Known Tight Bounds For The Multiplicative Complexity Of Boolean Functions

René Peralta*
National Institute of Standards and Technology

FewMult Workshop, May 2017

*joint work with Joan Boyar, Meltem Turan, Cagdas Calik

Asymptotic Complexity

- Shannon-Lupanov bound: over the basis $(\vee, \wedge, \neg)$, the circuit complexity of a predicate on n bits is about $\frac{2^n}{n}$ almost everywhere.

Asymptotic Complexity

- Shannon-Lupanov bound: over the basis $(\vee, \wedge, \neg)$, the circuit complexity of a predicate on n bits is about $\frac{2^n}{n}$ almost everywhere.
- Multiplicative complexity: basis is $(\wedge, \neg)$ and we count only the number of $\wedge$ gates.

Asymptotic Complexity

- Shannon-Lupanov bound: over the basis $(\vee, \wedge, \neg)$, the circuit complexity of a predicate on n bits is about $\frac{2^n}{n}$ almost everywhere.
- Multiplicative complexity: basis is $(\wedge, \neg)$ and we count only the number of $\wedge$ gates.
- (BPP,Nechiporuk): almost all Boolean predicates on n bits have multiplicative complexity at least $2^{\frac{n}{2}} - n$.

Asymptotic Complexity

- Shannon-Lupanov bound: over the basis $(\vee, \wedge, \neg)$, the circuit complexity of a predicate on n bits is about $\frac{2^n}{n}$ almost everywhere.
- Multiplicative complexity: basis is $(\wedge, \neg)$ and we count only the number of $\wedge$ gates.
- (BPP,Nechiporuk): almost all Boolean predicates on n bits have multiplicative complexity at least $2^{\frac{n}{2}} - n$.
- (...) : all Boolean predicates on n bits have multiplicative complexity at most $\frac{3}{\sqrt{2}}2^{\frac{n}{2}} - \frac{n}{2}$.

Asymptotic Complexity

- Shannon-Lupanov bound: over the basis $(\vee, \wedge, \neg)$, the circuit complexity of a predicate on n bits is about $\frac{2^n}{n}$ almost everywhere.
- Multiplicative complexity: basis is $(\wedge, \neg)$ and we count only the number of $\wedge$ gates.
- (BPP,Nechiporuk): almost all Boolean predicates on n bits have multiplicative complexity at least $2^{\frac{n}{2}} - n$.
- (...) : all Boolean predicates on n bits have multiplicative complexity at most $\frac{3}{\sqrt{2}}2^{\frac{n}{2}} - \frac{n}{2}$.

(notation): we will use arithmetic modulo 2 instead of $(\wedge, \neg)$.

Concrete, rather than asymptotic complexity

Our interest is in practical applications. Hence we are looking into the *concrete complexity* of circuit optimization problems.

Our interest is in practical applications. Hence we are looking into the *concrete complexity* of circuit optimization problems.

Majority of three

Consider the *threshold function*

$$T_2^3(x_1, x_2, x_3) = 1 \qquad \text{iff at least 2 inputs are 1.}$$

Our interest is in practical applications. Hence we are looking into the *concrete complexity* of circuit optimization problems.

Majority of three

Consider the *threshold function*

$$T_2^3(x_1, x_2, x_3) = 1 \qquad \text{iff at least 2 inputs are 1.}$$

$$T_2^3(x_1, x_2, x_3) \quad = \quad x_1x_2 + x_1x_3 + x_2x_3$$

Our interest is in practical applications. Hence we are looking into the *concrete complexity* of circuit optimization problems.

Majority of three

Consider the *threshold function*

$$T_2^3(x_1, x_2, x_3) = 1 \qquad \text{iff at least 2 inputs are 1.}$$

$$\begin{aligned} T_2^3(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_2x_3 \\ &= x_1(x_2 + x_3) + x_2x_3 \end{aligned}$$

Our interest is in practical applications. Hence we are looking into the *concrete complexity* of circuit optimization problems.

