The Effect of Random Restrictions on Formula Size

Russell Impagliazzo* Noam Nisan†

November 3, 1992

Abstract

We consider the formula size complexity of boolean functions over the base consisting of $\wedge$, $\vee$, and $\neg$ gates. In 1961 Subbotovskaya proved that, for any boolean function on n variables, setting all but a randomly chosen ϵn variables to randomly chosen constants, reduces the expected complexity of the induced function by at least a factor of $\epsilon^{1.5}$. This fact was used by Subbotovskaya to derive a lower bound of $\Omega(n^{1.5})$ for the formula size of the parity function, and more recently by Andreev who exhibited a lower bound of $\Omega(n^{2.5}/\log^{O(1)}(n))$ for a function in P.

We present the first improvement of Subbotovskaya's result, showing that the expected formula complexity of a function restricted as above is at most an $O(\epsilon^{\gamma})$ fraction of its original complexity, for $\gamma = (21 - \sqrt{73})/8 = 1.556....$

This allows us to give an improved lower bound of $\Omega(n^{2.556\cdots})$ for Andreev's function. At the time of discovery, this was the best formula lower bound known for any function in NP.

1 Introduction

When considering the complexity of a boolean function, it is natural to consider the properties of restrictions of the function, i.e., functions obtained by assigning values to a subset of the variables. In particular, it is often useful to examine the outcome of a restriction chosen at random from some distribution, and study the complexity of the induced function on the remaining variables.

This approach has achieved remarkable success in the study of constant depth circuits. Both the first and the strongest super-polynomial bounds for such circuits were proven by studying the effect of random restrictions on the circuit depth (see [3] for a survey). Much earlier, some success had been obtained by this approach concerning the formula size of boolean functions. When discussing formula size, one must be careful to specify the basis, since the formula size complexity may vary (by a polynomial) over different bases. In this paper, we only consider the DeMorgan basis, which consists of $\wedge, \vee$ and $\neg$ gates.

Subbotovskaya [4] observed in 1961 that, for *any* (non-trivial) boolean function f on n variables, assigning random values to all but a randomly chosen subset of variables of size ϵn, reduces the formula complexity of the function by at least an expected $\epsilon^{1.5}$ factor. She used this to prove a lower bound of $\Omega(n^{1.5})$ on formula complexity of the parity function. (Later, [5] used different methods to obtain a tight $\Theta(n^2)$ bound for parity.) Andreev [2] found a clever way of using Subbotvskaya's result to prove an even better bound, $\Omega(n^{2.5-o(1)})$, for an artificially constructed problem in P. The function used by Andreev has linear size circuits, and an obvious formula construction for the function yields an $O(n^3)$ upper bound.

Here, we consider the question of whether Subbotvskaya's exponent, 1.5, is indeed the best exponent describing the expected reduction in formula size under a random restriction. Let $L(f)$ be the minimum size DeMorgan formula computing a Boolean function f with n Boolean inputs. Let f^ϵ represent the random variable given by restricting $(1-\epsilon)n$ inputs to f uniformly chosen to uniformly chosen, independent elements of $\{0,1\}$.

Definition 1 *The* shrinkage exponent γ *for DeMorgan formulas is the least upper bound of the set of real numbers δ so that there are constants $A, B > 0$ so that for any Boolean function f and all $\epsilon > 0$,*

$$EXP[L(f^\epsilon)] \leq A\epsilon^\delta L(f) + B.$$

Since the shrinkage constant does not depend on the function computed, it is an intristic property of the computation model, the DeMorgan formulae size. Su shows that $\gamma \geq 1.5$, while the best upper bound we know of is the trivial:

Proposition 1 $\gamma \leq 2$.

This can be seen by the fact that the parity function has formulaes of size $\Theta(n^2)$. The question of the exact value of γ remains open. We conjecture that

Conjecture 1 $\gamma = 2$.

In this paper we show that the $\gamma \geq 1.5$ bound given by [4] is not tight and can be improved. We show that

Theorem 1 $\gamma \geq \frac{21-\sqrt{73}}{8} = 1.556\ldots$

This gives an improvement in the lower bound for Andreev's function and allows us to give the best lower bound known so far for formulae size. Let A_n denote the function over n variables defined in [2].

Theorem 2 *DeMorgan formulas for A_n require size $\Omega(n^{2.556\ldots})$.*

Recently, Paterson and Zwick [1] have further improved the lower bound on γ to $\gamma \geq (5-\sqrt{3})/2 = 1.63\ldots$, and hence also obtain a better lower bound for Andreev's function. Their proof combines ideas used here with some clever new techniques. We feel that small further improvements on this lower bound using these techniques should be possible by doing more extensive case analysis, but it is not at all clear that this will lead to matching upper and lower bounds. To prove or disprove our conjecture that $\gamma = 2$ might require a new approach.

2 Notation

DeMorgan Formulas

A formula over inputs $x_1, x_2, \ldots x_n$, is a tree whose nodes are labelled $\wedge$ ("and") or $\vee$ ("or") and whose leaves are labelled with literals (either an input or the negation of an input.) Such a formula computes a boolean function on its inputs in a natural way. The size of a formula F is its number of leaves. For a boolean function f, $L(f)$ is the least size of a formula computing f.

