A SUPERPOLYNOMIAL LOWER BOUND FOR A CIRCUIT COMPUTING THE CLIQUE FUNCTION WITH AT MOST $(1/6)\log\log N$ NEGATION GATES*

KAZUYUKI AMANO† AND AKIRA MARUOKA†

Abstract. In this paper, we investigate the lower bound on the number of gates in a Boolean circuit that computes the clique function with a limited number of negation gates. To derive strong lower bounds on the size of such a circuit we develop a new approach by combining three approaches: the restriction applied to constant depth circuits due to Håstad, the approximation method applied to monotone circuits due to Razborov, and the boundary covering developed in the present paper. We prove that if a circuit C with at most $\lfloor(1/6)\log\log m\rfloor$ negation gates detects cliques of size $(\log m)^{3(\log m)^{1/2}}$ in a graph with m vertices, then C contains at least $2^{(1/5)(\log m)^{(\log m)^{1/2}}}$ gates. No non-trivial lower bounds on the size of such circuits were previously known, even if we restrict the number of negation gates to be a constant. Moreover, it follows from a result of Fischer (Lecture Notes in Comput. Sci., 33 (1974), pp. 71–82) that if one can improve the number of negation gates from $\lfloor(1/6)\log\log m\rfloor$ to $\lfloor 2\log m\rfloor$ in the statement, then we have P $\neq$ NP. We also show that the problem of lower bounding the negation-limited circuit complexity can be reduced to the one of lower bounding the maximum of the monotone circuit complexity of the functions in a certain class of monotone functions.

Key words. circuit complexity, monotone circuit, negation-limited circuit, approximation method, lower bound, clique function

AMS subject classifications. Subject 06E30, 68Q17, 68Q25, 94C10

1. Introduction.

There has been substantial progress in obtaining strong lower bounds on the size of restricted Boolean circuits, such as constant depth circuits or monotone circuits, that compute certain functions. For example, exponential lower bounds were derived for the size of constant depth circuits computing the parity function [7] and for the size of monotone circuits computing the clique function and other functions ([1, 2, 5, 8, 9, 10, 13]). It is natural to ask if we could use the approaches developed so far to derive strong lower bounds for a more general model. In such a generalized model, we consider circuits with a limited number of negation gates. In fact, it so far remains open to derive non-trivial lower bounds on the size of a circuit computing a certain monotone function with, say, a constant number of negation gates [14]. Fischer [6] showed that for any function f on n variables, the size of the smallest circuit computing f with an arbitrary number of NOT gates and the

*A preliminary version of this paper appeared in Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS '98) (Lecture Notes in Computer Science, 1450), Brno, Czech Republic, 1998. pp. 399–408.

†Graduate School of Information Sciences, Tohoku University, Aoba 6-6-05, Aramaki, Sendai 980-8579, Japan ({ama|maruoka}@ecei.tohoku.ac.jp)

one with, at most, $\lceil \log(n+1) \rceil$ NOT gates are polynomially related (see also [4]). So if one can prove superpolynomial lower bounds on the size of circuits with at most $\lceil \log(n+1) \rceil$ NOT gates computing an explicit function in NP, then we have P$\neq$NP.

In this paper, we try to obtain superpolynomial lower bounds on the size of circuits computing an explicit function in NP with $O(\log\log n)$ NOT gates rather than $O(\log n)$ NOT gates. More precisely we prove the following: If a circuit C with at most $\lfloor (1/6)\log\log m \rfloor$ NOT gates detects cliques of size $(\log m)^{3(\log m)^{1/2}}$ in a graph with m vertices, then C contains at least $2^{(1/5)(\log m)^{(\log m)^{1/2}}}$ gates (Theorem 5.1). Note that the problem of detecting a clique in a graph with m vertices will be written as a Boolean function of $n = \binom{m}{2}$ variables. We also show that the problem of lower bounding the negation-limited circuit complexity can be reduced to the one of lower bounding the maximum of the monotone circuit complexity of the functions in a certain class of monotone functions. (Theorem 3.2).

To achieve the main results, we develop a new approach by combining three approaches: the restriction applied to constant depth circuits [7], the approximation method applied to monotone circuits [10, 11] and the boundary covering developed in the present paper. A Boolean function f partitions the Boolean cube into two regions, $f^{-1}(0)$ and $f^{-1}(1)$. We can think of the boundary between the two regions, defined as the collection of pairs of vectors (w, w') such that $f(w) \neq f(w')$, and the Hamming distance between w and w' is 1. The idea of the proof of the main theorem is as follows. First, we prove in Section 3 a theorem showing that the problem of proving a lower bound on the size of the negation-limited circuit computing a monotone function f can be reduced to the one of proving a lower bound on the maximum over sizes of monotone circuits such that the union of boundaries of the functions computed by the monotone circuits covers the boundary of the monotone function f (Theorem 3.2). Second, we analyze carefully in Section 4 the proof of Amano and Maruoka [2] that gives an exponential lower bound on the monotone circuit size of the clique function, and prove a statement saying that we still need a superpolynomial number of gates in a monotone circuit that implements even a certain small fraction of the boundary of the clique function (Theorem 4.1). Finally, we prove in Section 5 a statement (Theorem 5.1) that no matter what collection of monotone functions we take to cover the boundary of the clique function, the largest fraction of the boundary covered by a monotone function in the collection is more than what is needed to apply the result (Theorem 4.1) in Section 4. This is the most difficult part of the proof.

2. Preliminaries. For w in $\{0,1\}^n$, let w_i denote the value of the ith bit of w. Let w and w' be in $\{0,1\}^n$. We denote $w \leq w'$ if $w_i \leq w'_i$ for all $1 \leq i \leq n$, and $w < w'$ if $w \leq w'$ and $w \neq w'$. Let $\mathrm{Ham}(w, w')$ denote the Hamming distance between w and w', i.e., $\mathrm{Ham}(w, w') = |\{i \in \{1, \ldots, n\} \mid w_i \neq w'_i\}|$, where $|S|$ denotes the number of

elements in a set S. For two Boolean functions f and g of n variables, we write $f \leq g$ if $f(w) \leq g(w)$ for all $w \in \{0,1\}^n$.

A *Boolean circuit* is a directed acyclic graph with gate nodes (or, simply gates) and input nodes. The operation AND or OR is associated with each gate whose indegree is 2, whereas NOT is associated with each gate whose indegree is 1. A Boolean variable or a constant, namely, 0 or 1, is associated with each input node whose indegree is 0. There is one designated node in a Boolean circuit with outdegree 0, which is called the output gate. Each gate of the circuit computes a Boolean function in the obvious way and the function computed by the circuit is the function computed by the output gate. In particular, a circuit with no NOT gates is called a *monotone circuit*.

A Boolean function of n variables is *monotone* if $f(w) \leq f(w')$ holds for any $w, w' \in \{0,1\}^n$ such that $w \leq w'$. Let $\mathcal{M}^n$ denote the set of all monotone functions on n variables. It is well-known that f is monotone if and only if f can be computed by a monotone circuit. The *size* of a circuit C, denoted size(C), is the number of gates in the circuit C. The *circuit complexity* (respectively, *monotone circuit complexity*) of a function f, denoted size(f) (respectively, size$_{mon}(f)$), is the size of the smallest circuit (respectively, the smallest monotone circuit) computing f. For a function f and a positive integer t, the *circuit complexity with t limited negation* (negation limited complexity, for short) of a function f, denoted size$_t(f)$, is the size of the smallest circuit C that computes f and includes at most t NOT gates. If a function f cannot be computed with only t NOT gates, then size$_t(f)$ is undefined. For a circuit C and a gate g in C, let $C_g(w)$ denote the output of the gate g in the circuit C that has the input w. We sometimes assume that a gate g of circuit C is specified as an output gate of the circuit C. In such a case, we say the circuit computes the function $C_g(w)$, which will simply be denoted by $C(w)$. We say a gate g in a circuit C separates a pair of vectors (w, w') (or simply, a gate g separates (w, w') when no confusion arises) if $C_g(w) = 0$, $C_g(w') = 1$ and $\mathrm{Ham}(w, w') = 1$. In particular, when g is taken to be the output gate in C, we simply say that the circuit C separates such a pair (w, w'). Similarly, when a circuit separates a pair we say the function computed by the circuit separates the pair.

