In *34th Annual IEEE Symposium on Foundations of Computer Science*, pages 604–613, November 1993.

Directed vs. Undirected Monotone Contact Networks for Threshold Functions

Magnús Halldórsson* Jaikumar Radhakrishnan† K. V. Subrahmanyam†

Abstract

We consider the problem of computing threshold functions using directed *and* undirected *monotone contact networks. Our main results are the following. We show that there exist* directed *monotone contact networks that compute T_k^n, $2 \le k \le n-1$, of size $O(k(n-k+2)\log(n-k+2))$. This bound is almost optimal for small thresholds, since there exists an $\Omega(kn\log(n/(k-1)))$ lower bound. Our networks are described explicitly; the previously best upper bound known, obtained from the undirected networks of Dubiner and Zwick, used non-constructive arguments and gave directed networks of size $O(k^{3.99}n\log n)$. We show a lower bound of $\Omega(n\log\log\log n)$ on the size of* undirected *monotone contact networks computing T_{n-1}^n, improving the $2(n-1)$ lower bound of Markov. Combined with our upper bound result, this shows that directed monotone contact networks compute some threshold functions more easily than undirected networks.*

1 Introduction

1.1 Monotone contact networks and threshold functions

Let n and k be positive integers such that $1 \le k \le n$. The k-th *threshold function* T_k^n is a Boolean function on n variables that takes the value 1 precisely when there are at least k 1's in the input. Threshold functions have been widely studied in various circuit models (see Boppana and Sipser [BS], Wegener [W]). However, as noted by Karchmer and Wigderson [K], the computation of threshold functions is not completely understood even in the monotone formulas model. In this paper, we study the computation of threshold functions in a model related to monotone formulas, namely *monotone contact networks.*

A *monotone contact network* is a graph where each edge has a variable as its label. (In non-monotone networks, negated variables are also allowed to appear as labels.) We use $\mathrm{vars}(N)$ to denote the set of variables of the network N. For a pair (v,w) of vertices, the contact network computes the Boolean function $f_{v,w}$ as follows. On an assignment $y : \mathrm{vars}(N) \to \{0,1\}$, each edge is set to 0 or 1 according to the value of its label. Then $f_{v,w}(y) = 1$ if there is a path from v to w using only the edges with value 1, and $f_{v,w}(y) = 0$ otherwise. If N is a contact network with two distinguished vertices s (start) and t (terminal), then we refer to the function $f_{s,t}$ as the function computed by N. The *size* of a network is the number of edges in it.

It was shown by Karchmer and Wigderson [KW] that any formula computing the (s,t)-connectivity function needs size $n^{\Omega(\log n)}$. It is natural, then, to ask what functions are hard to compute even if we are given the power of connectivity. Monotone contact networks provide the natural model to study such questions. Monotone contact networks compute the (s,t)-connectivity function naturally in linear size. Moreover, any attempt to augment the power of monotone formulas by allowing connectivity gates leads immediately to the monotone contact networks model.

Monotone contact networks can be classified based on the nature of the underlying graph. When the underlying graph is undirected, such networks are called undirected monotone contact networks. We use $\mathrm{size}_{\mathrm{U}}(f)$ to denote the size of the smallest such network computing the function f. Improving the $\Omega(n\log n/\log\log n)$ lower bound of Lupanov [L], Krichevskii [Kr] showed that $\mathrm{size}_{\mathrm{U}}(T_2^n) = \Omega(n\log n)$. Since monotone formulas are a special case of such networks, upper bounds for monotone formulas apply to undirected monotone contact networks as well. Using the amplification method, Boppana [B] showed that there exist monotone formulas computing T_k^n of size $O(k^{4.3}n\log n)$. Applying the

*School of Information Science, Japan Advanced Institute of Science and Technology – Hokuriku, Tatsunokuchi, Ishikawa 923-12, JAPAN. magnus@jaist-east.ac.jp

†Theoretical Computer Science Group, Tata Institute of Fundamental Research, Bombay, INDIA 400 005. Work done while visiting JAIST.

amplification method directly to monotone undirected contact networks, Dubiner and Zwick [DZ] showed the existence of undirected monotone contact networks of size $O(n^{4.99})$, computing the majority function $T^n_{\lceil n/2 \rceil}$. Their method, when applied to other thresholds, yields $\mathrm{size}_\mathrm{U}(T^n_k) = O(k^{3.99} n \log n)$.

When the underlying graph is directed, such networks are called directed monotone contact networks. We use $\mathrm{size}_\mathrm{D}(f)$ to denote the size of the smallest such network computing the function f. Since undirected networks can be converted to directed networks by replacing each undirected edge by a pair of directed edges, an $O(k^{3.99} n \log n)$ upper bound for computing T^n_k and T^n_{n-k+1} holds also in this model. A lower bound of $\Omega(kn \log(n/(k-1))$ on the size of any directed monotone network computing T^n_k, $2 \le k \le \frac{n}{2}$, was obtained by Radhakrishnan [R]. If constant 1's are allowed to appear as labels in these networks, then they are called monotone contact-rectifier networks [Rz]. In this model the edges with label 1 are not considered when computing the size of the network. We use $\mathrm{size}_\mathrm{R}(f)$ to denote the size of the smallest such network computing the function f. The problem of computing threshold functions using monotone contact-rectifier networks was completely solved by Markov; he showed that $\mathrm{size}_\mathrm{R}(T^n_k) = k(n-k+1)$ (see also Moore and Shannon [MS]).