Restrictions of functions

Let the variables for f be $x_1, ..., x_n$. A literal for f has the form x_i or $\overline{x}_i$. The restriction f_{x_i} is the Boolean function defined by:

$$f_{x_i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n),$$

and the restriction $f_{\overline{x}_i}$ is that defined by:

$$f_{\overline{x_i}}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n).$$

f is called *independent* of x_i if $f_{x_i} = f_{\overline{x}_i}$. We say f is *vulnerable* to literal x if f_x is constant, and that f is *vulnerable* if it is vulnerable to some literal.

We will often use f' to denote the random variable f_x where x is a randomly chosen literal. We will denote by f^ϵ the random variable obtained by restricting all but randomly chosen ϵn variables to random values.

Restrictions of formulas

For a formula F computing a function f, and x a literal, let F_x be the following formula computing f_x. Replace all the occurences of x by the constant 1. Replace all occurences of $\overline{x}$ by 0. While possible, perform one of the following simplifications: replace any sub-formula of the form $0 \wedge (F')$ or $(F') \wedge 0$ by 0; replace any sub-formula of the form $0 \vee (F')$ or $(F') \vee 0$ by F'; replace any sub-formula of the form $1 \wedge (F')$ or $(F') \wedge 1$ by F'; replace any sub-formula of the form $1 \vee (F')$ or $F' \vee 1$ by 1. (This simplification is unique, but this fact is not needed for our purposes).

3 The value of γ, Subbotovskaya's Bound

For completeness, motivation, and because we will later use this result in our improvement, we present first Subbotovskaya's proof showing $\gamma \geq 1.5$.

Lemma 1 (Essentially due to [4]) $\gamma \geq 1.5$.

The lemma will follow directly from propositions 2 and 3.

Proposition 2 *Let f be any function on n variables, with $L(f) \geq 2$. Let f' be the result of fixing one input, at random, to a random value. Then $E[L(f')] \leq L(f) \cdot (1 - 1.5/n)$.*

Proof Let F be a minimal formula for f. Let x be any variable. We first observe that in a minimal formula there is never a leaf labelled with x or with $\overline{x}$ inside a neighbor tree of another leaf labeled with x or $\overline{x}$. The reason for this is that otherwise the leaf inside the neighbor tree could be replaced by a constant without changing the function computed, while reducing the formula size.

Consider any leaf labelled x or $\overline{x}$. If we fix x to either 0 or 1, we will be able to remove this leaf from F; for one of the two values, we will also be able to remove the neighbor tree for the leaf, removing at least one other leaf. Now, since no leaf labelled x appears in the neighbor set of such a leaf, the above simplifications all remove disjoint sets of leaves. If we add the total simplifications for setting x to 1 to those for setting x to 0, we get at least 3$\cdot$(number of leaves labelled x). Thus, if we take the sum over all variables of the total simplifications, it will total at least 3$\cdot$ (number of leaves of F)=$3 \cdot L(f)$. This gives a lower bound for the sum over all possible values of f' of $L(f) - L(f')$, so the expected difference will be at least this bound divided by the $2n$ possibilities for f', $3L(f)/(2n)$. This gives the proposition. □

Proposition 3 *Let L be a complexity measure of boolean functions that for every boolean function f on n variables satisfies $E[L(f')] \leq (1 - c/n) \cdot L(f)$, where f' is the result of restricting a random variable to a random value. Then*

$$E[L(f^\epsilon)] \leq O(\epsilon^c L(f))$$

.

Proof Let $k = \epsilon n$ and let $p_i = 1 - \frac{c}{(n-i+1)}$. Think of f^ϵ as being generated by picking one variable at random and setting it to a random value, obtaining a function f_1, and repeating this process $n-k$ times. By applying the assumption on L to f_i and by linearity of expectation, $E[L(f_i)] \leq p_i \cdot E[L(f_{i-1})]$. We thus have

$$\frac{E[L(f^\epsilon)]}{L(f)} \leq \left(1 - \frac{c}{n}\right) \cdot \left(1 - \frac{c}{n-1}\right) \cdots \left(1 - \frac{c}{k+1}\right) \leq O((k/n)^c),$$

where the final inequality is a simple excersize. □

4 The improved bound on γ

We claim that the bound given in lemma 1 can be strengthened. The intuition as to why this is so is that Proposition 2 fails to take into account the possibility that the neighbor tree of a leaf might have more than one leaf. In this case, killing the variable will reduce the size of the formula by more than 1.5 on the avarage.

Unfortunately, this possibility does not necessarily occur. Consider a balanced "and-or" tree, i.e. a formula whose tree structure is a balanced binary tree, whose leaves are distinct variables, and whose internal nodes are labelled $\wedge$ or $\vee$ according to whether their depth is even or odd. In this case, proposition 2 is tight; the expected simplification obtained by fixing any variable to 0 or 1 randomly will be exactly 1.5. However, doing so will create imbalances in the tree, making future "unexpected bonuses" more likely.

As we shall show, this situation (having a leaf who's neighbor tree is more than a leaf) will occur in significant amounts during the process of randomly restricting a large number of inputs to any formula. This is because, except in certain unusual circumstances, a random restriction to a leaf not in such a situation has a 50% chance of creating such a situation. To make the proof simpler, we introduce a potential function on formulas which, in effect, gives us "advance credit" for such situations as they are created.

