Linear Bounds on Circuit Complexity and Feebly One-way Permutations

Grigory Yaroslavtsev

Academic University, Saint-Petersburg, Russia
http://logic.pdmi.ras.ru/~grigory

April 5, 2010

Plan

1. Introduction
2. Upper bounds on circuit complexity
3. Lower bounds on circuit complexity
4. Feebly one-way families of permutations

Motivation

Practical:

- Logical design synthesis: smaller circuits — better designs.

Motivation

Practical:

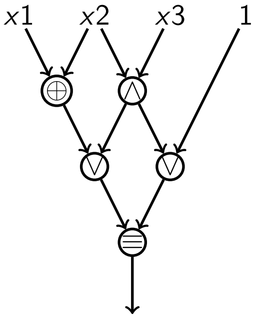
- Logical design synthesis: smaller circuits — better designs.

Theoretical:

- Circuits — very simple and natural model of computation. Many efforts spent — not too much known.

Boolean Circuits

- inputs: propositional variables $x_1, x_2, \ldots, x_n$ and constants 0, 1
- gates: binary functions
- fan-out of a gate is unbounded



Symmetric functions

Definition

A boolean function is symmetric if its value depends on the sum of the input values only.

Example: $MAJ(x_1, \ldots, x_n) = 1 \iff x_1 + \ldots + x_n \geq n/2$

Symmetric functions

Definition

A boolean function is symmetric if its value depends on the sum of the input values only.

Example: $MAJ(x_1, \ldots, x_n) = 1 \iff x_1 + \ldots + x_n \geq n/2$

Modular functions

Let $\mathrm{MOD}^n_{m,r}(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^n x_i \equiv r \pmod{m}$.

Example: $\mathrm{MOD}^n_{4,0}(x_1, \ldots, x_n = 1) \iff \sum_{i=1}^n x_i \equiv \{0, 4, 8, \ldots\}$

Stockmeyer's bounds for $\mathrm{MOD}_{4,0}^n$

- Stockmeyer constructed a circuit for $\mathrm{MOD}_{4,0}^n$ of size $2.5n + c$, using blocks with 6 inputs and 10 gates to add 4 new values to the remainder encoded by 2 bits and transfer the remainder encoded in 2 bits to the next block.

Stockmeyer's bounds for $\mathrm{MOD}_{4,0}^n$

- Stockmeyer constructed a circuit for $\mathrm{MOD}_{4,0}^n$ of size $2.5n + c$, using blocks with 6 inputs and 10 gates to add 4 new values to the remainder encoded by 2 bits and transfer the remainder encoded in 2 bits to the next block.
- This matches the corresponding lower bound $2.5n + c$ proved by him.

Applying practice to theory

"For many important functions there is a large gap between known lower and upper bounds. It might be helpful to know optimal circuits for such functions at least for small values of input size. Knowing this could help us to understand the structure of optimal circuits for general functions."

Williams, R. (2008)

Applying practice to theory

> *"For many important functions there is a large gap between known lower and upper bounds. It might be helpful to know optimal circuits for such functions at least for small values of input size. Knowing this could help us to understand the structure of optimal circuits for general functions."*
>
> *Williams, R. (2008)*

By finding efficient small circuits we can obtain upper bounds on circiut complexity.

Main idea

Bruteforce search

- The number $F(n, t)$ of circuits of size $\leq t$ with n input variables does not exceed
$$\left(16(t+n+2)^2\right)^t .$$
Each of t gates is assigned one of 16 possible binary Boolean functions that acts on two previous nodes, and each previous node can be either a previous gate ($\leq t$ choices) or a variable or a constant ($\leq n+2$ choices).

Main idea

Bruteforce search

- The number $F(n, t)$ of circuits of size $\leq t$ with n input variables does not exceed
$$\left(16(t+n+2)^2\right)^t .$$
Each of t gates is assigned one of 16 possible binary Boolean functions that acts on two previous nodes, and each previous node can be either a previous gate ($\leq t$ choices) or a variable or a constant ($\leq n+2$ choices).
- To find Stockmeyer's block (6 inputs, 10 gates) a naive bruteforce over $\sim 1.4 * 10^{37}$ circuits will be needed.

Main idea

Given a function $f : \{0,1\}^n \to \{0,1\}^m$ (n, m are constants) we transform the fact "there exists a circuit of size m computing function f" into a CNF formula and use SAT-solvers to check its satisfiability.

Main idea

Given a function $f : \{0,1\}^n \to \{0,1\}^m$ (n, m are constants) we transform the fact "there exists a circuit of size m computing function f" into a CNF formula and use SAT-solvers to check its satisfiability.