For the undirected and directed monotone contact networks models, however, these upper and lower bounds are not close. For $k \le n/2$, note the gap between the $\Omega(kn \log(n/(k-1))$ lower bound and the $O(k^{3.99} n \log n)$ upper bound for $\mathrm{size}_\mathrm{D}(T^n_k)$. For $k > n/2$, the best lower bound known is only $k(n-k+1)$ and this discrepancy is more pronounced; for instance, the best lower bound known for computing T^n_{n-1} is $2(n-1)$ which holds even for the stronger contact-rectifier networks model, while the best upper bound known is $O(n \log n)$ obtained by converting monotone formulas into monotone contact networks.

1.2 Our results

We show better upper bounds for directed monotone networks computing threshold functions.

Theorem 1 *For* $k = 2, 3, \ldots, n-1$, $\mathrm{size}_\mathrm{D}(T^n_k) = O(k(n-k+2)\log(n-k+2))$.
(T^n_1 and T^n_n have easy linear size networks.)

In contrast to the previous upper bound results that apply to this model, our networks are described explicitly without recourse to probabilistic existence arguments. Note that this upper bound comes close to the $\Omega(kn \log(n/(k-1)))$ lower bound for small thresholds; for the majority function it gives directed networks of size $O(n^2 \log n)$, whereas the best lower bound known is $\Omega(n^2)$. This result is also interesting for large thresholds. For T^n_{n-1} it promises linear size directed networks (in fact, the network for T^n_{n-1} is planar). This is in sharp contrast to many related models (e.g. monotone contact-rectifier networks, monotone formulas) where the complexities of computing T^n_k and T^n_{n-k+1} are the same. However, our linear upper bound for T^n_{n-1} and the $n \log n$ lower bound for T^n_2 cited above show that directed networks are more suited for large thresholds than small thresholds.

Motivated by this discovery, we ask if linear size monotone *undirected* networks exist for T^n_{n-1}. The upper bound of Theorem 1 makes essential use of the directedness of the networks. On the other hand, the best lower bound known, $2(n-1)$, does not preclude a linear upper bound.

Theorem 2 $\mathrm{size}_\mathrm{U}(T^n_{n-1}) = \Omega(n \log\log\log n)$.

The best upper bound known, obtained from formulas, is $O(n \log n)$. An $\Omega(n \log n)$ lower bound can be shown for two-terminal planar undirected contact networks by appealing to their strong duality property (see Harrison [H56]); for such networks the complexities of computing a Boolean function f and its dual[1] f^* are the same. In light of the disparity in the complexities of computing T^n_2 and T^n_{n-1}, no such duality result can be expected for directed monotone contact networks. Surprisingly, there is such a duality result for monotone contact-rectifier networks.

Theorem 3 *For two-terminal planar contact-rectifier networks the functions f and f^* have the same complexity.*

The upper bound in Theorem 1 was obtained by eliminating the rectifiers (constant 1's) from Markov's $k(n-k+1)$ size contact-rectifier networks. This suggests the possibility that rectifiers can always be eliminated with only a modest increase in size. Using a counting argument we show that this is not always possible.

Theorem 4 $\forall n > 1 \, \exists f : \{0,1\}^n \to \{0,1\}$ *such that* $\mathrm{size}_\mathrm{D}(f) \ge (n/(10 \log n))\, \mathrm{size}_\mathrm{R}(f)$.

[1] The dual function f^* is defined by $f^*(x) = \neg f(\neg x)$, where $\neg x$ denotes the component-wise negation of x.

Note that $\mathrm{size}_{\mathrm{D}}(f) \leq n \cdot \mathrm{size}_{\mathrm{R}}(f)$ because we may replace an edge labelled 1 with n edges in parallel with the n variables as labels without altering the function computed by the network.

We believe that the undirected and directed networks provide a clean setting for comparing the powers of directed and undirected connectivity. For example, a question of Stockmeyer (see Grigini and Sipser [GS, p. 73]) asks if $\mathrm{size}_{\mathrm{U}}$(directed (s,t)-connectivity) is polynomial. Theorem 1 and Theorem 2 give a separation of a factor of $\log\log\log n$ in the powers of the directed and undirected models for the function T^n_{n-1}. However, it is possible to obtain a slightly bigger separation for a somewhat less natural function.

Theorem 5 Let $F(x_1, x_2, \ldots, x_n, y) \equiv T^n_2(x_1, x_2, \ldots, x_n) \wedge y$. Then $\mathrm{size}_{\mathrm{D}}(F) \leq 3n$ whereas $\mathrm{size}_{\mathrm{U}}(F) \geq n \log n$.