Definition 2 *Let $\alpha = (9-\sqrt{73})/4$ and let $\delta = 1.5+\alpha/2$. (The reasons these choices for α and δ are optimal will be made clear at the point of the proof when it arises. However, we will be using the fact that $\alpha < 1/8$ all along.)*

Let F be a formula. Call a leaf ℓ of F a killer *if its neighbor tree has at least two leaves. Call a leaf of F a* victim *if it is in the neighbor tree of a killer. Let the* weight *of a leaf of F be $1-\alpha$ if the leaf is a victim, and 1 otherwise. Let $W(F)$, the weight of F, be the sum of the weights of its leaves. For a function f, let $W(f)$ be the least weight of a formula computing f.*

Note that $(1-\alpha)L(f) \leq W(f) \leq L(f)$, so a bound on the weight is equivalent to within a constant factor to a bound on the size. Using proposition 3 on the measure W, in order to prove our main theorem it suffices to show:

Lemma 2 *Let f be any Boolean function on n variables with $L(f) \geq 5$. Let $f' = f_x$ for x a randomly chosen literal of f. Then, $E[W(f')] \leq W(f)(1 - \delta/n)$.*

The rest of this section is now devoted to the proof of lemma 2.

Let F be a minimum weight formula for f. We will divide the leaves of F up into sub-trees and show that each sub-tree contributes its share to the overall simplification of the formula. We now describe how to break F into sub-trees.

Definition 3 *A* disjoint cover *of a tree is a set of nodes so that each leaf is either a descendent of or equal to exactly one node in the disjoint cover.*

Definition 4 *Let F be a formula computing f. We say an interior node with sub-formula G computing function g is* special *if one of the following hold:*

- *1 g is vulnerable;*
- *2a $G = (G_1) \wedge (G_2)$, where one of the G_i's is of the form $x \vee y$ for literals x and y, and the other is vulnerable to both $\overline{x}$ and $\overline{y}$;*
- *2b $G = (G_1) \vee (G_2)$, where one of the G_i's is of the form $x \wedge y$ for literals x and y, and the other is vulnerable to both x and y;*
- *3 There are independent literals y, z and u so that $g = maj(y, z, u)$, where maj is the majority function on three inputs.*

We say a node is maximally special *if it is special and no ancestor of the node is special.*

Proposition 4 *The maximally special nodes of a formula form a disjoint cover.*

Proof The above follows from the fact that every leaf's parent node is special. $\square$

We want to look at local changes in the maximally special sub-formulas of F when restricted by a literal x, and show that they combine to give the required average reduction. A sub-formula may contribute to the reduction in the weight of the overall formula in two ways. First, we count any reduction in the cost of the sub-formula. Second, if the sub-formula simplifies to a single leaf, we might get further simplifications as follows.

If the leaf's neighbor tree is independent of the variable labelling the leaf, and that neighbor has at least 2 nodes, the leaf is now a killer, and has created at least one new victim, so we get at least an additional α reduction in weight.

If the leaf has a neighbor tree which depends on the variable labelling the leaf, we can replace all occurences of this variable by an appropriate constant. Again, there will be two cases. The canonical examples of these two cases are 1:$f = g \wedge h$, where g simplifes to x and h simplifies to x; and 2: $f = g \wedge h$ where g simplifies to x and h simplifies to $x \wedge y$. In the first case, we simplify $f = x \wedge x = x$ to get a savings of 1; however, several factors might reduce this savings. First, since both g and h simplified to leaves, we must split the credit for this savings between them. Secondly, the function f might appear as a subformula in the neighbor tree of a killer, multiplying the savings by $1-\alpha$. Lastly, simplifying f might cause this killer no longer to be a killer. A case of this is $y \wedge (x \wedge x) = y \wedge x$. Here, two leaves are used to get a savings of $1-2\alpha$, so the credit per leaf is $1/2(1-2\alpha)$. In the second case, we simplify $f = x \wedge (x \wedge y) = x \wedge (1 \wedge y) = x \wedge y$, and get a savings of $1-2\alpha$, since x is no longer a killer. In the proof, we will combine these cases, and only claim a least common denominator of $1/2(1-2\alpha)$ credit per occurrence of either situation.

The following Lemma shows that these simplifications do not conflict with each other. We will only be concerned with the case where the formula is derived from F by replacing each maximally special sub-formula which originally computed function g in F by a minimal weight formula for g_x for some literal x. However, to simplify notation, we will prove the lemma for a general formula, possibly with leaves labelled by constants. (This is to handle the case when a maximally special sub-formula reduces to a constant.) The function computed by such a formula is an obvious extension of the definition for all formulas.

The weight and size of such a formula are not so easily defined; however, we avoid mentioning the weight of the formula F computing function f by referring solely to $W(f)$, the minimum weight for a formula without constants computing f.