Enconding

- All possible underlying graphs of circuit
- All possibilities for functions computed by gates
- Which gates are outputs
- The particular function computed by circuit

Results

Results

- New upper bound for $\mathrm{MOD}_{3,*}^{n}$: $3n + c$ in full binary basis B_2 (previous $5n + c$), using a block with 5 inputs and 9 gates.

Results

Results

- New upper bound for $\mathrm{MOD}_{3,*}^{n}$: $3n + c$ in full binary basis B_2 (previous $5n + c$), using a block with 5 inputs and 9 gates.
- New upper bound for $\mathrm{MOD}_{3,*}^{n}$: $5.5n + c$ in basis $U_2 = B_2 \setminus \{\oplus, \equiv\}$ (previous $7n + c$), using a block with 4 inputs and 11 gates.

Results

Results

- New upper bound for $\mathrm{MOD}^n_{3,*}$: $3n + c$ in full binary basis B_2 (previous $5n + c$), using a block with 5 inputs and 9 gates.
- New upper bound for $\mathrm{MOD}^n_{3,*}$: $5.5n + c$ in basis $U_2 = B_2 \setminus \{\oplus, \equiv\}$ (previous $7n + c$), using a block with 4 inputs and 11 gates.
- It is possible to prove exact bounds for circuits with ≤ 8 gates.

Random Functions are Complex

Random Functions are Complex

- Shannon counting argument: count how many different Boolean functions in n variables can be computed by circuits with t gates and compare this number with the total number 2^{2^n} of all Boolean functions.

Random Functions are Complex

- Shannon counting argument: count how many different Boolean functions in n variables can be computed by circuits with t gates and compare this number with the total number 2^{2^n} of all Boolean functions.
- The number $F(n, t)$ of circuits of size $\leq t$ with n input variables does not exceed
$$\left(16(t + n + 2)^2\right)^t .$$
Each of t gates is assigned one of 16 possible binary Boolean functions that acts on two previous nodes, and each previous node can be either a previous gate ($\leq t$ choices) or a variables or a constant ($\leq n + 2$ choices).

Random Functions are Complex

- Shannon counting argument: count how many different Boolean functions in n variables can be computed by circuits with t gates and compare this number with the total number 2^{2^n} of all Boolean functions.
- The number $F(n,t)$ of circuits of size $\leq t$ with n input variables does not exceed
$$\left(16(t+n+2)^2\right)^t .$$
Each of t gates is assigned one of 16 possible binary Boolean functions that acts on two previous nodes, and each previous node can be either a previous gate ($\leq t$ choices) or a variables or a constant ($\leq n+2$ choices).
- For $t = 2^n/(10n)$, $F(n,t)$ is approximately $2^{2^n/5}$, which is $\ll 2^{2^n}$.

Random Functions are Complex

- Shannon counting argument: count how many different Boolean functions in n variables can be computed by circuits with t gates and compare this number with the total number 2^{2^n} of all Boolean functions.
- The number $F(n,t)$ of circuits of size $\leq t$ with n input variables does not exceed
$$\left(16(t+n+2)^2\right)^t .$$
Each of t gates is assigned one of 16 possible binary Boolean functions that acts on two previous nodes, and each previous node can be either a previous gate ($\leq t$ choices) or a variables or a constant ($\leq n+2$ choices).
- For $t = 2^n/(10n)$, $F(n,t)$ is approximately $2^{2^n/5}$, which is $\ll 2^{2^n}$.
- Thus, the circuit complexity of almost all Boolean functions on n variables is exponential in n. Still, we do not know any explicit function with super-linear circuit complexity.

Known Lower Bounds

	circuit size	formula size
full binary basis B_2	$3n - o(n)$ [Blum]	$n^{2-o(1)}$ [Nechiporuk]
basis $U_2 = B_2 \setminus \{\oplus, \equiv\}$	$5n - o(n)$ [Iwama et al.]	$n^{3-o(1)}$ [Hastad]
monotone basis $M_2 = \{\vee, \wedge\}$	exponential [Razborov; Alon, Boppana; Andreev; Karchmer, Wigderson]	

Explicit Functions

Explicit Functions

- We are interested in explicitly defined Boolean functions of high circuit complexity.

Explicit Functions

- We are interested in explicitly defined Boolean functions of high circuit complexity.
- Not explicitly defined function of high circuit complexity: enumerate all Boolean functions on n variables and take the first with circuit complexity at least $2^n/(10n)$.

Explicit Functions

- We are interested in explicitly defined Boolean functions of high circuit complexity.
- Not explicitly defined function of high circuit complexity: enumerate all Boolean functions on n variables and take the first with circuit complexity at least $2^n/(10n)$.
- To avoid tricks like this one, we say that a function f is explicitly defined if $f^{-1}(1)$ is in NP.