In the remainder of this extended abstract we discuss our main contributions, Theorem 1 and Theorem 2. The upper bound of Theorem 1 is discussed in section 2. This result is obtained by replacing the constant 1's in Markov's contact-rectifier networks with variables. The main idea is to view Markov's network as a composition of certain networks called *Selector* networks. The upper bound then follows easily by observing that Selector networks can be realized efficiently without using constant 1's. We only sketch the main ideas of this proof.

The lower bound of Theorem 2 is discussed in section 3. Inevitably, we need to use properties of undirected graphs that set them apart from directed graphs. For example, our proof makes essential use of the following property that holds only for undirected graphs: *every (s,t)-path includes an odd number of edges from every minimal (s,t)-cutset.* To get our result we combine such graph theoretic properties with a notion of approximate threshold computation. Although each step in the proof uses elementary arguments, the proof as a whole is non-trivial and interesting. Since the method used is unusual, we present the proof in detail. The proofs of the remaining results claimed above are omitted from this extended abstract.

2 Directed Networks for T^n_k

Markov's construction. Markov's network computing T^n_k can be described as an $(n-k+1) \times (k+1)$ grid. The top left corner of the grid is vertex s and the bottom right corner vertex t. Edges are directed from left to right along rows and from top to bottom along columns. The rows are labelled 1 to $n-k+1$ from top to bottom. The edges in row i have labels $x_i, x_{i+1}, \ldots, x_{i+k-1}$ from left to right. Every vertical edge has 1 as its label. Note that we may collapse the first column into s and the last column into t without changing the behavior of the network.

The 1's in Markov's construction can be replaced by $(n-k+1)$ edges in parallel with labels x_1, $x_2, \ldots, x_{n-k+1}$ respectively. The network continues to compute T^n_k and is of size $O(k(n-k+1)^2)$. This construction, while close to the lower bound for very large thresholds, is much inferior to the $O(k^{3.99} n \log n)$ upper bound for small thresholds. We show how the 1's can be eliminated more efficiently.

Our construction. We shall need the following notation. Let N be a network with l start vertices, $s_1, s_2, \ldots, s_l$, and l end vertices, $t_1, t_2, \ldots, t_l$. We say that N realizes $\mathsf{Selector}(l)$ on variables $(a_1, a_2, \ldots, a_l, b_1, b_2, \ldots, b_l)$ if for $i, j = 1, 2, \ldots, l$,

$$f_{s_i,t_j} = \begin{cases} 0 & \text{if } i > j; \\ a_i \wedge b_j & \text{if } i \leq j. \end{cases}$$

Figure 1 shows directed networks, with and without 1's, realizing $\mathsf{Selector}(4)$. Indeed, we may obtain a network N computing T^n_k, similar to Markov's, by composing $k-1$ networks, each realizing $\mathsf{Selector}(n-k+1)$. In the composition, the ith network N_i, $i = 1, 2, \ldots, k-1$, realizes $\mathsf{Selector}(n-k+1)$ on variables $(x_i, x_{i+1}, \ldots, x_{n-k+i}, x_{i+1}, x_{i+2}, \ldots, x_{n-k+i+1})$. To obtain the threshold network, collapse the vertices $s_1(N_1), s_2(N_1), \ldots, s_{n-k+1}(N_1)$ into one and

Figure 1: Directed networks realizing $\mathsf{Selector}(4)$

call the resulting vertex $s(N)$, identify corresponding vertices of adjacent networks, that is, identify $t_i(N_j)$ with $s_i(N_{j+1})$ for $i = 1, 2, \ldots, n-k+1$ and $j = 1, 2, \ldots, k-2$, and collapse the vertices $t_1(N_{k-1}), t_2(N_{k-1}), \ldots, t_{n-k+1}(N_{k-1})$ into one vertex and call the resulting vertex $t(N)$. Thus, to obtain small networks computing T_k^n, it is sufficient to obtain small networks that realize $\mathsf{Selector}(n)$. Such a realization is possible using a simple recursive construction suggested by the example in Figure 1. We omit the details.

Lemma 1 *For all $n \geq 1$, there exists a directed contact network that realizes* $\mathsf{Selector}(n)$ *with size at most* $(n+1)\lceil\log(n+1)\rceil$. ∎

Theorem 1 *For all n and k, $2 \leq k \leq n-1$, there exist directed monotone contact networks of size $(k-1)(n-k+2)\lceil\log(n-k+2)\rceil$ computing T_k^n.*

Proof. As observed earlier, a network computing T_k^n may be obtained by composing $(k-1)$ networks that realize $\mathsf{Selector}(n-k+1)$. By Lemma 1, there exist networks of size $(n-k+2)\lceil\log(n-k+2)\rceil$ that realize $\mathsf{Selector}(n-k+1)$. Hence, there exist directed monotone networks computing T_k^n with size at most $(k-1)(n-k+2)\lceil\log(n-k+2)\rceil$. ∎

3 Lower bound for undirected networks computing T_{n-1}^n

3.1 Preliminaries

We now restrict our attention to undirected networks. We allow the constant 1 to appear as label in our networks. It is easy to see that edges labelled 1 can be contracted without altering the function computed by the network. However, for ease in presentation, we shall often let them remain, remembering not to consider them when estimating the size of the network.