Lemma 3 *Let F be a formula possibly with constant leaves computing f and let $c_1, ...c_l, l \geq 1$ be a disjoint cover of F. Let g_i be the function computed at c_i and let h_i be the function computed by the sibling of c_i. (If $l = 1$, c_l is the root and has no sibling; in this case, we define h_l to be the constantly zero function.) Let $J = |\{1 \leq i \leq l : g_i = x$ for some literal x and $L(h_i) \geq 2$ and h_i is independent of $x\}|$. Let $K = |\{1 \leq i \leq l : g_i = x$ for some literal x, and h_i is not independent of $x\}|$. Then $W(f) \leq \sum_{1 \leq i \leq l} W(g_i) - \alpha J - 1/2(1 - 2\alpha)K$.*

Proof Replace the tree at each c_i with a minimal weight formula (without constants) for g_i. Then remove constants using the rules in the definition of the restriction of a formula. This gives us a formula F' computing f. We claim that, if h_i is not a constant function, and c_i remains in F', then the sub-formula rooted at c_i is the original minimal weight formula computing g_i, and that at its sibling computes h_i. We prove this by induction on the number of applications of one of the rules in the definition of the restriction of a formula. The rule always involves a leaf labelled by a constant, and its neighbor tree. The leaf is always removed, and the leaf's neighbor tree is either removed from the tree or moved to the leaf's parent. By the induction hypothesis, the leaf cannot be in the tree rooted at c_i, since it started without constants and has not changed. By the assumption that h_i is not constant, and by the induction hypothesis that h_i also has not changed, it cannot be the sibling of c_i. Since c_i is not removed from the formula, it cannot be c_i itself. This leaves three cases: either the leaf's parent is in c_i's neighbor tree, the leaf's parent is an ancestor of c_i's parent, or the leaf's parent and c_i's parent have some least common ancestor different from either one. In the first case, since the rule is valid, it does not change the function computed by the neighbor tree. In the second case, since c_i is not removed, the change is to promote one ancestor of c_i to the place occupied by its parent, which does not change either the function computed at c_i or at its sibling. In the third case, the change does not affect c_i at all. As a result, we have that $\{c_i | c_i \in F'\}$ is still a disjoint cover for F', since each leaf that is not removed is part of the minimal weight formula for some c_i that is not removed.

Let $J' = |\{i : c_i \in F', g_i = x$ for some literal x, $L(h_i) \geq 2$ and h_i is independent of $x\}|$.

We claim next that $W(F') \leq \sum_{i, c_i \in F'} W(g_i) - \alpha J'$. Pick, for every $c_i \in F'$ with $g_i = x$ for a single literal x, and $L(h_i) \geq 2$, a leaf of the sub-tree now rooted at the sibling of c_i which is not a victim in this sub-tree. Note that we have not selected any node twice, since if it was selected for one c_i, it would be a victim in any sub-tree including c_i. Note also that each leaf so selected has weight 1 in the sub-tree of its unique cover c_j, but weight $1-\alpha$ in F'. Finally, note that no leaf has a higher weight in F' than in the sub-tree rooted at its cover.

Let $K' = |\{c_i \in F' : g_i = x$ for some literal x, and h_i is not independent of $x\}|$.

We will show that we can find a formula computing f with weight at most $W(F') - 1/2(1-2\alpha)K'$.

Proposition 5 *Let F' be any formula computing f with $K' = |\{\ell$ a leaf in F'— the function computed by ℓ's neighbor tree depends on the variable labelling $\ell\}|$. Then $L(f) \leq L(F') - K'/2$ and $W(f) \leq W(F') - 1/2(1-2\alpha)K'$.*

Proof We first prove $L(f) \leq L(F') - K'/2$ by induction. Assume, without loss of generality, that $F' = G \wedge H$, where G computes g and H computes h. If neither G nor H is a literal that the other depends on, we can use the induction hypothesis on each. If one (say G) is such a literal, and the other is not, then use the induction hypothesis on H to get $L(h) \leq L(H) - (K'-1)/2$. Then, since h depends on $x = g$, $L(h_x) \leq L(h) - 1 < L(H) - K'/2$. Then $f = x \wedge h_x$ as can be seen by looking at the cases $x = 0$, and $x = 1$, so $L(f) \leq 1 + L(h_x) \leq 1 + L(H) - K'/2 = L(F') - K'/2$. Finally, if both H and G are such literals, $K' = 2$, $L(f) \leq 1 = L(F') - 1 = L(F') - K'/2$.

Next, we prove by induction that $W(f) \leq W(F') - 1/2(1-2\alpha)K'$. Again, assume $F = G \wedge H$, where G computes g and H computes h. If neither G nor H is a literal, then we simply apply the induction hypothesis to each.

If G is a literal x and H is not, and h does not depend on x, then we can apply the first claim to H to show $L(h) \leq L(H) - K'/2$. Now, $W(f) \leq 1 + (1-\alpha)L(h) + \alpha$. (The extra α is needed if and only if $L(h) = 1$.) So, if $K \geq 2$, we have $W(f) \leq 1 + (1-\alpha)(L(H) - K'/2) + \alpha = W(F) - 1/2(1-\alpha)K' + \alpha = W(f) - 1/2(1-2\alpha)K' - 1/2\alpha K' + \alpha \leq W(f) - 1/2(1-2\alpha)K'$. If $K' = 1$, we have $L(h) \leq L(H) - 1/2$, so $L(h) \leq L(H) - 1$, since both are integers. Then we have $W(f) \leq 1 + (1-\alpha)(L(H) - 1) + \alpha = W(F) - (1-2\alpha) = W(F) - (1-2\alpha)K'$. This also handles the symmetrical situation, reversing G and H. If G and H are independent literals, then $K' = 0$, so the claim is trivial. If G and H are dependent literals, $K' = 2$ and $W(f) \leq 1 = W(F') - 1 = W(F') - K'/2$.