Throughout this paper, the function $\log x$ denotes the logarithm base 2 of x.

3. Relationship between Negation-Limited and Monotone Circuit Complexity. In this section, we explore a relationship between negation-limited circuit complexity and monotone circuit complexity for a monotone Boolean function.

DEFINITION 3.1. *Let f be a Boolean function of n variables. A boundary graph of f, denoted $G(f)$, is defined as follows: $G(f) = (V, E)$ is a directed graph with $V = \{0,1\}^n$ and $E = \{(w, w') \mid Ham(w, w') = 1,\ f(w) = 0$ and $f(w') = 1\}$.*

So a boundary graph of f is a graph whose edge set consists of pairs that are

separated by the function f. Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two graphs on the same set V of vertices. Then the *union* $G_1 \cup G_2$ is defined to be the graph $(V, E_1 \cup E_2)$. Furthermore, we say G_1 contains G_2, denoted by $G_1 \supseteq G_2$, if $E_1 \supseteq E_2$ holds.

THEOREM 3.2. *Let f be a monotone function of n variables. For any positive integer t,*

$$size_t(f) \geq \min_{F'=\{f_1,\ldots,f_\alpha\}\subseteq \mathcal{M}^n} \left\{ \max_{f'\in F'} \{size_{mon}(f')\} \;\middle|\; \bigcup_{f'\in F'} G(f') \supseteq G(f) \right\} ,$$

where $\alpha = 2^{t+1} - 1$. Here the functions f_i are not necessarily distinct, so F' could be a multiset. □

This theorem shows that the problem of deriving lower bounds on the negation-limited circuit complexity of a monotone function can be reduced to the one of deriving the maximum monotone circuit complexity over the monotone functions such that the corresponding boundary graphs of the functions cover that of the original function. Intuitively, this is because any pair of vectors (w, w') separated by a function f, which is supposed to be computed by a negation-limited circuit C, belongs to the boundary of a monotone circuit obtained by restricting outputs of some NOT gates (possibly, all of the NOT gates) in C, to constants 0 or 1 appropriately and throwing away the remaining NOT gates in C. So lines where we place restriction to constants are outputs of some NOT gates inside of the circuit rather than inputs to the entire circuit as in the case of the lower bound proof for constant depth circuits.

It is worthwhile to note that the set of all variables $F' = \{x_1, \ldots, x_n\}$ satisfies the condition $\cup_{f'\in F'} G(f') \supseteq G(f)$ for any monotone function f. Hence, if $\alpha \geq n$, that is, $t \geq \log(n+1) - 1$, the right-hand side of the inequality in Theorem 3.2 does not give a non-trivial lower bound.

Proof of Theorem 3.2. Let f be a monotone function of n variables. Let C be the smallest circuit that computes f using no more than t NOT gates, that is, size(C) =size$_t(f)$. Without loss of generality, we assume the number of NOT gates in C is given by t. Furthermore, without loss of generality we assume that the output gate of C is not a NOT gate. Let $g_1, \ldots, g_t$ be a list of NOT gates of C arranged in the topological ordering and g_{t+1} be the output gate of the entire circuit C. For $0 \leq i \leq t$ and $u = (u_1, \ldots, u_i) \in \{0,1\}^i$, let C_u denote the subcircuit of C obtained by fixing the output of the NOT gates g_j to constant u_j for $1 \leq j \leq i$ and making the input to g_{i+1} in C the output of the circuit C_u. In particular, for the empty sequence λ, C_λ denotes the circuit obtained by making the input to g_1 in C the output of entire circuit C_λ.

Clearly, for any u of length at most t, the function computed by the circuit C_u

is monotone, and the size of the circuit C_u is not greater than the size of the circuit C. Then it is easy to see that, for any $(w, w') \in \{0,1\}^n \times \{0,1\}^n$ separated by the circuit C, there exists $0 \leq i \leq t$ and $u \in \{0,1\}^i$ such that the circuit C_u separates the (w, w'). This is because as such an i we can simply take i such that g_{i+1} is the first gate in the sequence $(g_1, \ldots, g_{t+1})$ such that $C_{g_{i+1}}(w) \neq C_{g_{i+1}}(w')$, and put $u_j = g_j(w)(= g_j(w'))$ for $1 \leq j \leq i$. The number of the circuits represented as C_u for $u \in \{0,1\}^*$ such that $|u| \leq t$ is given by $\sum_{j=0}^{t} 2^j = 2^{t+1} - 1 = \alpha$. Hence, denoting by f_j's functions computed by circuit C_u's, we have $\bigcup_{f' \in F'} G(f') \supseteq G(f)$ for $F' = \{f_1, \ldots, f_\alpha\}$. Thus, since $\mathrm{size}_t(f)(= \mathrm{size}(C)) \geq \mathrm{size}_{mon}(f')$ for any $f' \in F'$, the proof is completed. □

4. Hardness of Approximating Clique Function. The clique function, denoted CLIQUE(m, s), of $m(m-1)/2$ variables is defined to take the value 1 if and only if the undirected graph on m vertices represented in the obvious way by the input contains a clique of size s. For a positive integer s_2, a graph on m vertices is called *good* if it consists of a clique on some set of s_2 vertices, and contains no other edges. Let $I(m, s_2)$ denote the set of such good graphs. For a positive integer s_1, a graph on m vertices is called *bad* if there is a partition of the vertices into $m \bmod (s_1 - 1)$ sets of size $\lceil m/(s_1-1) \rceil$ and $s_1 - 1 - (m \bmod (s_1 - 1))$ sets of size $\lfloor m/(s_1-1) \rfloor$ such that any two vertices chosen from different sets have an edge between them, and no other edges exist. Let $O(m, s_1)$ denote the set of such bad graphs.

For $1 \leq s_1 \leq s_2 \leq m$, let $F(m, s_1, s_2)$ denote the set of all the monotone functions f of $\binom{m}{2}$ variables representing a graph G on m vertices such that the function f outputs 0 if G contains no clique of size s_1, outputs 1 if G contains a clique of size s_2, and outputs an arbitrary value otherwise. We remark that $F(m, s, s) = \{\mathrm{CLIQUE}(m, s)\}$ and that if $s_1 < s_2$ then $F(m, s_1, s_2)$ consists of more than one function. For any function f in $F(m, s_1, s_2)$, the value of f is 1 for any good graph, and is 0 for any bad graph. Moreover a good graph is minimal in the sense that removing any edge from the graph destroys the clique of size s_2. On the other hand, a bad graph is maximal in the sense that adding any edge to the graph makes the graph contain a clique of size s_1.

In this section, we prove the following theorem, which will be needed in the next section to prove the main theorem. This theorem says that, if $64 \leq s_1 \leq s_2$ and $s_1^{1/3} s_2 \leq m/200$, then no function in $F(m, s_1, s_2)$ can be approximated by a feasible monotone circuit.

THEOREM 4.1. *Let s_1 and s_2 be positive integers such that $64 \leq s_1 \leq s_2$ and $s_1^{1/3} s_2 \leq m/200$. Suppose that C is a monotone circuit and that the fraction of good graphs in $I(m, s_2)$ such that C outputs 1 is at least $h = h(s_2)$. Then at least one of the following holds:*

(i) The number of gates in C is at least $(h/2)2^{s_1^{1/3}/4}$,

(ii) The fraction of bad graphs in $O(m, s_1)$ such that C outputs 0 is at most $2/s_1^{1/3}$.

□

Let $\Pr_{v \in I(m,s_2)}[E(v)]$ denote the probability of event $E(v)$ provided that the uniform distribution over $I(m, s_2)$ is assumed, and similarly for $\Pr_{u \in O(m,s_1)}[E(u)]$. Theorem 4.1 is restated as follows: Assume that the conditions on the parameters described in the theorem are satisfied. If monotone circuit C is such that

$$\Pr_{v \in I(m,s_2)}[C(v) = 1] \geq h,$$

and

$$\Pr_{u \in O(m,s_1)}[C(u) = 0] > \frac{2}{s_1^{1/3}}$$

then

$$\text{size}(C) \geq \left(\frac{h}{2}\right) 2^{s_1^{1/3}/4}.$$

The proof of Theorem 4.1 is done employing the symmetric version of the method of approximation [2, 5, 8, 9, 13]. In particular, we use arguments similar to the proof of an exponential lower bound on the monotone circuit complexity of the clique function (Theorem 3.1 in [2]). The key to the proof is to define the approximate operations $\overline{\vee}$ (which approximates an OR gate) and $\overline{\wedge}$ (which approximates an AND gate) in terms of DNF and CNF formulas such that the size of terms and clauses in the formulas is limited appropriately. For the purpose of the arguments of the current paper, we adopt the same definition for the approximate operations as in [2] except for the values of the parameters l and r in their definitions, and follow their arguments to obtain Theorem 4.1.