We shall use the following graph theoretic notation. In a graph G, a path from vertex v to vertex w will be called a (v, w)-path. For a contact network N, a set $S \subseteq E(N)$ is called a (v, w) *edge cutset* if there is no (v, w)-path in the graph obtained from N by deleting the edges in S. Extending this notation, we shall refer to a set $C \subseteq \text{vars}(N)$ as a (v, w) *variable cutset* if the set consisting of all the edges whose labels appear in C forms a (v, w) edge cutset. When the nature of the cutset is clear from the context, we shall omit the prefixes edge and variable and just refer to the set as a (v, w)-cutset. We say that a path P in N *avoids* a variable $x \in \text{vars}(N)$ if P contains no edge with label x.

3.2 Overview

The function T_{n-1}^n satisfies two conditions. First, it accepts all inputs with $(n-1)$ 1's. Second, it rejects all inputs with $(n-2)$ 1's. For monotone networks, these two conditions contribute to the difficulty of the task of computing T_{n-1}^n. It turns out that we may relax the second condition considerably without making the task too easy. For example, in the case of formulas, the following holds [Kr]:

> Any formula that accepts all inputs with $(n-1)$ 1's and rejects all inputs with $(n-k)$ or fewer 1's, for $k \geq 2$, has size at least $n\log(n/(k-1))$.

This motivates the following definitions. Assume that $n \geq k \geq 2$.

- We call G an (n, k)-network if G has n variables, G accepts all inputs with at least $(n-1)$ 1's, and G rejects all inputs with at least k 0's.

For example, a network computing T_{n-1}^n is an $(n, 2)$-network. The fact about formulas can then be stated as follows: *Every (n, k)-formula ((n, k) series-parallel network) such that $n \geq 2^\tau(k-1)$, has size at least $n\tau$.* We would like to prove a similar statement for contact networks. First we need a definition.

- A network G is *τ-robust* if G is an (n, k)-network for some (n, k) such that $n \geq k^{2^\tau}$.

We shall show that every τ-robust network on n variables has size $cn\log\tau$ for some constant c. Since a network computing T_{n-1}^n is an $(n, 2)$-network, this would imply that

$$\text{size}_{\text{U}}(T_{n-1}^n) = \Omega(n\log\log\log n). \tag{1}$$

For $x \in \text{vars}(G)$, let $\mu_G(x)$ be the number of occurrences of x in G. Let $\mu(G)$ be the maximum multiplicity of a variable in G. That is,

$$\mu(G) = \max_{x\in\text{vars}(G)} \mu_G(x).$$

Let

$$l(\tau) = \min_{G: G \text{ is } \tau\text{-robust}} \mu(G).$$

We shall show that

$$l(\tau) \geq \log_3(\tau). \tag{2}$$

Then (1) will follow from the following lemma, which is an application of an observation due to Krichevskii [Kr]. We omit its proof.

Lemma 2 *If G is a τ-robust network with n variables, then G has size at least $(l(\tau)-1)n$.* ∎

Our proof of inequality (2) uses induction on τ. We show that every τ-robust network contains a $(\tau/3)$-robust subnetwork M. Moreover, this $(\tau/3)$-robust network will not be optimal, that is, $\mu(M) \geq l(\tau/3)+1$. It will then follow that $l(\tau) \geq \log_3(\tau)$. To obtain the non-optimal network M from N, we need to perform certain operations on N. We now describe these operations informally without giving the details. We shall repeatedly make use of the following simple facts about (n,k)-networks.

Observation 3 *1. N is an (n,k)-network if and only if the following two conditions hold.*

- *For every variable $x_i \in vars(N)$, there is an (s,t)-path in N avoiding x_i.*
- *Every subset of $vars(N)$ with k variables is an (s,t)-cutset.*

2. If N is an (n,k)-network and $S \subseteq vars(N)$ then the network obtained from N by setting all the variables in S to 1 is an $(n-|S|,k)$-network with variable set $vars(N)-S$.

In a network computing T^n_{n-1} no (s,t)-path can avoid both x_1 and x_2. Since we are dealing with (n,k)-networks, this useful property is not available to us automatically. However, with some reduction in the number of variables this property can be acquired. We shall show how a networks N_1 (say) can be obtained from N, for all $x_j \in \mathrm{vars}(N_1)-\{x_1\}$, no (s,t)-path in N_1 can avoid both x_1 and x_j. We call such an occurrence of x_1 *critical.*

Next, we obtain a network N_2 in which the variable x_1 appears only once. The reason this network is useful is as follows. Suppose the edge on which x_1 appears is (A,B). Since every path avoiding any other variable contains x_1, such paths must use the edge (A,B). Assume that they all use the edge in the direction, A to B. We focus on the subnetwork between s and A and the subnetwork between B and t. Now, if there are short paths (containing few variables) from s to A and from B to t, then there would be a short path from s to t. But this would contradict the robustness property of N_2. Hence, one of the two subnetworks must be reasonably robust. Also, as discussed above, for every variable they contain a path avoiding that variable. So one of these two networks will give us the desired subnetwork M. Then, it only needs to be shown that this subnetwork is non-optimal. For this we observe that in the network N_2 there is a path avoiding x_1. Now since x_1 is critical this path must include every other variable. It turns out that these occurrences are not essential for the computation of our subnetwork M. This implies that the network M is not optimal. The argument for this is discussed in section 3.4, when the 'special paths' are used. This is the only place where we use the fact that our network is undirected.