The only case left is $g = x$ for some literal x where h depends on x (and the symmetrical situation, handled in the same way.) Applying the first claim to H, we have $L(h_x) \leq L(h)-1 \leq L(H)-1/2(K'-1)-1 = L(H)-1/2K'-1/2$. As before, $f = x \wedge h_x$, so $W(f) \leq 1+(1-\alpha)L(h_x)+\alpha \leq 1+(1-\alpha)(L(H)-K'/2)-(1-\alpha)1/2+\alpha \leq W(F)-(1-\alpha)K'/2$ since $\alpha \leq 1/3$

□

Returning to Lemma 3, we apply Proposition 5 to the formula F' to get $W(f) \leq W(F') - (1-2\alpha)1/2K' \leq \sum_{i|c_i \in F'} W(g_i) - \alpha J' - 1/2(1-2\alpha)K'$. Now, note that $J - J' + K - K' \leq |\{i|c_i \notin F', W(g_i) = L(g_i) = 1\}|$. So $Sum_{1\leq i \leq l} W(g_i) \geq \sum_{i|c_i \in F'} W(g_i) + J - J' + K - K'$. Hence, $W(f) \leq Sum_{1 \leq i \leq l} W(g_i) - \alpha J - 1/2(1-2\alpha)K$ as claimed. □

We now prove Lemma 2. Let f be a Boolean function and F a minimum weight formula computing f. For x a literal for f, let $\Delta_x = W(f) - W(f_x)$. Lemma 2 is equivalent to:

$$\sum_x \Delta_x > (2\delta)W(f) = (3+\alpha)W(f)$$

.

where the sum is over all $2n$ literals.

Let $c_1, ...c_l$ be the maximally special nodes in F. Let G^i be the sub-formula rooted at c_i, H^i the sub-formula rooted at the sibling of c_i, g^i be the function computed by G^i, and h^i the function computed by H^i. (Note: In the case, l=1, c_1 is the root of f and we define H^1 to be the empty formula and h^1 to be the constant 0 function.) Since no node above a maximally special node is vulnerable, no maximally special node is in the neighbor tree of a killer, so we have $W(f) = W(F) = \sum_{1 \leq i \leq l} W(G^i) = \sum_{1 \leq i \leq l} W(g^i)$.

We show that each maximally special sub-formula contributes its share to the overall reduction in the weight of F, where the contribution is as given by Lemma 3. More precisely,

Definition 5 *Let $\Delta^i_x = W(g^i) - W(g^i_x) + 1/2(1-2\alpha)$, if g^i_x is a literal y so that h^i_x depends on y, $W(g^i) - W(g^i_x) + \alpha$ if g^i_x is a literal y so that h^i_x is independent of y and $L(h^i_x) \geq 2$, and $W(g^i) - W(g^i_x)$ otherwise.*

From Lemma 3, we have:

$$W(f_x) \leq \sum_{1 \leq i \leq l} W(g^i_x) - \sum_{1 \leq i \leq l} \Delta^i_x - (W(g^i) - W(g^i_x)) = W(f) - \sum_{1 \leq i \leq l} \Delta^i_x$$

, so $\Delta_x \geq \sum_{1 \leq i \leq l} \Delta_x^i$.

Therefore, to prove Lemma 2, it suffices to prove:

Proposition 6 *For each* $1 \leq i \leq l$, $\sum_x \Delta_x^i \geq (3+\alpha)W(g_i)$.

Before we get to the proof of proposition 6, we will need the following characteristic of minimum weight formulas:

Proposition 7 *Let G be a subformula of F, a minimum weight formula computing f so that G is not in the neighbor tree of a killer. If the function g computed by G is vulnerable, then one of the children of G's root is a leaf.*

Proof (of proposition 7) Without loss of generality, assume $g_x = 1$ for literal x. Assume neither child of G's root is a leaf. Then $L(g) \geq 2$. Now since $g_x = 1$, $g = x \vee g_{\overline{x}} = x \vee G_{\overline{x}}$. $W(x \vee G_{\overline{x}}) = 1 + (1-\alpha)L(G_{\overline{x}}) \leq 1 + (1-\alpha)(L(G)-1) \leq \alpha + (W(G) - 2\alpha)) = W(G) - \alpha$, since at least one leaf of G is labelled x or $\overline{x}$ and since at least two leaves have weight 1 in G. Since G is not in the neighbor tree of a killer, this implies that replacing G with $x \vee G_{\overline{x}}$ yields a smaller weight formula for f than F. □

We now prove Proposition 6.

Proof Since g^i is a special function, it must fall into one of the following catagories.

Case 1a: g^i is vulnerable and $L(G^i) \geq 3$.

By Proposition 7, and without loss of generality, we can assume $G^i = z \wedge G'$ for some literal z and formula G'. Since $z \wedge G' = z \wedge G'_z$, G' must be independent of z. Let g' be the funtion computed by G'. We have $W(g^i) = 1 + (1-\alpha)L(g')$. We further divide into the cases $L(g') = 2$ and $L(g') \geq 3$.