In what follows we present a rough sketch of a proof of Theorem 4.1. As in [2], a monotone circuit is assumed to be converted to satisfy the following conditions: Any input of an OR gate (respectively, an AND gate) is connected to either an output of AND gate (respectively, an OR gate) or an input node; and the output gate of the circuit is an AND gate. It is easy to see that in order to convert a monotone circuit to satisfy these conditions, we need to at most double the size of the circuit.

A DNF formula is a disjunction of conjunctions of input variables and each conjunction of a DNF formula is called a *term*. A CNF formula is a conjunction of disjunctions of input variables and each disjunction of a CNF formula is called a *clause*. Let t be a term or a clause. The *endpoint set* of t is a set of all endpoints of the edges corresponding to variables in t. The *size* of t is defined to be the cardinality of the endpoint set of t. The approximate operations $\overline{\vee}$ and $\overline{\wedge}$ are defined as follows:

$\overline{\vee}$: Let f_1^D and f_2^D be two functions, represented by monotone DNF formulas, feeding into an $\overline{\vee}$ gate. $f_1^D \,\overline{\vee}\, f_2^D$ is the CNF formula obtained by transforming the monotone DNF formula $f_1^D \vee f_2^D$ into the monotone CNF formula and then taking away all the clauses whose size exceeds r.

$\overline{\wedge}$: Let f_1^C and f_2^C be two functions, represented by monotone CNF formulas, feeding into an $\overline{\wedge}$ gate. $f_1^C \,\overline{\wedge}\, f_2^C$ is the DNF formula obtained by transforming the monotone CNF formula $f_1^C \wedge f_2^C$ into the monotone DNF formula and then taking away all the terms whose size exceeds l.

In what follows, AND and OR gates are also written as $\wedge$ and $\vee$ gates, respectively. Given a monotone circuit C, the circuit obtained by replacing all $\vee$ and $\wedge$ gates in C by $\overline{\vee}$ and $\overline{\wedge}$ gates, respectively, is denoted by $\overline{C}$, which will be called the approximator circuit corresponding to C. The approximate operations and the approximator circuits are introduced to derive good lower bounds on the size of circuits computing a certain function. Theorem 4.1 derives a lower bound on the size of monotone circuits approximately computing the clique function. Its proof, based on the approximation method, is as follows. First, we show that the number of good and bad graphs that are classified incorrectly by an approximator circuit is large (Lemma 4.3). More precisely, for an approximator circuit $\overline{C}$ arbitrarily given, $\overline{C}$ outputs 0 for a large number of good graphs or yields 1 for a large number of bad graphs. Second, we show that the number of good and bad graphs for which a usual gate and the corresponding approximate gate behave differently is small. More precisely, it is only for a small number of bad graphs that $\vee$ gate outputs 0 and $\overline{\vee}$ gate yields 1 (Lemma 4.4), whereas it is only for a small number of good graphs that $\wedge$ gate outputs 1 and $\overline{\wedge}$ gate yields 0 (Lemma 4.5). Recall that, in general, the usual gates and the corresponding approximate gates behave differently because long clauses or long terms are taken away when defining the approximate operations based on the formulas. Finally, since for each good or bad graph classified wrongly by an approximator circuit there exist an approximate gate in $\overline{C}$ that behaves differently from the corresponding usual gate on that graph, the number of approximate gates that compensate for a large number of good or bad graphs misclassified by the entire circuit must be large (Theorem 4.1).

Choose $l = \lfloor s_1^{1/3}/4 \rfloor$ and $r = \lfloor 30 s_1^{1/3} \rfloor$. Put $w = m \bmod (s_1 - 1)$. A simple calculation shows the following.

FACT 4.2. *$|I(m, s_2)| = (m!)/(s_2!(m - s_2)!)$ and*

$$|O(m, s_1)| = \frac{m!}{(\lceil m/(s_1-1) \rceil !)^w (\lfloor m/(s_1-1) \rfloor !)^{s_1-1-w} w!(s_1-1-w)!}.$$

We proceed to the technical parts of the proof. Although the arguments are analogous to that presented in [2], we give proofs here to make this paper self-contained.

LEMMA 4.3. *Let C be a monotone circuit. An approximator circuit $\overline{C}$ outputs identically* 0 *or the fraction of bad graphs in $O(m, s_1)$ such that $\overline{C}$ outputs* 1 *is at least* $1 - s_1^{-1/3}$.

Proof. Let $\overline{f}$ be the output of an approximator circuit $\overline{C}$. Because of the assumption that the output gate of the approximator circuit is an AND gate, $\overline{f}$ can be represented by a monotone DNF formula consisting of terms of size at most l. If $\overline{f}$ is identically 0 then the first conclusion holds. If not, then there is a term t whose size is at most l such that $\overline{f} \geq t$ holds. In what follows, bad graphs are represented as one to one mapping from the vertex set to $\{(1,1), \ldots, (1, \lceil m/(s_1-1) \rceil), \ldots, (s_1 - 1, 1), \ldots, (s_1 - 1, \lfloor m/(s_1-1) \rfloor)\}$, so there are many mappings corresponding to one bad graph. Such a mapping specifies a bad graph in the obvious way: Two vertices in the graph have an edge between them if and only if the mapping assigns pairs to the vertices with different first components. The function in question will be estimated in terms of the ratio of the corresponding mappings. It is easy to see that the ratio of mappings that satisfy the condition that there is a variable x in the term t such that the two vertices incident to x are assigned a pair with the same first component, i.e., the term t outputs 0 on the bad graphs specified by such mappings, is at most

$$\frac{l(l-1)}{2} \frac{\lceil m/(s_1-1) \rceil}{m} < \frac{s_1^{2/3}}{32} \frac{2}{s_1} < s_1^{-1/3}.$$

This completes the proof. □

LEMMA 4.4. *Suppose $\vee$ gate and $\overline{\vee}$ gate are given as inputs the same monotone formulas such that the size of any term in the formulas is at most l. The number of bad graphs in $O(m, s_1)$ for which the OR and $\overline{\vee}$ gates produce different outputs (the OR gate produces 0, whereas the $\overline{\vee}$ gate produces 1) is at most*

$$\frac{(m/s_1^{1/6})^{r+1}(m-r-1)!}{(\lceil m/(s_1-1) \rceil !)^w (\lfloor m/(s_1-1) \rfloor !)^{s_1-1-w} w!(s_1-1-w)!}, \tag{4.1}$$

where $w = m \bmod (s_1 - 1)$.

Proof. Let f_1^D and f_2^D denote monotone formulas, such that the size of any term in the formulas is at most l. Let $f_1^D \vee f_2^D$ and $f_1^D \overline{\vee} f_2^D$ be denoted by f^D and f^C, respectively. Let $t_1, \ldots, t_q$ be the complete list of the terms in f^D. We shall count the number of bad graphs u such that both $f^D(u) = 0$ and $f^C(u) = 1$ hold. As in the proof of Lemma 4.3, bad graphs are represented as the mappings described there. Since each bad graph in $O(m, s_1)$ is represented exactly as $(\lceil m/(s_1-1) \rceil !)^w (\lfloor m/(s_1-1) \rfloor !)^{s_1-1-w} w!(s_1-1-w)!$ mappings as in the proof of Lemma 4.3, it suffices to show that the number of mappings corresponding to the bad graphs described in the lemma is at most $(m/s_1^{1/6})^{r+1}(m-r-1)!$. In order to count such mappings we will count the number of ways of choosing one or two vertices in a certain manner repeatedly

from the variables in $t_1, \ldots, t_q$ so that the corresponding bad graph u satisfies the conditions $f^D(u) = 0$ and $f^C(u) = 1$.

