2n}$ is at most $c \cdot s$ for some constant c since the size of the original $C_{n,2n}$ is s. By Lemma 2, the original $C_{n,2n}$ has n terminal-disjoint paths from x_i to $t_j(x_i)$, which define a one-to-one mapping for each F-gate, from some subset of its input terminals to its output terminals. In the replaced $\Pi_{r,r'}$ there are vertex-disjoint paths for such mapping. Thus, the original terminal-disjoint paths are transformed into n vertex-disjoint paths in $G_{n,2n}$, and thus $G_{n,2n}$ implements T_n. □

Proof of Theorem 3. Suppose for the sake of contradiction that MERGE$(n, 2n)$ is F-Easy. By definition, there is a circuit $C_{n,2n}$ which satisfies the condition of Lemma 3 and whose size is $o(n \log n)$. Then Lemma 3 implies that there is a graph $G = (V, E)$ such that it implements T_n and that $|E| = o(n \log n)$, contradicting to Lemma 1. □

4. Concluding remarks

Thus, we need to seek another function towards our goal. One such candidate is what we call *d-Approximated Sort* defined as follows:

Definition 2. Let $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ be a Boolean function and let m be the number of input one's and m' be output one's. Then f is called d-Approximated Sort if $y_1 \leq y_2 \leq \cdots \leq y_n$ and $|m - m'| \leq n/d$.

$o(\log n)$-Approximated Sort is Maj-Easy for the following reason: (i) We can compute the n-input threshold function of any specific threshold value by using $2n$-input Majority by setting $\{0, 1\}$'s to the extra n inputs appropriately. (ii) Let $d = o(\log n)$. Then d-Approximated Sort can be constructed by using d n-input threshold functions whose d threshold values distribute evenly between 0 and n. At this moment, we do not have any nontrivial lower bounds for its size.

Acknowledgments

The first author would like to thank Mike Paterson for valuable comments. He was also supported in part by Scientific Research Grant, Ministry of Japan, 1609211 and 16300003.

References

[1] M. Ajtai, J. Komlós, E. Szemerédi, An $O(n \log n)$ sorting network, in: Proc. of 15th STOC, 1983, pp. 1–9.
[2] K. Amano, A. Maruoka, A superpolynomial lower bound for a circuit computing the clique function with at most $(1/6) \log \log n$ negation gates, SIAM J. Comput. 35 (1) (2005) 201–216.
[3] K. Amano, A. Maruoka, J. Tarui, On the negation-limited circuit complexity of merging, Discrete Appl. Math. 126 (1) (2003) 3–8.
[4] A.E. Andreev, On a method for obtaining lower bounds for the complexity of individual monotone functions, Sov. Math. Doklady 31 (3) (1985) 530–534.
[5] R. Beals, T. Nishino, K. Tanaka, On the complexity of negation-limited boolean networks, SIAM J. Comput. 27 (5) (1998) 1334–1347.
[6] V.E. Beneš, Optimal rearrangeable multistage connecting networks, Bell Syst. Tech. J. 43 (1964) 1641–1656.
[7] P.E. Dunne, Lower bound on the monotone network complexity of threshold functions, in: Proc. 22nd Ann. Allerton Conf. on Communication, Control and Computing, 1984, pp. 911–920.
[8] D. Harnik, R. Raz, Higher lower bounds on monotone size, in: Proc. of 32nd STOC, 2000, pp. 378–387.
[9] K. Iwama, H. Morizumi, An explicit lower bound of $5n - o(n)$ for Boolean circuits, in: Proc. of 27th MFCS, in: LNCS, vol. 2420, 2002, pp. 353–364.
[10] K. Iwama, H. Morizumi, J. Tarui, Negation-limited complexity of parity and inverters, in: Proc. of 17th ISAAC, in: LNCS, vol. 4288, 2006, pp. 223–232.
[11] O. Lachish, R. Raz, Explicit lower bound of $4.5n - o(n)$ for boolean circuits, in: Proc. of 33rd STOC, 2001, pp. 399–408.
[12] E.A. Lamagna, The complexity of monotone networks for certain bilinear forms, routing problems, sorting, and merging, IEEE Trans. Comput. 28 (10) (1979) 773–782.
[13] E.A. Lamagna, J.E. Savage, Combinational complexity of some monotone functions, in: Proc. 15th Ann. IEEE Symp. on Switching and Automata Theory, 1974, pp. 140–144.
[14] D. Long, The monotone circuit complexity of threshold functions, University of Oxford, 1986, unpublished manuscript.
[15] N. Pippenger, Communication networks, in: J.V. Leeuwen (Ed.), Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, Elsevier/MIT Press, 1990, pp. 805–833.
[16] N. Pippenger, L.G. Valiant, Shifting graphs and their applications, J. ACM 23 (3) (1976) 423–432.
[17] A.A. Razborov, Lower bounds on the monotone complexity of some boolean functions, Sov. Math. Doklady 31 (1985) 354–357.
[18] C. Schnorr, Zwei Lineare Untere Schranken für die Komplexität Boolescher Funktionen, Computing 13 (1974) 155–171.
[19] S. Sung, K. Tanaka, Lower bounds on negation-limited inverters, in: Proc. of 2nd DMTCS: Discrete Mathematics and Theoretical Computer Science Conference, 1999, pp. 360–368.
[20] S. Sung, K. Tanaka, An exponential gap with the removal of one negation gate, Inform. Process. Lett. 82 (3) (2002) 155–157.
[21] L.G. Valiant, Graph-theoretic properties in computational complexity, J. Comput. System Sci. 13 (3) (1976) 278–285.
[22] L.G. Valiant, Short monotone formulae for the majority function, J. Algorithms 5 (3) (1984) 363–366.
[23] I. Wegener, The Complexity of Boolean Functions, in: Wiley-Teubner Series in Computer Science, 1987.
[24] U. Zwick, A $4n$ lower bound on the combinatorial complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions, SIAM J. Comput. 20 (1991) 499–505.