Majority of three

Consider the *threshold function*

$$T_2^3(x_1, x_2, x_3) = 1 \qquad \text{iff at least 2 inputs are 1.}$$

$$\begin{aligned} T_2^3(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_2x_3 \\ &= x_1(x_2 + x_3) + x_2x_3 \\ &= (x_1 + x_2)(x_1 + x_3) + x_1. \end{aligned}$$

Our interest is in practical applications. Hence we are looking into the *concrete complexity* of circuit optimization problems.

Majority of three

Consider the *threshold function*

$$T_2^3(x_1, x_2, x_3) = 1 \qquad \text{iff at least 2 inputs are 1.}$$

$$\begin{aligned} T_2^3(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_2x_3 \\ &= x_1(x_2 + x_3) + x_2x_3 \\ &= (x_1 + x_2)(x_1 + x_3) + x_1. \end{aligned}$$

So $c_\wedge(T_2^3) = 1$.

Quadratic Forms

What about the **quadratic form**

$$f(\vec{x}) = x_1x_2 + x_3x_5 + x_2x_4 + x_1x_3 + x_2x_5 + x_4x_5 + x_1x_5?$$

Quadratic Forms

What about the **quadratic form**

$$f(\vec{x}) = x_1x_2 + x_3x_5 + x_2x_4 + x_1x_3 + x_2x_5 + x_4x_5 + x_1x_5?$$

(Mirwald and Schnorr): The multiplicative complexity of a quadratic form on n-variables is half the rank of an associated $n \times n$ matrix over $GF(2)$.

Quadratic Forms

What about the **quadratic form**

$$f(\vec{x}) = x_1x_2 + x_3x_5 + x_2x_4 + x_1x_3 + x_2x_5 + x_4x_5 + x_1x_5?$$

(Mirwald and Schnorr): The multiplicative complexity of a quadratic form on n-variables is half the rank of an associated $n \times n$ matrix over $GF(2)$.

... at most $\lfloor \frac{n}{2} \rfloor$.

Quadratic Forms

What about the **quadratic form**

$$f(\vec{x}) = x_1x_2 + x_3x_5 + x_2x_4 + x_1x_3 + x_2x_5 + x_4x_5 + x_1x_5?$$

(Mirwald and Schnorr): The multiplicative complexity of a quadratic form on n-variables is half the rank of an associated $n \times n$ matrix over $GF(2)$.

... at most $\lfloor \frac{n}{2} \rfloor$.

This is a constructive result. We can efficiently find a $\wedge-$optimal circuit for any quadratic form.

Threshold Functions

What about T_k^n ?

Threshold Functions

What about T_k^n ?

$$
\begin{aligned}
T_3^5 \quad = \quad & x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + \\
& x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4 + x_2x_3x_5 + \\
& x_2x_4x_5 + x_3x_4x_5 + x_1x_2x_3x_4 + \\
& x_1x_2x_3x_5 + x_1x_2x_4x_5 + x_1x_3x_4x_5 + \\
& x_2x_3x_4x_5
\end{aligned}
$$

Threshold Functions

What about T_k^n ?

$$
\begin{aligned}
T_3^5 \quad = \quad & x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + \\
& x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4 + x_2x_3x_5 + \\
& x_2x_4x_5 + x_3x_4x_5 + x_1x_2x_3x_4 + \\
& x_1x_2x_3x_5 + x_1x_2x_4x_5 + x_1x_3x_4x_5 + \\
& x_2x_3x_4x_5
\end{aligned}
$$

It turns out only 3 multiplications are needed.

Symmetric Functions

- A function is symmetric if it only depends on the Hamming Weight (number of 1s) in the input.

Symmetric Functions

- A function is symmetric if it only depends on the Hamming Weight (number of 1s) in the input.
- (BPP) The multiplicative complexity of any symmetric predicate on n bits is at most
$$n + 3\sqrt{n}.$$

The Elementary Symmetric Functions

Σ_k^n is the predicate on n bits computed by the sum of all terms of degree k.