Suppose that we somehow already assigned distinct pairs of integers to endpoints of variables from terms $t_1, \ldots, t_{i-1}$ so as to make all these terms take the value 0 and that we proceed to the term t_i. We now typically choose one or two vertices from the endpoints of variables from t_i in the way described below and then proceed to the next term t_{i+1}.

We first consider two extreme cases. If there is a variable in t_i already assigned 0 by the partial assignment, we skip to the next term t_{i+1}. The other extreme case occurs when all the variables in t_i are, so far, assigned 1. In this case the term t_i will never take value 0, hence we do not need to consider the case.

If neither of these extreme cases happens, choose a variable from the term t_i such that at least one of the vertices associated with the variable is not assigned a pair of integers. There are two cases to consider: If exactly one of the vertices has been assigned, then assign a pair to the remaining vertex whose first component is identical to the first component of the pair of integers associated with the variable so that the variable takes the value 0. In this case, there are at most $\lfloor m/(s_1 - 1) \rfloor \leq m/(s_1 - 1)$ ways of assigning the pairs of integers to the vertex. On the other hand, if both of the vertices have not been chosen, assign to these vertices pairs of integers with their first components being the same so that the variable associated with two vertices takes the value 0. So, for the two vertices, there are at most $(s_1 - 1)(\lceil m/(s_1 - 1) \rceil)(\lceil m/(s_1 - 1) \rceil - 1) \leq 2m^2(s_1 - 1)$ ways of assigning the pairs of integers.

Let k be the number of variables in term t_i such that exactly one of the vertices corresponding to the variables is assigned a pair of integers so far. Then there exist at most $l(l-1)/2 - k$ variables in term t_i such that none of the vertices associated with the variables has been assigned a pair of integers so far. So the number of ways of choosing an unassigned vertex in the endpoints of variables in t_i and assigning a pair of integers to the chosen vertex is at most

$$\begin{aligned} &\max_{0 \leq k \leq l(l-1)/2} \{k(m/(s_1 - 1)) + \sqrt{(l(l-1)/2 - k)(2m^2/(s_1 - 1))}\} \\ &\leq (l(l-1)/2)(m/(s_1 - 1)) + \sqrt{(l(l-1)/2)(2m^2/(s_1 - 1))}. \end{aligned} \tag{4.2}$$

This is because doing something to two vertices in i ways can be regarded as doing something to a vertex appropriately in $\sqrt{i}$ ways twice successively. Recall that $l = \lfloor s_1^{1/3}/4 \rfloor$. We have

$$\text{Eq.(4.2)} \leq \frac{s_1^{2/3}}{32}\frac{2m}{s_1} + \sqrt{\frac{s_1^{2/3}}{32}\frac{4m^2}{s_1}} < \frac{m}{32s_1^{1/3}} + \frac{m}{2s_1^{1/6}} < \frac{m}{s_1^{1/6}}.$$

By the definition of $\overline{\nabla}$ gate, a bad graph u corresponding to a mapping specified

in this way satisfies $f^D(u) = 0$ and $f^C(u) = 1$ only if there exist more than r vertices assigned to pairs of integers in the above procedure. Thus the number of mappings corresponding to such bad graphs is at most the number of the ways of assigning the $r + 1$ vertices to the pairs of integers in the manner described above multiplied by the number of ways of assigning the remaining $m - r - 1$ vertices arbitrarily to the remaining distinct pairs of integers, which is given by $(m/s_1^{1/6})^{r+1}(m - r - 1)!$. This completes the proof. □

LEMMA 4.5. *The number of good graphs in $I(m, s_2)$ for which the AND and $\overline{\wedge}$ gates produce different outputs (the AND gate produces 1, whereas the $\overline{\wedge}$ gate produces 0) is at most*

$$\frac{(2rs_2)^{l+1}(m - l - 1)!}{s_2!(m - s_2)!}.$$

Proof. The proof is similar to that of Lemma 4.4. Suppose an AND gate and an $\overline{\wedge}$ gate are given as input for the same monotone CNF formulas, denoted f_1^C and f_2^C. Let $f^C = f_1^C \wedge f_2^C$ and $f^D = f_1^C \,\overline{\wedge}\, f_2^C$. Let $c_1, \ldots, c_q$ be the complete list of clauses in f^C. Note that, for each clause c_i, the size of c_i is at most r and so the clause c_i contains at most $r(r-1)/2$ variables. The number in question is equal to the number of good graphs v such that $f^C(v) = 1$ and $f^D(v) = 0$.

Instead of the mappings from vertices to pairs of integers in the case of Lemma 4.4, we consider one to one mappings from the vertex set to the set of integers $\{1, \ldots, m\}$. Such a mapping is thought to specify a good graph such that the set of vertices assigned with integers from 1 to s_2 forms a clique. Since each good graph in $I(m, s_2)$ is represented as exactly $s_2!(m - s_2)!$ mappings, it suffices to show that the number of mappings corresponding to good graphs v such that $f^C(v) = 1$ and $f^D(v) = 0$ is at most $(2rs_2)^{l+1}(m - l - 1)!$.

As in the proof of Lemma 4.4 we proceed from c_1 up to c_q repeatedly by assigning one or two vertices to integers from $\{1, \ldots, m\}$ so that the resulting good graph v satisfies the condition that $f^C(v) = 1$ and $f^D(v) = 0$. Suppose that we somehow already assigned distinct integers to the endpoints of variables from clauses $c_1, \ldots, c_{i-1}$ so as to make all these clauses take the value 1 and we proceed to clauses c_i. We now choose one or two vertices from the endpoints of variables from c_i in a way similar to that in the proof of Lemma 4.4 and then proceed to the next clause c_{i+1}. As in the case of the proof of Lemma 4.4, we only need to consider the case where there remain variables in c_i such that assigning one or two endpoints, not chosen so far, of these variables makes the clause c_i take the value 1. Since the number of endpoints of vertices in c_i is at most r, the number of ways of choosing a vertex and assigning an integer in the manner described above is at most $r(s_2 - 1)$, while the

number of the ways of choosing two adjacent vertices and assigning integers is at most $(r(r-1)/2)s_2(s_2-1)$. Thus the number of the ways per a vertex is at most

$$r(s_2-1)+\sqrt{(r(r-1)/2)s_2(s_2-1)} < 2rs_2.$$

Thus, in a way similar to the proof of Lemma 4.4, it is easily seen that the number of mappings corresponding to good graphs v such that $f^C(v)=1$ and $f^D(v)=0$ is at most $(2rs_2)^{l+1}(m-l-1)!$ because clauses of length more than l are taken away when CNF formula $f_1^C \wedge f_2^C$ is transformed into DNF formulas $f_1^C \overline{\wedge} f_2^C$. This completes the proof of the lemma. □

Proof of Theorem 4.1. Assume that a monotone circuit C is such that $\Pr_{v\in I(m,s_2)}[C(v)=1] \geq h(s_2)$ and $\Pr_{u\in O(m,s_1)}[C(u)=0] > 2/s_1^{1/3}$ hold. To prove the theorem, it suffices to show size$(C) \geq (h/2)2^{s^{1/3}/4}$. From Lemma 4.3, the approximator circuit $\overline{C}$ satisfies $\Pr_{v\in I(m,s_2)}[C(v) \neq \overline{C}(v)] \geq h(s_2)$ or $\Pr_{u\in O(m,s_1)}[C(u) \neq \overline{C}(u)] > 1/s_1^{1/3}$. Thus, in view of Fact 4.2, Lemmas 4.3, 4.4 and 4.5, the size of C is at least

$$\frac{1}{2}\min\left(\frac{h(s_2)m!}{(2rs_2)^{l+1}(m-l-1)!}\ ,\ \frac{m!}{s_1^{1/3}(m/s_1^{1/6})^{r+1}(m-r-1)!}\right). \tag{4.3}$$

The coefficient $1/2$ here is needed to take into account the fact that circuit C is assumed to be modified so the AND and OR gates alternate along any path from an input to the output in the circuit. An elementary calculation completes the proof. □

5. Proof of the Main Theorem. The goal of this section is to prove Theorem 5.1, which says that $\lfloor (1/6)\log\log m \rfloor$ NOT gates are not enough to compute the clique function feasibly. We do not intend here to optimize the constant $1/6$ in the number of NOT gates.