The details for the various steps are presented below. We shall use these steps when we prove the main lemma (Lemma 10, section 3.4). However, we suggest that the reader go directly to section 3.4 and obtain an overview of the argument presented there before verifying the details for the intermediate steps.

3.3 The intermediate steps

Making x_1 critical.

Lemma 4 *Let G be an (n,k)-network. Then, there is an (m,k)-network H, with $m \geq n/(k-1)$, such that*

(a) *$\mu(G) \geq \mu(H)$; and*

(b) *$x_1 \in vars(H)$, and for every variable $x_j \in vars(H)-\{x_1\}$, no (s,t)-path avoids both x_1 and x_j.*

Proof. We use induction on k. For $(k=2)$, we have that G computes T^n_{n-1}. If x_1 is not one of the variables of G then rename some variable as x_1. Note that this network now has property (b). Hence we take H to be the network G itself with some variable renamed as x_1 if necessary.

For the induction step, assume that the lemma holds for $k=r\geq 2$; we shall show that it holds for $k=r+1$. As before, if x_1 is not one of the variables of G, rename some variable of G as x_1. Let

$$S_1 = \{x_j \in \mathrm{vars}(G): \text{ every } (s,t)\text{-path avoiding } x_j \text{ contains } x_1\}.$$

Let $m_1 = |S_1 \cup \{x_1\}|$. Let H_1 be the network obtained from G by fixing all variables outside $S_1 \cup \{x_1\}$ at 1. Clearly, H_1 has properties (a) and (b), and H_1 is an $(m_1, r+1)$-network.

Let G' be the network obtained from G by fixing x_1 at 0 (deleting all edges with label x_1) and all variables in S_1 at 1. Clearly, $\mu(G') \leq \mu(G)$. We claim that G', with $\mathrm{vars}(G') = \mathrm{vars}(G)-(S_1\cup\{x_1\})$, is an $(n-m_1,r)$-network. To justify this, we use Observation 3. First, we show that for every variable $x_j \in G'$, there is an (s,t)-path in G' avoiding x_j. Since G is an (n,k)-network, there is an (s,t)-path in G avoiding x_j. Since $x_j \in \mathrm{vars}(G')$, $x_j \notin S_1$. By the definition of S_1, G has an (s,t)-path avoiding both x_1 and x_j. In G', this path may have some labels set on it to 1, but none to 0 (because it avoids x_1). Thus, this is an (s,t)-path in G' avoiding x_j. Next, we verify that every subset of vars(G') with r variables is an (s,t)-cutset of G'. Let C be a subset of vars(G') with r variables. Since G is an $(n, r+1)$-network, $C \cup \{x_1\}$ is an (s,t)-cutset in G. Hence, C is an (s,t)-cutset in G'. This completes the justification of our claim.

Since the statement holds when $k = r$, we get, by applying the induction hypothesis to the network G', an (m_2, r)-network H_2, such that

(a) $\mu(G') \geq \mu(H_2)$; and

(b) $x_1 \in$ vars(H_2), and for every variable $x \in$ vars(H_2)$-\{x_1\}$, no (s,t)-path avoids both x_1 and x.

Further $m_2 \geq (n - m_1)/(r-1)$. It follows that $m_2(r-1) + m_1 \geq n$. That is, $r \max\{m_2, m_1\} \geq n$ or $\max\{m_2, m_1\} \geq n/r$. Hence, one of H_1 and H_2 meets the requirements for H in the statement of the lemma. ∎

To just one x_1 Next we obtain a network in which x_1 appears only once.

Lemma 5 *If G is an (n,k)-network where $\{x_1, x_j\}$ is an (s,t)-cutset for every $x_j \in vars(G) - \{x_1\}$, then there is an (m,k)-network H with $m \geq \frac{n-1}{2\mu(G)} + 1$, such that*

(a) *there is only one edge (A, B) with label x_1; and*

(b) *for $x_j \in vars(H) - \{x_1\}$, every (s,t)-path avoiding x_j has the form $p \cdot (A,B) \cdot q$.*

Proof. First we obtain a network H' that has property (a). H' will be an (n', k)-network with $n' \geq \frac{n-1}{\mu(G)} + 1$.