The Elementary Symmetric Functions

Σ_k^n is the predicate on n bits computed by the sum of all terms of degree k.

$$T_3^5 = \Sigma_3^5 + \Sigma_4^5$$

The Elementary Symmetric Functions

Σ_k^n is the predicate on n bits computed by the sum of all terms of degree k.

$$\begin{aligned}
T_3^5 &= \Sigma_3^5 + \Sigma_4^5 \\
&= x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + \\
&\quad x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4 + x_2x_3x_5 + \\
&\quad x_2x_4x_5 + x_3x_4x_5 + x_1x_2x_3x_4 + \\
&\quad x_1x_2x_3x_5 + x_1x_2x_4x_5 + x_1x_3x_4x_5 + \\
&\quad x_2x_3x_4x_5
\end{aligned}$$

The Elementary Symmetric Functions

Σ_k^n is the predicate on n bits computed by the sum of all terms of degree k.

$$\begin{aligned} T_3^5 &= \Sigma_3^5 + \Sigma_4^5 \\ &= x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + \\ &\quad x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4 + x_2x_3x_5 + \\ &\quad x_2x_4x_5 + x_3x_4x_5 + x_1x_2x_3x_4 + \\ &\quad x_1x_2x_3x_5 + x_1x_2x_4x_5 + x_1x_3x_4x_5 + \\ &\quad x_2x_3x_4x_5 \end{aligned}$$

This is not an accident: *any symmetric function decomposes into a sum of elementary symmetric functions*.

Known Values Of $c_\wedge(\Sigma^n_k)$

Σ^n_i	i						
n	2	3	4	5	6	7	8
3	1	2	–	–	–	–	–
4	2	2	3	–	–	–	–
5	2	3	3	4	–	–	–
6	3	3	4	4	5	–	–
7	3	4	4	5	5	6	–
8	4	4	5-6	5	6	6	7

Known Values Of $c_\wedge(\Sigma_k^n)$

Σ_i^n	i						
n	2	3	4	5	6	7	8
3	1	2	–	–	–	–	–
4	2	2	3	–	–	–	–
5	2	3	3	4	–	–	–
6	3	3	4	4	5	–	–
7	3	4	4	5	5	6	–
8	4	4	5-6	5	6	6	7

The complexity of Σ_4^8 is an open problem.

Known Values Of $c_\wedge(\Sigma_k^n)$

Σ_i^n	i						
n	2	3	4	5	6	7	8
3	1	2	–	–	–	–	–
4	2	2	3	–	–	–	–
5	2	3	3	4	–	–	–
6	3	3	4	4	5	–	–
7	3	4	4	5	5	6	–
8	4	4	5-6	5	6	6	7

The complexity of Σ_4^8 is an open problem.

There is a monotonicity conjecture $c_\wedge(\Sigma_k^n) \leq c_\wedge(\Sigma_{k+1}^n)$.

Known Values Of $c_\wedge(\Sigma_k^n)$

Σ_i^n	i						
n	2	3	4	5	6	7	8
3	1	2	–	–	–	–	–
4	2	2	3	–	–	–	–
5	2	3	3	4	–	–	–
6	3	3	4	4	5	–	–
7	3	4	4	5	5	6	–
8	4	4	5-6	5	6	6	7

The complexity of Σ_4^8 is an open problem.

There is a monotonicity conjecture $c_\wedge(\Sigma_k^n) \leq c_\wedge(\Sigma_{k+1}^n)$.

In fact, all known values of $c_\wedge(\Sigma_m^n)$ satisfy $\left\lfloor \frac{n+m}{2} \right\rfloor - 1$.