THEOREM 5.1. *For any sufficiently large integer m,*

$$size_{\lfloor (1/6)\log\log m\rfloor}(CLIQUE(m,(\log m)^{3(\log m)^{1/2}})) > 2^{(1/5)(\log m)^{(\log m)^{1/2}}}.$$

Before proceeding to the proof, we describe the idea behind the proof.

Let f be CLIQUE(m,s). Suppose to the contrary that a small circuit C with t NOT gates computes f. By Theorem 3.2, there are $2^{t+1}-1(=\alpha)$ monotone functions $f_1,\ldots,f_\alpha$ such that each of them can be computed by a monotone circuit and that $\cup_{i\in\{1,\ldots,\alpha\}} G(f_i) \supseteq G(f)$.

All proofs of lower bounds on the size of a monotone circuit computing a certain function are based on the observation that the circuit separates the minterms and maxterms of the target function [2, 5, 8, 9, 10, 13] (See Fig. 5.1). Instead of focusing on the separation of the minterms and maxterms of the target function f, we consider

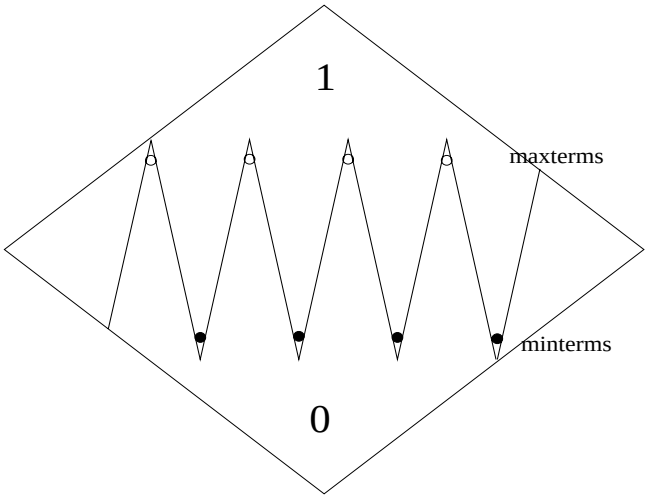

FIG. 5.1. *A figure showing the minterms and maxterms of CLIQUE in the Boolean cube.*

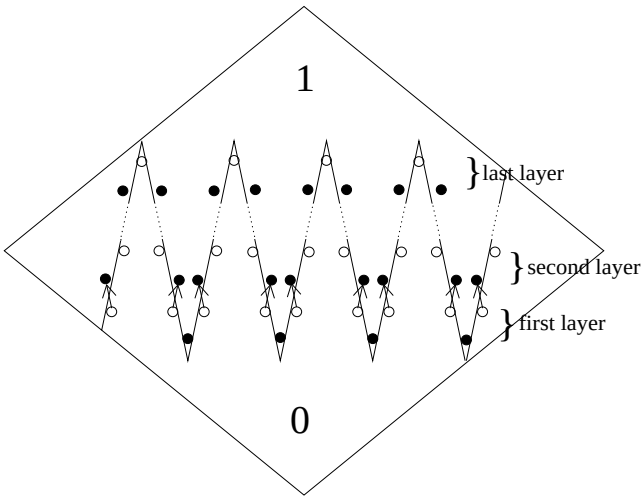

FIG. 5.2. *Layered Structure of the good and bad graphs: The good graphs and bad graphs are described as solid circles and open circles, respectively. A small monotone circuit cannot separate its good graphs and bad graphs in any one layer. When a bad graph u in the i-th layer is connected by an arrow to a good graph v in the $(i+1)$-th layer, they satisfy the condition $v \geq u$.*

separating pairs of vectors (w, w') in $G(f)$, that is, pairs (w, w') such that $f(w) = 0$, $f(w') = 1$ and $\mathrm{Ham}(w, w') = 1$. Theorem 3.2 says that, if f can be computed by a small circuit C with t NOT gates, then it follows that there exists a collection of monotone functions $f_1, \ldots, f_\alpha$ such that they separate $G(f)$ (i.e., $\cup_{i \in \{1,\ldots,\alpha\}} G(f_i) \supseteq G(f)$) and that each of functions $f_1, \ldots, f_\alpha$ can be computed by a small monotone circuit, where $\alpha = 2^{t+1} - 1$. In what follows we shall show that the circuit C is not small by showing that it is not the case that all of the function f_i's satisfying the

above condition can be computed by small monotone circuits. In fact we only pay attention to a subset of $G(f)$ which contains pairs that seem to be hard to separate. As the following example shows, it is crucial to decide which subset of $G(f)$ we pay attention to. As an example of a bad choice for a subset, take the subset $G'(f)$ defined as :

$$G'(f) = \{(u, u^+) \mid u \in O(m,s) \text{ and } (u,u^+) \in G(f)\}$$
$$\cup \{(v^-, v) \mid v \in I(m,s) \text{ and } (v^-, v) \in G(f)\}.$$

Clearly the number of 1's in u is the same for all $u \in O(m,s)$, and similarly for $v \in I(m,s)$. So if we set f_1 and f_2 to be the two threshold functions whose threshold values are the number of 1's in v and that in u^+, then we have $G(f_1) \cup G(f_2) \supseteq G'(f)$. On the other hand, it is known that a monotone circuit of size $O(n \log n)$ can compute a threshold function on n variables for any given threshold value. So we cannot derive a contradiction by the usual approximation method argument. We will give a subset which we use as the subset of $G(f)$ consisting of edges illustrated in Fig. 5.2.

Let $l_0 < l_1 < \cdots < l_\alpha$ be a monotone increasing sequence of integers, where $l_0 = s$, $l_\alpha = m$, and others are chosen appropriately later. For $1 \leq i \leq \alpha$, a graph v is called *good in the i-th layer* if v consists of a clique of size l_{i-1}, and contains no other edges. For $1 \leq i \leq \alpha$, a graph u is called *bad in the i-th layer* if there exist l_i vertices and a partition of these l_i vertices into $s-1$ blocks with nearly equal size such that u has an edge between any two vertices chosen from different blocks and no other edges. In other words, a bad graph in the i-th layer is a $(s-1)$-partite complete graph on some subset of vertices of size l_i. Note that, for any good graph v in the first layer, there is an edge in $G(f)$ whose head is v because deleting an edge from v breaks the clique in v. (Recall that each vertex of the boundary graph $G(f)$ corresponds to an input graph of the clique function.) Similarly, for any bad graph u in any layer, there is an edge in $G(f)$ whose tail is u because adding an appropriate edge (between vertices in a block) to u ends up with having a clique of size s. In what follows we will prove that if monotone functions $f_1, \ldots, f_\alpha$ separate $G(f)$ and each $f_1, \ldots, f_\alpha$ can be computed by a small monotone circuit, then a contradiction follows. In order to prove it, because we must consider α functions which are supposed to separate $G(f)$, we must fully exploit the computational complexity of the function CLIQUE, considering α layers and focusing on the separation between good and bad graphs in each layer.

Since $\cup_{i \in \{1,\ldots,\alpha\}} G(f_i) \supseteq G(f)$, there exists a function, say f_1, in $\{f_1, \ldots, f_\alpha\}$ such that $G(f_1)$ contains at least $1/\alpha$ fraction of edges of $G(f)$ ending at good graphs in the first layer, and hence f_1 outputs 1 on at least $1/\alpha$ fraction of good graphs in the first layer. Since we can use Theorem 4.1 to show that every small monotone

circuit that outputs 1 for a certain fraction of good graphs in the first layer must output 1 for a large number of bad graphs in the same layer, the function f_1 takes the value 1 for a large number of bad graphs u in the first layer. By adding an edge appropriately to such a u we get a graph u^+ which contains a clique of size s. Hence there are many edges, denoted (u, u^+), which are not included in the edges in $G(f_1)$. Since $\cup_{i\in\{1,\ldots,\alpha\}} G(f_i) \supseteq G(f)$, there exists a function, say f_2, in $\{f_2, \ldots, f_\alpha\}$ such that $G(f_2)$ contains at least $1/\alpha$ fraction of such edges (u, u^+). On the other hand, because f_2 is monotone and $f_2(u^+) = 1$, f_2 takes the value 1 on the good graph v in the second layer such that $u^+ \leq v$. Applying Theorem 4.1 again, we can conclude that f_2 outputs 1 for a large number of bad graphs in the second layer. It can be shown that f_1 also outputs 1 for such bad graphs.