Explicit Functions

- We are interested in explicitly defined Boolean functions of high circuit complexity.
- Not explicitly defined function of high circuit complexity: enumerate all Boolean functions on n variables and take the first with circuit complexity at least $2^n/(10n)$.
- To avoid tricks like this one, we say that a function f is explicitly defined if $f^{-1}(1)$ is in NP.
- Usually, under a Boolean function f we actually understand an infinite sequence $\{f_n \mid n = 1, 2, \ldots\}$.

Known Lower Bounds for Circuits over B_2

Known Lower Bounds

$2n - c$	[Schnorr, 74]
$2.5n - o(n)$	[Paul, 77]
$2.5n - c$	[Stockmeyer, 77]
$3n - o(n)$	[Blum, 84]

Known Lower Bounds for Circuits over B_2

Known Lower Bounds

$2n - c$	[Schnorr, 74]
$2.5n - o(n)$	[Paul, 77]
$2.5n - c$	[Stockmeyer, 77]
$3n - o(n)$	[Blum, 84]

This Talk

In this talk, we will present a proof of a $7n/3 - c$ lower bound which is as simple as Schnorr's proof of $2n - c$ lower bound.

Known Lower Bounds for Circuits over B_2

Known Lower Bounds

$2n - c$	[Schnorr, 74]
$2.5n - o(n)$	[Paul, 77]
$2.5n - c$	[Stockmeyer, 77]
$3n - o(n)$	[Blum, 84]

This Talk

In this talk, we will present a proof of a $7n/3 - c$ lower bound which is as simple as Schnorr's proof of $2n - c$ lower bound.

Gate Elimination

All the proofs are based on the so-called gate elimination method. This is essentially the only known method for proving lower bounds on circuit complexity.

Gate Elimination Method

The main idea

Gate Elimination Method

The main idea

- Take an optimal circuit for the function in question.

Gate Elimination Method

The main idea

- Take an optimal circuit for the function in question.
- Setting some variables to constants obtain a subfunction of the same type (in order to proceed by induction) and eliminate several gates.

Gate Elimination Method

The main idea

- Take an optimal circuit for the function in question.
- Setting some variables to constants obtain a subfunction of the same type (in order to proceed by induction) and eliminate several gates.
- A gate is eliminated if it computes a constant or a variable.

Gate Elimination Method

The main idea

- Take an optimal circuit for the function in question.
- Setting some variables to constants obtain a subfunction of the same type (in order to proceed by induction) and eliminate several gates.
- A gate is eliminated if it computes a constant or a variable.
- By repeatedly applying this process, conclude that the original circuit must have had many gates.

Gate Elimination Method

The main idea

- Take an optimal circuit for the function in question.
- Setting some variables to constants obtain a subfunction of the same type (in order to proceed by induction) and eliminate several gates.
- A gate is eliminated if it computes a constant or a variable.
- By repeatedly applying this process, conclude that the original circuit must have had many gates.

Remark

This method is very unlikely to produce nonlinear lower bounds.

The Class $Q^n_{2,3}$

Definition

A function $f\colon \{0,1\}^n \to \{0,1\}$ belongs to the class $Q^n_{2,3}$ if

The Class $Q_{2,3}^n$

Definition

A function $f\colon \{0,1\}^n \to \{0,1\}$ belongs to the class $Q_{2,3}^n$ if

1. for all different $i, j \in \{1, \ldots, n\}$, one obtains at least three different subfunctions by replacing x_i and x_j by constants;

The Class $Q_{2,3}^n$

Definition

A function $f\colon \{0,1\}^n \to \{0,1\}$ belongs to the class $Q_{2,3}^n$ if

1. for all different $i, j \in \{1, \ldots, n\}$, one obtains at least three different subfunctions by replacing x_i and x_j by constants;
2. for all $i \in \{1, \ldots, n\}$, one obtains a subfunction in $Q_{2,3}^{n-1}$ (if $n \geq 4$) by replacing x_i by any constant.

The Class $Q_{2,3}^n$

Definition

A function $f\colon \{0,1\}^n \to \{0,1\}$ belongs to the class $Q_{2,3}^n$ if

1. for all different $i, j \in \{1, \ldots, n\}$, one obtains at least three different subfunctions by replacing x_i and x_j by constants;
2. for all $i \in \{1, \ldots, n\}$, one obtains a subfunction in $Q_{2,3}^{n-1}$ (if $n \geq 4$) by replacing x_i by any constant.

Modular functions

The Class $Q^n_{2,3}$

Definition

A function $f\colon \{0,1\}^n \to \{0,1\}$ belongs to the class $Q^n_{2,3}$ if

1. for all different $i, j \in \{1, \dots, n\}$, one obtains at least three different subfunctions by replacing x_i and x_j by constants;
2. for all $i \in \{1, \dots, n\}$, one obtains a subfunction in $Q^{n-1}_{2,3}$ (if $n \geq 4$) by replacing x_i by any constant.