Other known values

- $c_\wedge(\Sigma_2^n) = \lfloor \frac{n}{2} \rfloor$
- $c_\wedge(\Sigma_3^n) = \lceil \frac{n}{2} \rceil$
- $c_\wedge(\Sigma_{n-1}^n) = n - 2$
- $c_\wedge(\Sigma_{n-2}^n) = n - 2$
- $c_\wedge(\Sigma_{n-3}^n) = n - 3$

Other known values

- $c_\wedge(\Sigma_2^n) = \lfloor \frac{n}{2} \rfloor$
- $c_\wedge(\Sigma_3^n) = \lceil \frac{n}{2} \rceil$
- $c_\wedge(\Sigma_{n-1}^n) = n - 2$
- $c_\wedge(\Sigma_{n-2}^n) = n - 2$
- $c_\wedge(\Sigma_{n-3}^n) = n - 3$

More formulas, and useful identities in (BP 1998, TCS 396 pp. 223 – 246)

Known Values Of $c_\wedge(T_k^n)$

$c_\wedge(T_i^n)$	i							
n	1	2	3	4	5	6	7	8
3	2	1	2	–	–	–	–	–
4	3	3	3	3	–	–	–	–
5	4	3	3	3	4	–	–	–
6	5	5	4	4	5	5	–	–
7	6	5	6	4	6	5	6	–
8	7	7	7	7	7	7	7	7

Known Values Of $c_\wedge(E_k^n)$

$c_\wedge(E_i^n)$	i								
n	0	1	2	3	4	5	6	7	8
3	2	2	2	2	–	–	–	–	–
4	3	2	2	2	3	–	–	–	–
5	4	4	3	3	4	4	–	–	–
6	5	4	5	3	5	4	5	–	–
7	6	6	6	6	6	6	6	6	–
8	7	6	6	6	**6**	6	6	6	7

(Cagdas and Turan): $c_\wedge(E_4^8) = 6$.

Hamming Weight

The *Hamming Weight* $H(x_1, \ldots, x_n)$ is the number of 1s among the x_is.

Hamming Weight

The *Hamming Weight* $H(x_1, \ldots, x_n)$ is the number of 1s among the x_is.

Computing the binary representation of $H()$, such as

$$H(1,0,1,0,1,1,0,1) = 101_2,$$

is a basic operation for integer arithmetic.

Hamming Weight

Denote by H^n the Hamming Weight function on n bits.

Hamming Weight

Denote by H^n the Hamming Weight function on n bits.

It turns out $c_\wedge(H^n) = n - h(n)$, where $h(n)$ is the Hamming Weight of n.

Hamming Weight

Denote by H^n the Hamming Weight function on n bits.

It turns out $c_\wedge(H^n) = n - h(n)$, where $h(n)$ is the Hamming Weight of n.
e.g. $c_\wedge(H^7) = 7 - 3 = 4$ since $7 = 111_2$.

Hamming Weight

It turns out the k^{th} least significant bit of H^n is $\Sigma^n_{2^k}$.

Hamming Weight

It turns out the k^{th} least significant bit of H^n is $\Sigma^n_{2^k}$.

For example

$$H^{13} = \Sigma^{13}_8 \;\; \Sigma^{13}_4 \;\; \Sigma^{13}_2 \;\; \Sigma^{13}_1$$

Hamming Weight

It turns out the k^{th} least significant bit of H^n is $\Sigma^n_{2^k}$.

For example

$$H^{13} = \Sigma^{13}_8 \;\; \Sigma^{13}_4 \;\; \Sigma^{13}_2 \;\; \Sigma^{13}_1$$

... more identities like this.

Hamming Weight

It turns out the k^{th} least significant bit of H^n is $\Sigma^n_{2^k}$.

For example

$$H^{13} = \Sigma^{13}_8 \;\; \Sigma^{13}_4 \;\; \Sigma^{13}_2 \;\; \Sigma^{13}_1$$

... more identities like this.

More generally: the multiplicative complexity of the Hamming Weight implies bounds on the complexity of integer sum, integer multiplication, binary polynomial multiplication, finite field arithmetic, ...