By continuing the above argument, we can conclude that every function $f_1, \ldots,$ f_α outputs 1 on some bad graph u in the last layer, contradicting the fact that $\cup_{i\in\{1,\ldots,\alpha\}} G(f_i) \supseteq G(f)$. This is the outline of the proof.

Proof of Theorem 5.1. Let m be a sufficiently large integer. Put $t = \lfloor (1/6)\log\log m \rfloor$, $s = (\log m)^{3(\log m)^{1/2}}$, $M = 2^{(1/5)(\log m)^{(\log m)^{1/2}}}$ and $\alpha = 2^{t+1} - 1$. We suppose to the contrary that a circuit C with at most t NOT gates computes CLIQUE(m, s) and that size$(C) \leq M$. From Theorem 3.2, there are monotone functions $f_1, \ldots, f_\alpha \in \mathcal{M}^n$ such that $\mathrm{size}_{mon}(f_i) \leq M$ for any $1 \leq i \leq \alpha$, and $\bigcup_{i\in\{1,\ldots,\alpha\}} G(f_i) \supseteq G(\mathrm{CLIQUE}(m, s))$.

Let $l_0 = s$, $l_\alpha = m$ and for every $j = 1, \ldots, \alpha-1$, let $l_j = m^{1/10+(1/3)(j-1)/(\log m)^{1/6}}$. Since $l_{\alpha-1} \leq m^{1/10+(1/3)(2^{(1/6)\log\log m+1})/(\log m)^{1/6}} = m^{1/10+2/3} < m^{9/10}$, we have $l_0 < l_1 < \cdots < l_\alpha$. Let V be the set of m vertices of the graph associated with CLIQUE. For $j \in \{0, \ldots, \alpha\}$, let $\mathcal{L}_j$ denote $\{L \subseteq V \mid |L| = l_j\}$ and let $\mathcal{L}_j(L)$ denote $\{L' \subseteq L \mid L' \in \mathcal{L}_j\}$. For $i \in \{1, \ldots, \alpha\}$ and $L_i \in \mathcal{L}_i$, a graph v is called *good on the set L_i in the i-th layer* if it consists a clique of size l_{i-1} on some $L_{i-1} \in \mathcal{L}_{i-1}(L_i)$ (i.e., $|L_{i-1}| = l_{i-1}$ and $L_{i-1} \subseteq L_i$), and contains no other edges. For $i \in \{1, \ldots, \alpha\}$ and $L_i \in \mathcal{L}_i$, a graph u is called *bad on the set L_i in the i-th layer* if there is a partition of L_i into $V_1, \ldots, V_{s-1}$ such that

(i) $|V_i| \in \{\lfloor |L_i|/(s-1) \rfloor, \lceil |L_i|/(s-1) \rceil\}$ for $i = 1, \ldots, s-1$,

(ii) u has an edge (w, w') if and only if $w \in V_i$ and $w' \in V_j$ such that $i \neq j$, i.e., u is a complete $(s-1)$-partite complete graph on the vertex set L_i.

Let I_{L_i} (respectively, O_{L_i}) denote the set of all good (respectively, bad) graphs on the set L_i in the i-th layer. Note that a good graph in the first layer (respectively, a bad graph in the last layer) is a minterm (respectively, a maxterm) of CLIQUE(m, s). We also note that there is a one to one correspondence between I_{L_i} and $I(l_i, l_{i-1})$, and between O_{L_i} and $O(l_i, s)$, where $I(l_i, l_{i-1})$ and $O(l_i, s)$ are defined in Section 4. Hence a function in $F(l_i, s, l_{i-1})$ can be viewed as separating the graphs into two groups, I_{L_i} and O_{L_i}. Since $s^{1/3} l_{i-1} \leq l_i/200$ holds, the following corollary is straightforward

from Theorem 4.1. This corollary says that a small monotone circuit cannot separate O_{L_i} and I_{L_i} for any i and for any vertex set $L_i \in \mathcal{L}_i$.

COROLLARY 5.2. *Let $i \in \{1, \ldots, \alpha\}$ and $L_i \in \mathcal{L}_i$. Suppose that C is a monotone circuit and the fraction of good graphs in I_{L_i} (i.e., the set of good graphs on L_i in the i-th layer) such that C outputs 1 is at least h. Then at least one of the followings holds:*

(i) The number of gates in C is at least $(h/2)2^{s^{1/3}/4}$,

(ii) The fraction of bad graphs in O_{L_i} (i.e., the set of bad graphs on L_i in the i-th layer) such that C outputs 0 is at most $2/s^{1/3}$. □

Proof of Theorem 5.1 (continued). For $L \subseteq V$, let v_L denote a graph corresponding to a clique on the set L and having no other edges. Recall that $\mathcal{L}_0 = \{L \subseteq V \mid |L| = s\}$. Thus for any $L_0 \in \mathcal{L}_0$, there exists $u < v_{L_0}$ such that the edge (u, v_{L_0}) is in $G(\mathrm{CLIQUE}(m, s))$. Hence there exists i_1 in $\{1, \ldots, \alpha\}$ such that $\Pr_{L_0 \in \mathcal{L}_0}[\exists u < v_{L_0} \quad (u, v_{L_0}) \in G(f_{i_1})] \geq 1/\alpha > 1/2^{t+1}$ holds, and this implies

$$\Pr_{L_0 \in \mathcal{L}_0}[f_{i_1}(v_{L_0}) = 1] \geq \frac{1}{2^{t+1}}. \tag{5.1}$$

Then we can obtain

$$\Pr_{L_1 \in \mathcal{L}_1}\left[\Pr_{v \in I_{L_1}}[f_{i_1}(v) = 1] \geq \frac{1}{2^{t+2}}\right] \geq \frac{1}{2^{t+2}}. \tag{5.2}$$

This is because from Eq. (5.1), we have

$$\begin{aligned}\sum_{L_1 \in \mathcal{L}_1} |\{v \in I_{L_1} \mid f_{i_1}(v) = 1\}| &\geq \frac{1}{2^{t+1}}\binom{m}{l_0}\binom{m-l_0}{l_1-l_0} \\ &= \frac{1}{2^{t+1}}\binom{m}{l_1}\binom{l_1}{l_0} \qquad (5.3)\end{aligned}$$

But if Eq. (5.2) does not hold then

$$\begin{aligned}\sum_{L_1 \in \mathcal{L}_1} |\{v \in I_{L_1} \mid f_{i_1}(v) = 1\}| &< \binom{m}{l_1}\frac{1}{2^{t+2}}\binom{l_1}{l_0} + \binom{m}{l_1}\left(1 - \frac{1}{2^{t+2}}\right)\binom{l_1}{l_0}\frac{1}{2^{t+2}} \\ &< \frac{2}{2^{t+2}}\binom{m}{l_1}\binom{l_1}{l_0} = \frac{1}{2^{t+1}}\binom{m}{l_1}\binom{l_1}{l_0},\end{aligned}$$

contradicting Eq. (5.3).

Now we call $L_1 \in \mathcal{L}_1$ *dense* if $\Pr_{v \in I_{L_1}}[f_{i_1}(v) = 1] \geq 1/2^{t+2}$ holds. Put $h = 1/2^{t+2}$. An easy calculation shows $h \geq 1/m$. Thus by applying Claim 5.2 to every dense L_1, we have $\mathrm{size}_{mon}(f_{i_1}) \geq (1/2m)2^{s^{1/3}/4} = 2^{(1/4)(\log m)^{(\log m)^{1/2}} - \log m - 1} > M$ or $\Pr_{u \in O_{L_1}}[f_{i_1}(u) = 1] \geq 1 - 2/s^{1/3} \geq 1/2$ for any dense L_1. Since the former contradicts the assumption that $\mathrm{size}_{mon}(f_{i_1}) \leq M$, we have $\Pr_{u \in O_{L_1}}[f_{i_1}(u) = 1] \geq 1/2$ for any dense L_1. By Eq. (5.2), we have

$$\Pr_{L_1 \in \mathcal{L}_1}\left[\Pr_{u \in O_{L_1}}[f_{i_1}(u) = 1] \geq \frac{1}{2}\right] \geq \frac{1}{2^{t+2}}. \tag{5.4}$$

The proof will be by induction on a level of the layers. We use Eq. (5.4) as the basis of the induction, and the induction steps are as follows.