Intuitively, the case $L(g') \geq 3$ is a bonanza for us, since then setting z to 0 gives us a large savings. Let y be any literal that G' depends on. Then $W(g^i_y) \leq 1 + (1-\alpha)L({g'}_y) + \alpha$, the additional α only being necessary when $L({g'}_y) = 1$. In this case, $\Delta^i_y \geq (1-\alpha)(L(g) - L({g'}_y) - \alpha$. Also $g_z = g'$, so $\Delta^i_z = 1 - (W(g') - (1-\alpha)L(g')) \geq 1 - \alpha L(g')$. Finally, $g_{\overline{z}} = 0$, so $\Delta^i_{\overline{z}} = 1 + (1-\alpha)L(g')$. Thus,

$$\sum_x \Delta_x^i \geq 2 + (1-2\alpha)L(g') + (\sum_{g' \; depends \; on \; literal \; y} (1-\alpha)(L(g'_y) - L(g'))) - \alpha)$$

$$\leq 2 + (1-2\alpha)L(g') + (1-\alpha)(\sum_{g' \; depends \; on \; literal \; y} (L(g) - L({g'}_y)) - 2\alpha L(g').$$

Since $L(g') \geq 2$, we can apply Subbotovskaya's bound on g' to get that this is at least: $2 + (1-2\alpha)L(g') + 3(1-\alpha)L(g') - 2\alpha L(g') = 2 + (4-7\alpha)L(g')$. Using the inequalities $L(g') \geq 3$ and $\alpha \leq 1/8$, we easily calculate that $2 + (4-7\alpha)L(g') \geq (3+\alpha)(1+(1-\alpha)L(g')) = (3+\alpha)W(g^i)$.

If $L(g') = 2$, $W(g^i) = 3 - 2\alpha$, and g^i is either of the form $z \wedge u \wedge v$ for literals u and v, or of the form $z \wedge (u \vee v)$. In the former case, we see that $\sum_x \Delta^i_x = 3(4-4\alpha) = 12 - 12\alpha$. In the latter, we can compute $\Delta^i_z = 1 - 2\alpha$, $\Delta^i_{\overline{z}} = 3 - 2\alpha$, $\Delta^i_u = \Delta^i_v = 2 - 2\alpha$, and $\Delta^i_{\overline{u}} = \Delta^i_{\overline{v}} = 1 - 2\alpha$, for a total of $10 - 12\alpha$. Thus the first case is clearly better than the second, which is one of three limit cases for α. We chose α as the positive root of $10 - 12\alpha = (3+\alpha)(3-2\alpha)$, so the lemma holds in this case.

Case 1b: $L(G^i) = 2$.

Without loss of generality, assume $g^i = y \wedge z$ for literals y and z. $\sum_x \Delta^i_x = \Delta^i_y + \Delta^i_{\overline{y}} + \Delta^i_z + \Delta^i_{\overline{z}}$. Now $\Delta^i_{\overline{y}} = \Delta^i_{\overline{z}} = 2$. Since we assume $L(f) \geq 5$, c^i cannot be the root of F, so it has a parent and sibling, and h^i is not constant. Further, the parent must be labelled with an $\vee$, since otherwise its function would be vulnerable to $\overline{y}$. The normal case is that h^i is independent of y and z; since $L(h^i) \geq 2$ (or else c^i wouldn't be maximally special), this means that we get an additional α bonus when we set y or z to 1. Thus, each gives a $3+\alpha$ total contribution. This is the second limiting case for α. We show that if this is not the case, either we get at least this much in total contributions for other reasons, the original formula was not minimal, or else c^i's parent is special, being covered by one of the cases 2b or 3 of the definition of special sub-formulas.

Formally, we divide into sub-cases as follows. If both $L(h^i_y) \geq 2$ and $L(h^i_z) \geq 2$, then $\Delta^i_y, \Delta^i_z \geq 1+\alpha$, and we have $\sum_x \Delta^i_x \geq 6 + 2\alpha = (3+\alpha)W(g^i)$. If h^i_y depends on z, then $\Delta^i_y = 1 + 1/2(1-2\alpha) \geq 1 + 2\alpha$ (from $\alpha < 1/6$) and $\Delta^i_z \geq 1$. So again $\sum_x \Delta^i_x \geq 6 + 2\alpha$. Similarly if h^i_z depends on y.

If none of the above is true, then $h^i_y = (h^i_y)_z = (h^i_z)_y = h^i_z = h'$ for some h', and $L(h') \leq 1$. If h' is constant, then c^i is not maximally special, since its parent is special (Case 2b, since c^i's parent is labelled $\vee$).

If $L(h') = 1$, $h' = u$ for some literal u independent of y and z. We divide into 3 sub-cases: h^i is monotone increasing in y; h^i is monotone decreasing in y, or neither.

In the last subcase, we claim that the formula $F' = G^i \vee H^i$ computed at c^i's parent can be computed by a smaller weight formula. Let $f' = (z \wedge y) \vee h^i$

be the function computed by the parent. Then $f' = (y \wedge f'_y) \vee (\overline{y} \wedge f'_{\overline{y}}) = (y \wedge (z \vee u)) \vee (\overline{y} \wedge h^i_{\overline{y}})$. $h^i_{\overline{y}}$ depends on both u and z, since $(h^i_{\overline{y}})_z = u$. So $L(h^i_{\overline{y}}) \geq 2$ and $W(f') \leq 2 + (1-\alpha)(L(h^i_{\overline{y}}) + 2))$. Since h^i is not monotone in either y or $\overline{y}$, y must appear both positively and negated in any formula for h^i. Hence, $L(h^i_{\overline{y}}) \leq L(h_i) - 2$. Therefore, $W(F') = W(G^i) + W(H^i) \geq 2 + 1 + (1-\alpha)(L(h^i) - 1) = \alpha + 2 + (1-\alpha)L(h^i) \geq \alpha + W(f')$, so F' is not minimal. Since c^i's parent is not in the neighbor tree of any killer, this implies that the formula F is not minimal, contrary to assumption.