Let the edges with label x_1 in G be $e_1, e_2, \ldots, e_r$, where $r = \mu(x_1) \leq \mu(G)$. For $x_i \in$ vars(G) $- \{x_1\}$, let $g(x_i)$ be the smallest index j such that there is an (s,t)-path avoiding x_i that contains none of $e_{j+1}, \ldots, e_r$. (Every such path must use e_j.) That is, for $x_i \in$ vars(G) $- \{x_1\}$,

$$g(x_i) = \min\{j : \text{there exists an } (s,t)\text{-path avoiding } x_i \text{ and } e_{j+1}, \ldots, e_r\}.$$

For every variable $x_i \in$ vars(G) $- \{x_1\}$, there exists an (s,t)-path in G that avoids x_i. Because $\{x_1, x_i\}$ is a cutset, each such path must contain at least one of $e_1, \ldots, e_r$. Hence $g(x_i)$ is well defined for $x_i \in$ vars(G) $- \{x_1\}$. Let

$$V_j = \{x_i \in \text{vars}(G) - \{x_1\} : g(x_i) = j\}, \text{ for } j = 1, \ldots, r.$$

Since $\bigcup_{j=1}^{r} |V_j| = n - 1$, there is a j such that $|V_j| \geq (n-1)/r$. For this j, let H' be the network obtained from G by setting all variables NOT in $\{x_1\} \cup V_j$ to 1, the edges $e_1, \ldots, e_{j-1}$ to 1 and deleting the edges $e_{j+1}, \ldots, e_r$. Clearly, the edge e_j ($= (A, B)$, say) is the only edge with label x_1. We now verify, using Observation 3, that H' is an $(|V_j| + 1, k)$-network. First, we show that for every $x_i \in$ vars(H') there is an (s,t)-path in H' avoiding x_i. There is such a path in G that does not use $e_{j+1}, \ldots, e_r$. Hence, there is an (s,t)-path in H' avoiding x_i. (Note that the (s,t)-path in G avoiding x_1 does not use any of $e_1, e_2, \ldots, e_r$, and therefore continues to serve in H'.) By our definition of V_j, for every $x_i \in V_j$, the path in H' avoiding x_j must use the edge e_j. Hence, $\{x_1, x_i\}$ is an (s,t)-cutset for all $x_i \in V_j$. Every subset S of V_j with k variables is an (s,t)-cutset of G. Since no occurrence of any variable in S has been replaced by 1, S is an (s,t)-cutset for H' also. It follows that every subset of vars(H') with k variables is an (s,t)-cutset in H'. Thus, we have verified that H' is an $(|V_j| + 1, k)$-network.

To obtain the network H satisfying the condition (b) we observe that for an $x_j \in$ vars(H') $- \{x_1\}$, all (s,t)-paths in H' avoiding x_j have either the form $p \cdot (A,B) \cdot q$ or the form $p \cdot (B,A) \cdot q$, but not both. For at least half the variables in vars(H') $- \{x_1\}$, the path must have the same form. By fixing the remaining variables in vars(H') $- \{x_1\}$ at the value 1 (invoking Part 2 of Observation 3), and suitably renaming A and B, we obtain an (m,k)-network with properties (a) and (b) such that $m \geq \frac{n-1}{2\mu(G)} + 1$. ∎

Getting special paths We need the following definitions. We call a network N *optimal* if

$$\mu(N) = \min_{H : H \text{ computes } f_N} \mu(H).$$

For a network N and a variable $x \in$ vars(N), we say that N is *x-optimal* if through every edge e with label x there is an (s,t)-path on which the label x appears only at e. First we make the following easy observations.

Observation 6 *1. If N is not x-optimal then there is an edge in N whose label x can be replaced by 1 without changing f_N.*

2. If N is not x-optimal for some variable $x \in vars(N)$ and we replace some occurrence of some other variable by 1, then the new network is also not x-optimal.

Lemma 7 *If N is optimal then N is x-optimal for some $x \in vars(N)$.*

Proof. Suppose N is not x-optimal for any $x \in \text{vars}(N)$. By repeatedly using Observation 6, we can replace one occurrence of each label $x \in \text{vars}(N)$ by 1, without changing f_N. But then, the final network obtained would contradict the optimality of N. ∎

We use the following fact that relates (s,t)-paths and (s,t)-cutsets in undirected graphs.

Observation 8 *In an undirected graph, every (s,t)-path must include an* odd *number of edges from every minimal (s,t)-cutset.*

Lemma 9 *Suppose G is an (n,k)-network such that*

(a) *$x_1 \in vars(G)$ appears only once;*

(b) *for the variable $x_j \in vars(G) - \{x_1\}$, G is x_j-optimal, and every path avoiding x_j contains the x_1.*

Then there is an (s,t)-path in G avoiding x_1 and containing only one edge labelled x_j.

Proof. Property (b) implies that, the edges with labels x_1 and x_j form an (s,t)-cutset. Let C_j be a minimal such cutset. Note that C_j must include an edge e with label x_j and the lone edge with label x_1. Since G is x_j-optimal, there is an (s,t)-path on which x_j appears only at e. This path cannot contain x_1, for then it would contain exactly *two* edges from the minimal cutset C_j, contradicting Observation 8. ∎

3.4 The main lemma and the theorem

Lemma 10 $l(\tau) \geq \lfloor \log_3(\tau) \rfloor$.