CLAIM 5.3. *Suppose $c_1 > 1$ and $c_2 > 1$. Put $c_3 = \alpha$. Let $f_1, \ldots, f_{c_3}$ be the monotone functions such that $\cup_{i\in\{1,\ldots,c_3\}} G(f_i) \supseteq G(CLIQUE(m,s))$ and $size_{mon}(f_i) \leq M$ for any $1 \leq i \leq c_3$. Suppose that, for distinct indices $i_1, \ldots, i_k \in \{1, \ldots, c_3\}$,*

$$\Pr_{L_k\in\mathcal{L}_k}\left[\Pr_{u\in O_{L_k}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq 1/c_1\right] \geq 1/c_2$$

holds. If $c_1c_2c_3 \leq s^{1/3}/8$, then there exists $i_{k+1} \in \{1, \ldots, c_3\}\backslash\{i_1, \ldots, i_k\}$ such that

$$\Pr_{L_{k+1}\in\mathcal{L}_{k+1}}\left[\Pr_{u\in O_{L_{k+1}}}\left[f_{i_1}(u) = \cdots = f_{i_k}(u) = f_{i_{k+1}}(u) = 1\right] \geq \frac{1}{4c_1c_2c_3}\right] \geq \frac{1}{2c_2c_3}.$$

For a proof of this claim, see the appendix.

Proof of Theorem 5.1 (continued). First we claim that for any $k \in \{1, \ldots, \alpha\}$, there are k distinct indices $i_1, \ldots, i_k \in \{1, \ldots, \alpha\}$ such that

$$\Pr_{L_k\in\mathcal{L}_k}\left[\Pr_{u\in O_{L_k}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq \frac{1}{2^{k^2(t+2)}}\right] \geq \frac{1}{2^{k(t+2)}} \tag{5.5}$$

holds. The claim is proved by induction on k. The basis, $k = 1$, is trivial from Eq. (5.4). Now we suppose the claim holds for any $k \leq l$ and let $k = l + 1$. By the induction hypothesis, we have

$$\Pr_{L_l\in\mathcal{L}_l}\left[\Pr_{u\in O_{L_l}}[f_{i_1}(u) = \cdots = f_{i_l}(u) = 1] \geq \frac{1}{2^{l^2(t+2)}}\right] \geq \frac{1}{2^{l(t+2)}}.$$

Putting $c_1 = 2^{l^2(t+2)}$, $c_2 = 2^{l(t+2)}$ and $c_3 = \alpha$, we have $4c_1c_2c_3 \leq 2^{2+l^2(t+2)+l(t+2)+(t+1)} \leq 2^{(l+1)^2(t+2)}$, $2c_2c_3 \leq 2^{1+l(t+2)+t+1} = 2^{(l+1)(t+2)}$ and $c_1c_2c_3 \leq 2^{(l+1)^2(t+2)}/4 \leq 2^{(2^{t+1})^2(t+2)}/4 \leq 2^{2^{3t}}/8 \leq 2^{2^{(1/2)\log\log m}}/8 = 2^{\sqrt{\log m}}/8 < (\log m)^{\sqrt{\log m}}/8 = s^{1/3}/8$. Thus by Claim 5.3,

$$\Pr_{L_{l+1}\in\mathcal{L}_{l+1}}\left[\Pr_{u\in O_{L_{l+1}}}\left[f_{i_1}(u) = \cdots = f_{i_{l+1}}(u) = 1\right] \geq \frac{1}{4c_1c_2c_3}\right] \geq \frac{1}{2c_2c_3}$$

holds. Therefore

$$\begin{aligned}\Pr_{L_{l+1}\in\mathcal{L}_{l+1}}\left[\Pr_{u\in O_{L_{l+1}}}\left[f_{i_1}(u) = \cdots = f_{i_{l+1}}(u) = 1\right] \geq \frac{1}{2^{(l+1)^2(t+2)}}\right] &\geq \frac{1}{2c_2c_3}\\ &\geq \frac{1}{2^{(l+1)(t+2)}}.\end{aligned}$$

This completes the induction step and hence the proof of the claim.

Recalling $\mathcal{L}_\alpha = \{V\}$ and setting k in Eq. (5.5) to α, we have $\Pr_{u\in O_V}[\forall i \in \{1, \ldots, \alpha\} \quad f_i(u) = 1] > 0$. Thus there exist $u \in O_V$ and $u^+ \in \mathrm{CLIQUE}(m,s)^{-1}(1)$ such that $(u, u^+) \in G(\mathrm{CLIQUE}(m,s))$ and $(u, u^+) \notin G(f_i)$ for any $i \in \{1, \ldots, \alpha\}$. This implies that $\bigcup_{i\in\{1,\ldots,\alpha\}} G(f_i) \not\supseteq G(\mathrm{CLIQUE}(m,s))$, completing the proof. □

6. Concluding Remarks. There are still many interesting questions yet to be answered in the line of research pursued in the present paper. An obvious challenge is to improve the number of negation gates in the main theorem to $\omega(\log\log n)$. Another interesting problem is to show a tradeoff between the circuit size and the number of negation gates in a circuit to compute a certain monotone function. Analyzing the size complexity more carefully along the line suggested in this paper might help to explore such a tradeoff. Note that we have recently proved that such a tradeoff exists for the merging function MERGE(n,n), which is a collection of monotone functions that merges two presorted binary sequence each of length n into a sorted sequence of length $2n$, by showing size$_t$(MERGE(n,n)) $= \Theta(n\log n/2^t)$ for every $t = 0,\ldots,\log\log n$ [3].

Finally, it should be noted that there is a large obstacle in generalizing our techniques to obtain a good lower bound for a circuit without restricting the number of negation gates. This comes from the notion of "Natural Proofs" introduced by Razborov and Rudich [12]. They proved that almost all known combinatorial lower bound proof techniques are "natural", and such proofs cannot yield a good lower bound for general circuit complexity under some commonly believed cryptographic assumption. Our techniques seem to fall under the category of Natural Proofs although we have not tried to give a formal proof. Some radically different techniques would be needed to improve the number of negation gates in our main theorem to, say, $\log n$.

7. Acknowledgments. The authors would like to thank the anonymous referees for their helpful suggestions on improving the presentation of this paper.

Appendix. *Proof of Claim 5.3.* Let $\mathcal{L}_k^{bad}$ denote the collection of sets $L_k \in \mathcal{L}_k$ with $\Pr_{u\in O_{L_k}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq 1/c_1$. By the assumption of Claim 5.3, we have

$$\Pr_{L_k\in\mathcal{L}_k}[L_k \in \mathcal{L}_k^{bad}] \geq \frac{1}{c_2}. \tag{A.1}$$

Let $u \in O_{L_k}$ be such that $f_{i_1}(u) = \cdots = f_{i_k}(u) = 1$. By the definition of boundary graphs, none of $G(f_{i_1}),\ldots,G(f_{i_k})$ contains an edge from u. Note that CLIQUE$(m,s)(u) = 0$. Let u^+ be a graph obtained from u by adding an arbitrary edge whose both endpoints are in L_k. Clearly, Ham$(u,u^+) = 1$, CLIQUE$(m,s)(u^+) = 1$ and $(u,u^+) \in G(\text{CLIQUE}(m,s))$. Since $u^+ \leq v_{L_k}$, we have

$$\forall L_k \in \mathcal{L}_k^{bad}\exists u \in O_{L_k}\exists u^+ \leq v_{L_k} \quad (u,u^+) \in \bigcup_{j\in\{1,\ldots,c_3\}\setminus\{i_1,\ldots,i_k\}} G(f_j).$$