Bound on all functions on n inputs

Denote by f_n a function on n inputs. Note that the function $f = x_1 \cdot x_2 \cdots x_n$ has multiplicative complexity $n-1$.

- $\forall f_4 \quad : \quad c_\wedge(f_4) \leq 3$

Bound on all functions on n inputs

Denote by f_n a function on n inputs. Note that the function $f = x_1 \cdot x_2 \cdots x_n$ has multiplicative complexity $n-1$.

- $\forall f_4 \quad : \quad c_\wedge(f_4) \leq 3$

  In fact, all 4-bit bijections have multiplicative complexity at most 5 (Zajac et. al.).

Bound on all functions on n inputs

Denote by f_n a function on n inputs. Note that the function $f = x_1 \cdot x_2 \cdots x_n$ has multiplicative complexity $n-1$.

- $\forall f_4 \quad : \quad c_\wedge(f_4) \leq 3$

  In fact, all 4-bit bijections have multiplicative complexity at most 5 (Zajac et. al.).

- $\forall f_5 \quad : \quad c_\wedge(f_5) \leq 4$.

Bound on all functions on n inputs

Denote by f_n a function on n inputs. Note that the function $f = x_1 \cdot x_2 \cdots x_n$ has multiplicative complexity $n-1$.

- $\forall f_4 \quad : \quad c_\wedge(f_4) \leq 3$

  In fact, all 4-bit bijections have multiplicative complexity at most 5 (Zajac et. al.).

- $\forall f_5 \quad : \quad c_\wedge(f_5) \leq 4$.

- $\exists f_7 \quad : \quad c_\wedge(f_7) \geq 7$ (a simple counting argument).

Bound on all functions on n inputs

Denote by f_n a function on n inputs. Note that the function $f = x_1 \cdot x_2 \cdots x_n$ has multiplicative complexity $n-1$.

- $\forall f_4 \quad : \quad c_\wedge(f_4) \leq 3$

  In fact, all 4-bit bijections have multiplicative complexity at most 5 (Zajac et. al.).

- $\forall f_5 \quad : \quad c_\wedge(f_5) \leq 4$.

- $\exists f_7 \quad : \quad c_\wedge(f_7) \geq 7$ (a simple counting argument).

More good stuff using SAT solvers by Courtois, Zajac and others.

What about functions on six inputs?

What about functions on six inputs?

There are 150,357 affine equivalence classes.

What about functions on six inputs?

There are 150,357 affine equivalence classes.

- Codish et al conjecture $\forall f_5 \quad : \quad c_\wedge(f_5) \leq 5$.

What about functions on six inputs?

There are 150,357 affine equivalence classes.

- Codish et al conjecture $\forall f_5 \quad : \quad c_\wedge(f_5) \leq 5$.
- At NIST we ran a large computation that enumerates circuits by the location of the AND gates. If our code is correct, there are 931 affine equivalence classes with multiplicative complexity 6.

What about functions on six inputs?

There are 150,357 affine equivalence classes.

- Codish et al conjecture $\forall f_5 \quad : \quad c_\wedge(f_5) \leq 5$.
- At NIST we ran a large computation that enumerates circuits by the location of the AND gates. If our code is correct, there are 931 affine equivalence classes with multiplicative complexity 6.

checking, checking , $\ldots$.

Multiplicative complexity is not hopeless

- In the last few years we have developed a number of tools to bound multiplicative complexity.
- these results are constructive, so we can build circuits.
- when we build a circuit with “few” multiplications, it often has large linear components.

A new logic synthesis method

To build a circuit for a given function we can try the following

1. construct a circuit with few multiplications;
2. optimize the linear part.

Some new results

- A circuit for the S-box of AES with depth 16 and 125 gates.
- A circuit for multiplication in $GF(2^{16})$ with depth 6 and size 106.
- Built a circuit for 16-bit arithmetic which reduced by 2/3 the size of circuit in MILCOM 2015.