In the case that h^i is monotone increasing in y, we show that either c^i's parent is of the form covered by case 3 of the definition of special, or the formula F' rooted there is not minimal. We'll use the notation $f_1 \leq f_2$ to say that boolean function f_1 implies f_2. Then we have $h^i \leq h^i_y = u$, so $h^i_{\overline{u}} = 0$. Hence, $h^i = u \wedge h^i_u$. So $u = h^i_y = u \wedge (h^i_u)_y$, so $(h^i_u)_y = 1$. Similarly, $(h^i_u)_z = 1$. Hence, $h^i = u \wedge (y \vee z \vee ((h^i_u)_{\overline{y}})_{\overline{z}})$. Let $h'' = ((h^i_u)_{\overline{y}})_{\overline{z}}$. If h'' is constant, then the parent of c_i is special by case 3, since it computes $(y \wedge z) \vee (u \wedge (y \vee z)) = maj(y, z, u)$. Otherwise, consider the function $f' = (y \wedge z) \vee h^i = (y \wedge z) \vee (u \wedge (y \vee z \vee h''))$ computed at c_i's parent, and the formula $F' = G^i \vee H^i$ rooted there. Then $W(F') = W(G^i) + W(H^i) \geq 2 + 1 + (1-\alpha)(L(h^i) - 1) = 3 + (1-\alpha)((L(h'') + 2)$. However, f' is monotone in y, so $f' = (y \wedge f'_y) \vee f'_{\overline{y}} = (y \wedge (z \vee u)) \vee h^i_{\overline{y}} = (y \wedge (z \vee u)) \vee (u \wedge (z \vee h''))$. Thus, $W(f') \leq 2 + (1-\alpha)(L(h'') + 3) \leq W(F') - \alpha$. Therefore, F' is not a minimal formula, and thus neither is F, contrary to assumption.

Finally, if h^i is monotone decreasing in y , then $u = h^i_y \leq h^i \leq f'$ and $f'_u = 1$. . But then f' is vulnerable, so c^i is not maximally special.

Thus, all cases that do not yield the desired bound on $\sum_x \Delta^i_x$ do not arise in minimal formulas.

Case 2a: g^i is not vulnerable, and it is of the form: $(y \wedge z) \vee g'$, where g' is vulnerable to both y and z. Note that $g'_y = g'_z = 0$, because otherwise g^i is vulnerable.

The prototypical example for such a g^i is the parity function on two literals, which will be our last limiting case for α. However, the limit achieved will be exactly the same as for the case of $y \wedge z$.

Indeed, in the sub-case $L(g') = 2$, it is easy to see that $g^i = \overline{y} \oplus z$. In this case, since $L(f) \geq 5$, c^i cannot be the root of F, and so h^i is not constant. If $L(h^i_y), L(h^i_z), L(h^i_{\overline{y}})$, and $L(h^i_{\overline{z}})$ are all at least 2, then $\sum_x \Delta^i_x \geq 4(3+\alpha)$, which is what we desire, since $W(g^i) = 4$. Assume this does not occur. Then since $L(h^i) \geq 2$, h^i is dependent on at least one of y or z. If both of h^i_y and $h^i_{\overline{y}}$ are independent of z, then h^i is independent of z. Then h^i is dependent on y, and $h^i_z = h^i_{\overline{z}} = h^i$, so both are dependent on y. Similarly if both of h^i_z and $h^i_{\overline{z}}$ are independent of y, then both h^i_y and $h^i_{\overline{y}}$ are dependent on z. Thus, for two out of the four settings $y, \overline{y}, z, \overline{z}$ we have the corresponding Δ^i at least $3 + 1/2(1-2\alpha) > 3 + 2\alpha$, so $\sum_x \Delta^i_x \geq 12 + 4\alpha = 4(3+\alpha)$ as desired.

In the sub-case $L(g') \geq 3$, we have $g' = \overline{y} \wedge \overline{z} \wedge ((g')_{\overline{y}})_{\overline{z}}$. Let $g'' = ((g')_{\overline{y}})_{\overline{z}}$, $L = L(((g')_{\overline{y}})_{\overline{z}})$. Then $W(g') = 2 - \alpha + (1-\alpha)L$ and $W(g^i) = 2 + W(g') = 4 - \alpha + (1-\alpha)L$. In this case, we argue that the sums of the reductions in $W(g')$ more than compensate for a possible missing 2α in the reductions for $(y \wedge z)$. If $L = 1$, then $g = (y \wedge z) \vee (\overline{y} \wedge \overline{z} \wedge s)$ for some literal s. Then $W(g^i) = 5 - 2\alpha$, and it is easy to compute that $\sum_x (W(g^i) - W(g^i_x)) = 18 - 12\alpha > 15 + 5\alpha = (5)(3+\alpha) > W(g^i)(3+\alpha)$ (since $\alpha < 3/17$).