Therefore there exists $l \in \{1,\ldots,c_3\}\setminus\{i_1,\ldots,i_k\}$ such that

$$\Pr_{L_k\in\mathcal{L}_k^{bad}}[\exists u \in O_{L_k}\exists u^+ \leq v_{L_k} \quad (u,u^+) \in G(f_l)] \geq \frac{1}{c_3}$$

holds. If $(u, u^+) \in G(f_l)$ for $u \in O_{L_k}$, then $f_l(u^+) = 1$, which together with $u^+ \leq v_{L_k}$ implies $f_l(v_{L_k}) = 1$ by the monotonicity of f_l. Thus we can conclude that $\Pr_{L_k \in \mathcal{L}_k}[f_l(v_{L_k}) = 1 \mid L_k \in \mathcal{L}_k^{bad}] \geq 1/c_3$. From this and Eq. (A.1), there exists $l \in \{1, \ldots, c_3\}/\{i_1, \ldots, i_k\}$ such that

$$\Pr_{L_k \in \mathcal{L}_k}[L_k \in \mathcal{L}_k^{bad} \text{ and } f_l(v_{L_k}) = 1] \geq \frac{1}{c_2 c_3}. \tag{A.2}$$

Now we choose an index l arbitrarily that satisfies the above inequality and let $i_{k+1} = l$. Letting $\mathcal{L}_k^{target}$ denote a collection of sets $L_k \in \mathcal{L}_k$ such that $L_k \in \mathcal{L}_k^{bad}$ and $f_{i_{k+1}}(v_{L_k}) = 1$, we have $\Pr_{L_k \in \mathcal{L}_k}[L_k \in \mathcal{L}_k^{target}] \geq 1/(c_2 c_3)$. By a similar arguments to the derivation of Eq. (5.2), we have

$$\Pr_{L_{k+1} \in \mathcal{L}_{k+1}}\left[\Pr_{L_k \in \mathcal{L}_k(L_{k+1})}[L_k \in \mathcal{L}_k^{target}] \geq \frac{1}{2c_2 c_3}\right] \geq \frac{1}{2c_2 c_3}. \tag{A.3}$$

Now we call a $L_{k+1} \in \mathcal{L}_{k+1}$ *dense* if

$$\Pr_{L_k \in \mathcal{L}_k(L_{k+1})}[L_k \in \mathcal{L}_k^{target}] \geq \frac{1}{2c_2 c_3} \tag{A.4}$$

holds, and let $\mathcal{L}_{k+1}^{dense}$ denote a collection of all dense sets in $\mathcal{L}_{k+1}$. Note that

$$\Pr_{v \in I_{L_{k+1}}}[f_{i_{k+1}}(v) = 1] \geq 1/(2c_2 c_3)$$

for any dense $L_{k+1} \in \mathcal{L}_{k+1}^{dense}$. Put $h = 1/(2c_2 c_3) > 1/m$. Thus by applying Claim 5.2 to every dense L_{k+1}, we have $\mathrm{size}_{mon}(f_{i_{k+1}}) > (1/2m)2^{s^{1/3}/4} > M$ or $\Pr_{u \in O_{L_{k+1}}}[f_{i_{k+1}}(u) = 0] \leq 2/s^{1/3} \leq 1/(4c_1 c_2 c_3)$ for any dense L_{k+1}. (We use the assumption $c_1 c_2 c_3 \leq s^{1/3}/8$ in Claim 5.3 here.) Since the former contradicts the assumption $\mathrm{size}_{mon}(f_{i_{k+1}}) \leq M$, we have

$$\Pr_{u \in O_{L_{k+1}}}[f_{i_{k+1}}(u) = 0] \leq \frac{1}{4c_1 c_2 c_3}, \tag{A.5}$$

for any $L_{k+1} \in \mathcal{L}_{k+1}^{dense}$. By Eq. (A.4), for any dense $\mathcal{L}_{k+1}$, we have

$$\Pr_{L_k \in \mathcal{L}_k(L_{k+1})}\left[\Pr_{u \in O_{L_k}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq \frac{1}{c_1}\right] \geq \frac{1}{2c_2 c_3}. \tag{A.6}$$

From the above inequality, we can get

$$\Pr_{u \in O_{L_{k+1}}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq \frac{1}{2c_1 c_2 c_3}. \tag{A.7}$$

To derive this, we consider the bipartite graph $G = (U_1, U_2, E)$ with vertex sets $U_1 = O_{L_{k+1}}$ and $U_2 = \cup_{L_k \in \mathcal{L}_k(L_{k+1})} O_{L_k}$ and the edge set

$$E = \{(u_1, u_2) \in U_1 \times U_2 \mid u_1 \geq u_2\}.$$

Let U_2^{mark} be the set of $u_2 \in U_2$ such that $f_{i_1}(u_2) = \cdots = f_{i_k}(u_2) = 1$ and let $N(U_2^{mark}) \subseteq U_1$ be the set of vertices adjacent to vertices in U_2^{mark}. By the monotonicity of f_i's, for every $u_1 \in N(U_2^{mark})$, $f_{i_1}(u_1) = \cdots = f_{i_k}(u_1) = 1$ holds. From Eq. (A.6), we have $|U_2^{mark}| \geq |U_2|/(2c_1c_2c_3)$. Clearly every vertex in U_1 has the same degree, and similarly for U_2. Hence we can prove that $|N(U_2^{mark})| \geq |U_1|/(2c_1c_2c_3)$, which implies Eq. (A.7).

By Eqs. (A.5) and (A.7), we have

$$\Pr_{u \in O_{L_{k+1}}} \left[f_{i_1}(u) = \cdots = f_{i_{k+1}}(u) = 1 \right] \geq \frac{1}{2c_1c_2c_3} - \frac{1}{4c_1c_2c_3} = \frac{1}{4c_1c_2c_3}$$

for any $L_{k+1} \in \mathcal{L}_{k+1}^{dense}$. Claim 5.3 is straightforward from this and Eq. (A.3). □

REFERENCES

[1] N. Alon and R. B. Boppana, *The monotone circuit complexity of Boolean functions*, Combinatorica, 7(1) (1987), pp. 1–22.

[2] K. Amano and A. Maruoka, *The potential of the approximation method*, SIAM J. on Comput. 33(2) (2004), pp. 433–447. (A preliminary version was in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, Burlington, Vermont, 1996, pp. 431–440)

[3] K. Amano, A. Maruoka and J. Tarui, *On the negation-limited circuit complexity of merging*, Discrete Applied Mathematics, 126(1) (2003), pp. 3–8.

[4] R. Beals, T. Nishino and K. Tanaka, *More on the complexity of negation-limited circuits*, in Proceedings of the 27th Annual ACM Symposium on Theory of Computing, Las Vegas, Nevada, 1995. pp. 585–595.

[5] C. Berg and S. Ulfberg, *Symmetric approximation arguments for monotone lower bounds without sunflowers*, *Comput. Complexity*, 8(1) (1999) pp. 1–20.

[6] M.J. Fischer, *The complexity of negation-limited networks – A brief survey*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 33, (1974) pp. 71–82.

[7] J. Håstad, *Almost optimal lower bounds for small depth circuits*, in Proceedings of the 18th Annual ACM Symposium on Theory of Computing, Berkeley, California, 1986, pp. 6–20.

[8] D. Harnik and R. Raz, *Higher lower bounds on monotone size*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, Portland, OR, 2000, pp. 378–387.

[9] S. Jukna, *Finite limits and monotone computations: The lower bounds criterion*, in Proceedings of the 12th IEEE Conference on 12th Computational Complexity, Ulm, Germany, 1997, pp. 302–313.

[10] A.A. Razborov, *Lower bounds on the monotone complexity of some Boolean functions*, Soviet Math. Doklady, 31 (1985), pp. 354–357.

[11] A.A. Razborov, *On the method of approximations*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, 1989, pp. 167–176.

[12] A.A. Razborov and S. Rudich, *Natural Proofs*, J. Comput. System Sci., 55(1) (1997), pp. 24–35.

[13] J. Simon and S.C. Tsai, *On the bottleneck counting argument*, Theoret. Comput. Sci., 237 (2000), pp. 429–437.

[14] M. Santha and C. Wilson, *Limiting negations in constant depth circuits*, SIAM J. on Comput., 22(2) (1993) pp. 294–302.