Proof. Since $l(\tau)$ is a non-decreasing function of τ, it is enough to verify the claim for all τ's that are powers of 3. For $\tau = 1$, the claim is trivial. Hence, assume that τ (≥ 3) is a power of 3 and the claim holds for all smaller powers of 3. Let G be τ-robust, that is, G is an (n,k)-network with $n \geq k^{2^\tau}$. We shall show that $\mu(G) \geq \log_3(\tau)$. It will follow that $l(\tau) \geq \lfloor \log_3(\tau) \rfloor$.

Making x_1 critical: Using Lemma 4 we get an (n',k)-network G', with

$$n' \geq n/(k-1), \tag{3}$$

such that

(1a) $\mu(G) \geq \mu(G')$; and

(1b) $x_1 \in \text{vars}(G')$, and for every variable $x_j \in \text{vars}(G') - \{x_1\}$, no (s,t)-path avoids both x_1 and x_j.

To just one x_1: From G', using Lemma 5, we get an (m,k)-network H, with

$$m \geq \frac{n'-1}{2\mu(G')} + 1 \geq \frac{n-k+1}{2\mu(G)(k-1)} + 1, \tag{4}$$

such that

(2a) there is only one edge (A,B) with label x_1; and

(2b) for every $x_j \in \text{vars}(H) - \{x_1\}$, every (s,t)-path avoiding x_j has the form $p \cdot (A,B) \cdot q$.

We assume that H is a minimal such network, that is, we cannot replace any variable label by 1 and retain these properties. Then Part 1 of Observation 6 implies that H has the following property.

(2c) H is x_j-optimal for every $x_j \in \text{vars}(H)$.

Since H is an (m,k)-network, properties (2a) and (2b) imply that the following holds for H.

(2d) For $x_j \in \text{vars}(H) - \{x_1\}$, there is an (s,A)-path p_j (and a (B,t)-path q_j) avoiding both x_1 and x_j.

Using special paths: Let H_1 be the network obtained from H by deleting the lone edge (A,B) with label x_1, setting $s(H_1) = s(H)$ and setting $t(H_1) = A$. Let H_2 be the network obtained from H by deleting (A,B), setting $s(H_2) = B$ and setting $t(H_2) = t(H)$.
Claim: H_1 and H_2 are not optimal.
Proof of claim. We shall only prove that H_1 is not optimal. That H_2 also is not optimal can be obtained by symmetry. Suppose H_1 is optimal. Using Lemma 7, we conclude that for some variable $x_i \in \text{vars}(H_1)$, H_1 is x_i-optimal. Since H has properties (2a), (2b) and (2c), we get from Lemma 9 that there is an edge e and an (s,t)-path in H avoiding x_1 on which x_i appears only at e. Let the path be $P \cdot (a,b) \cdot Q$, where $e = (a,b)$. Since H_1 is x_i-optimal, there is an (s,A)-path in H_1 on which x_i appears only at e. Suppose this path is of the form $p \cdot (a,b) \cdot q$. Using property (2d) we get that $p_i \cup q \cup Q$ contains an (s,t)-path in H avoiding both x_1 and x_i. This contradicts (2b). On the other hand if the path has the form $p \cdot (b,a) \cdot q$, then $p \cup Q$ contains such a path, again contradicting (2b). (End of Claim.)

The induction: Assume that

$$\mu(G) < \log_3(\tau). \tag{5}$$

We shall show that either $\mu(H_1) \geq \log_3(\tau/3)$ or $\mu(H_2) \geq \log_3(\tau/3)$. Since by the claim above these networks are not optimal, we will get that $\mu(G) \geq \log_3(\tau/3) + 1 = \log_3(\tau)$, contradicting (5).

Suppose no (s, A)-path in H_1 avoids $k^{2^{\tau/3}}$ variables from $\mathrm{vars}(H_1)$. Then every subset of $\mathrm{vars}(H_1)$ of size $k^{2^{\tau/3}}$ is an (s, A)-cutset in H_1. Also, (2d) implies that for every $x_j \in \mathrm{vars}(H_1)$ there is an (s, A)-path in H_1 avoiding x_j. It follows from Observation 3 that H_1 is an $(m-1, k^{2^{\tau/3}})$-network on the variables $\mathrm{vars}(H_1) = \mathrm{vars}(H) - \{x_1\}$, where (4) gives

$$m - 1 \geq \frac{n-k+1}{2\mu(G)(k-1)}.$$

Now,

$$\left(k^{2^{\tau/3}}\right)^{2^{\tau/3}} \leq \frac{n-k+1}{2\log_3(\tau)(k-1)},$$

since $k \geq 2$, $\tau \geq 3$ and $n \geq k^{2^\tau}$. Hence H_1 is $(\tau/3)$-robust. By the claim above, H_1 is not optimal. Hence, using the induction hypothesis, we have

$$\mu(G) \geq \mu(H) \geq \mu(H_1) \geq l(\tau/3) + 1 \geq \log_3(\tau),$$

contradicting (5).