Otherwise, $L \geq 2$, and $W(g^i_{\overline{z}}) = W(g^i_{\overline{y}}) = 1 + (1-\alpha)L$ so $\Delta^i_{\overline{y}} = \Delta^i_{\overline{z}} = 3 - \alpha$. Also, $W(g^i_y) = W(g^i_z) = 1$, so $\Delta^i_y = \Delta^i_z = 3 - \alpha + (1-\alpha)L$. Finally, for every x independent of y and z, $\Delta^i_x \geq (1-\alpha)(L - L((g''_x))) - \alpha$. Combining these, we have: $\sum_x \Delta^i_x \geq 12 - 4\alpha + 2(1-\alpha)L + (1-\alpha)\sum_x (L - L(g''_x)) - 2\alpha L$. Applying Subbotovskaya's bound to g'', this total is at least $12 - 4\alpha + (5 - 7\alpha)L$. Using $\alpha \leq 1/8$, $12 - 4\alpha + (5-7\alpha)L - (3+\alpha)(W(g^i)) \geq 12 - 4\alpha + (5-7\alpha)L - (3+\alpha)L(g^i) = 12 - 4\alpha + (5-7\alpha)L - (3+\alpha)(4+L) = -8\alpha + (2-8\alpha)L \geq L - 1 > 0$, so the total is greater than $(3+\alpha)(W(g^i))$ as desired.

Case 2b g^i is not vulnerable, and it is of the form: $(y \vee z) \wedge g'$, where g' is vulnerable to both $\overline{y}$ and $\overline{z}$.

This case is the dual of Case 2a; the proof is identical.

Case 3: $g^i = maj(y, z, u)$ for some literals y, z, u.

It is easy to calculate that $W(g^i) = 5 - 2\alpha$ and that $W(g^i_x) = 2$ for any x dependent on one of y, z or u. Thus, $\sum_x \Delta^i_x \geq 6(3 - 2\alpha) \geq 5(3+\alpha) \geq (3+\alpha)W(g^i)$, from $\alpha \leq 3/17$.

This ends the proof of proposition 6 and hence of Lemma 2. $\square$

5 The Andreev Bound

Definition 6 *Let the* k-multiplexor *function* $MUX(p_1, ...p_k, f_0, ..f_{2^k-1})$ *be the following boolean function with* $k + 2^k$ *inputs. The output of the function is* f_i*, where* $0 \leq i \leq 2^k - 1$ *is the number whose binary representation is* $p_1, ...p_k$.

Andreev's function *with parameters* k, l *is the boolean function with* $n = lk + 2^k$ *inputs given by:*

$$A(x_{1,1} \ldots x_{1,l} \ldots x_{k,1} \ldots x_{k,l}, f_0, \ldots f_{2^k-1}) =$$

$$MUX(\oplus(x_{1,1}, \ldots x_{1,l}), \ldots, \oplus(x_{k,1}, \ldots, x_{k,l}), f_0, \ldots f_{2^k-1})$$

.

The best bounds are achieved when lk and 2^k are of the same order of magnitude. We thus prove the lower bound for the function A_n defined on $n = kl + 2^k$ variables where $l = 2^k/k$.

Theorem 3 *(Essentially due to [2]) DeMorgan formulas for* A_n *require size* $n^{\gamma+1-o(1)}$.

Proof(of Theorem) In the course of the proof we will assume that γ satisfies that $E[L(f^\epsilon)] \leq O(\epsilon^\gamma L(f))$. The proof for the case where γ does not satisfy this but is only the supremum of numbers that satisfy this follows trivially.

First, let us choose a restriction to the f_i's such that the induced function on the multiplexer has complexity $2^k/k$. I.e. such that a function defined by $g(p_1, \ldots, p_k) = MUX(p_1, \ldots p_k, f_0, \ldots f_{2^k-1})$ has complexity $2^k/k$. Such an assignment exists since there exists a boolean function on k variables with this complexity (see e.g. [3]) and all boolean functions on k variables can be realized by restrictions to the f_i's. Let B be the restriction of A_n to the remaining kl variables. It is clear that $L(A_n) \geq L(B)$, It thus suffices to prove that $L(B) \geq n^{1+\gamma-o(1)}$.

Consider any restriction A^* of B with the property that for each $1 \leq i \leq k$ at least one $x_{i,j}$ remains unfixed. Then $L(A^*) \geq 2^k/k$, since by further restricting the inputs to A^* so that exactly one $x_{i,j}$ remains for each i, and by substituting the complements of variables for variables as neeeded, we obtain the function g.

Consider now a random restriction of the inputs to B leaving only $O(k \log l)$ inputs unfixed, i.e., B^ϵ for $\epsilon = \frac{\log l}{l}$. With at least constnant probability this random restriction has the above property, thus $E[L(B^\epsilon)] \geq \Omega(2^k/k)$. On the other hand, we have $E[L(B^\epsilon)] \leq O(L(B)\epsilon^\gamma)$.

Combining these upper and lower bounds for $E[L(B^\epsilon)]$ yields $L(B) \geq \Omega(\frac{2^k}{k\epsilon^\gamma}) = \Omega(\frac{n^{1+\gamma}}{\log^{2\gamma+1}})$. □

References

[1] M. S. Paterson and U. Zwick. Shrinkage of de Morgan formulae under restriction. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 324–333, 1991.

[2] A.E. Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. In *Moscow Univ. Math. Bull* 42(1)(1987), 63-66.

[3] R. Boppana and M. Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science.*

[4] B.A..Subbotovskaya Realizations of linear functions by formulas using $+, *, -$. In *Soviet Mathematics Doklady* 2(1961), 110-112.

[5] V.M. Khrapchenko. A method of determining lower bounds for the complexity of Π-schemes. In *Math. Notes Acad. Sciences USSR* 10(1971), 474-479.