On the other hand, suppose some (s, A)-path P in H_1 avoids $m' \geq k^{2^{\tau/3}}$ variables from $\mathrm{vars}(H_1)$. Let V be the set of these variables. In this case, we consider the network H_2. Let $\hat{H}$ be the optimal network computing f_{H_2}. By the claim above, H_2 is not optimal. Therefore, we have

$$\mu(H_2) \geq \mu(\hat{H}) + 1. \qquad (6)$$

Consider the network H' with $\mathrm{vars}(H') = V$ obtained from $\hat{H}$ by setting all variables not in V to 1. We now show that H' is an (m', k)-network. Let S be a subset of V of with k variables. Since H is an (m, k)-network, S is an (s, t)-cutset in H. Since S does not contain any variables in P and $x_1 \notin S$, S must be a (B, t)-cutset in H. It follows that S is also a (B, t)-cutset in H_2 and hence an $(s(\hat{H}), t(\hat{H}))$-cutset in $\hat{H}$. Also, (2d) implies that for every variable $x_j \in V$, there is a (B, t)-path in H_2 avoiding x_j, and therefore an $(s(\hat{H}), t(\hat{H}))$-path in $\hat{H}$ avoiding x_j. Thus, using Observation 3, it follows that H' is an (m', k)-network. Since $m' \geq k^{2^{\tau/3}}$, H' is $(\tau/3)$-robust. Then, using (6), we have

$$\mu(G) \geq \mu(H) \geq \mu(H_2) \geq \mu(\hat{H}) + 1 \geq \mu(H') + 1 \geq l(\tau/3) + 1 \geq \log_3(\tau),$$

again contradicting (5).

Thus (5) cannot hold and the induction is complete. ∎

Theorem 2 $\mathrm{size}_{\mathrm{U}}(T^n_{n-1}) = \Omega(n \log\log\log n)$.

Proof. The theorem is a direct consequence of Lemma 2, Lemma 10 and the fact that any network computing T^n_{n-1} is $(\log\log n)$-robust. ∎

Acknowledgment

Shiva Chaudhuri, Vivek Gore, and Rakesh Sinha read the earlier versions of this paper carefully, pointed out several errors and suggested many improvements. We are grateful to them for their contributions.

References

[B] R. Boppana. *Amplification of Probabilistic Boolean Formulas.* Advances in Computing Research Vol. 5 (S. Micali ed., JAI Press, Greenwich, CT), 1989 pp. 27–45.

[BS] R. B. Boppana and M. Sipser. *The Complexity of Finite Functions.* Chapter 14, The Handbook of Theoretical Computer Science, (J. van Leeuwen, ed.), Elsevier Science Publishers B. V., 1990, pp. 759–804.

[DZ] M. Dubiner and U. Zwick. *Amplification and Percolation.* In the Proceedings of the 33rd IEEE FOCS, 1992, pp. 258–267.

[GS] M. Grigini and M. Sipser. *Monotone Complexity.* Boolean Function Complexity. LMS Lecture Note Series 169, Cambridge University Press, 1992, pp. 55–75.

[H64] G. Hansel. *Nombre minimal de contacts de fermature nécessaires pour réaliser une fonction booléenne symétrique de n variables.* C. R. Acad. Sci. Paris 258 (1964), pp. 6037–6040.

[H56] M. A. Harrison. *Introduction to Switching and Automata Theory.* McGraw-Hill Series in Systems Science, 1956.

[K] M. Karchmer. *Communication Complexity: A New Approach to Circuit Depth.* The MIT Press, 1989.

[KW] M. Karchmer and A. Wigderson. *Monotone circuits for connectivity require super-logarithmic depth.* SIAM Journal of Discrete Mathematics 3 (1990) pp. 255–265.

[Kr] R. E. Krichevskii. *Complexity of contact circuits realizing a function of logical algebra.* Sov. Phys. Dokl. 8 (1964) pp. 770–772.

[L] O. B. Lupanov. *On comparing the complexity of realization of monotone contact networks containing only closing contacts and by arbitrary contact networks.* Sov. Phys. Dokl. 7 (1962) pp. 486–489.

[MS] E. F. Moore and C. E. Shannon. *Reliable circuits using less reliable relays.* J. Franklin Inst. 262, 1956, pp. 191-208 and pp. 281–297.

[M] A. A. Markov. *On minimal switching-and-rectifier networks for monotone symmetric functions (in Russian).* In Problems of Cybernetics, vol. 8, pp. 117–121, Nauka, 1962.

[R] J. Radhakrishnan. *Better Bounds for Threshold Formulas.* In the proceedings of the 32nd IEEE FOCS, 1991, pp. 314–323.

[Rz] A. A. Razborov. *Lower Bounds for Deterministic and Nondeterministic Branching Programs.* Proc. of the 8th FCT, LNCS 529, Springer-Verlag, pp. 47–60.

[W] I. Wegener. *The Complexity of Boolean Functions.* Wiley-Teubner Series in Computer Science, 1987.