Modular functions

- Let $\mathrm{MOD}^n_{m,r}(x_1, \dots, x_n) = 1 \iff \sum_{i=1}^n x_i \equiv r \pmod{m}$.

The Class $Q_{2,3}^n$

Definition

A function $f\colon \{0,1\}^n \to \{0,1\}$ belongs to the class $Q_{2,3}^n$ if

1. for all different $i, j \in \{1, \dots, n\}$, one obtains at least three different subfunctions by replacing x_i and x_j by constants;
2. for all $i \in \{1, \dots, n\}$, one obtains a subfunction in $Q_{2,3}^{n-1}$ (if $n \geq 4$) by replacing x_i by any constant.

Modular functions

- Let $\mathrm{MOD}_{m,r}^n(x_1, \dots, x_n) = 1 \iff \sum_{i=1}^n x_i \equiv r \pmod{m}$.
- Then $\mathrm{MOD}_{3,r}^n, \mathrm{MOD}_{4,r}^n \in Q_{2,3}^n$, but $\mathrm{MOD}_{2,r}^n \notin Q_{2,3}^n$.

Schnorr's $2n$ Lower Bound

Theorem

If $f \in Q^n_{2,3}$, then $C(f) \geq 2n - 8$.

Schnorr's $2n$ Lower Bound

Theorem

If $f \in Q^n_{2,3}$, then $C(f) \geq 2n - 8$.

Proof

Schnorr's $2n$ Lower Bound

Theorem

If $f \in Q^n_{2,3}$, then $C(f) \geq 2n - 8$.

Proof

- Induction on n. If $n \leq 4$, then the statement is trivial.

Schnorr's $2n$ Lower Bound

Theorem

If $f \in Q^n_{2,3}$, then $C(f) \geq 2n - 8$.

Proof

- Induction on n. If $n \leq 4$, then the statement is trivial.
- Consider an optimal circuit and its top gate Q which is fed by different variables x_i and x_j (they are different, since the circuit is optimal).

Schnorr's $2n$ Lower Bound

Theorem

If $f \in Q^n_{2,3}$, then $C(f) \geq 2n - 8$.

Proof

- Induction on n. If $n \leq 4$, then the statement is trivial.
- Consider an optimal circuit and its top gate Q which is fed by different variables x_i and x_j (they are different, since the circuit is optimal).
- Note that $Q = Q(x_i, x_j)$ can only take two values, 0 and 1, when x_i and x_j are fixed.

Schnorr's $2n$ Lower Bound

Theorem

If $f \in Q^n_{2,3}$, then $C(f) \geq 2n - 8$.

Proof

- Induction on n. If $n \leq 4$, then the statement is trivial.
- Consider an optimal circuit and its top gate Q which is fed by different variables x_i and x_j (they are different, since the circuit is optimal).
- Note that $Q = Q(x_i, x_j)$ can only take two values, 0 and 1, when x_i and x_j are fixed.
- Thus, either x_i or x_j fans out to another gate P.

Schnorr's $2n$ Lower Bound

Theorem

If $f \in Q^n_{2,3}$, then $C(f) \geq 2n - 8$.

Proof

- Induction on n. If $n \leq 4$, then the statement is trivial.
- Consider an optimal circuit and its top gate Q which is fed by different variables x_i and x_j (they are different, since the circuit is optimal).
- Note that $Q = Q(x_i, x_j)$ can only take two values, 0 and 1, when x_i and x_j are fixed.
- Thus, either x_i or x_j fans out to another gate P.
- By assigning this variable, we eliminate at least two gates and get a subfunction from $Q^{n-1}_{2,3}$. □

AND-type Gates vs XOR-type Gates

Binary functions

The set B_2 of all binary functions contains 16 functions $f(x, y)$:

AND-type Gates vs XOR-type Gates

Binary functions

The set B_2 of all binary functions contains 16 functions $f(x, y)$:

1. 2 constants: 0, 1

AND-type Gates vs XOR-type Gates

Binary functions

The set B_2 of all binary functions contains 16 functions $f(x, y)$:

1. 2 constants: 0, 1
2. 4 degenerate functions: x, $\bar{x}$, y, $\bar{y}$.

AND-type Gates vs XOR-type Gates

Binary functions

The set B_2 of all binary functions contains 16 functions $f(x, y)$:

1. 2 constants: 0, 1
2. 4 degenerate functions: x, $\bar{x}$, y, $\bar{y}$.
3. 2 XOR-type functions: $x \oplus y \oplus a$, where $a \in \{0, 1\}$.

AND-type Gates vs XOR-type Gates

Binary functions

The set B_2 of all binary functions contains 16 functions $f(x, y)$:

1. 2 constants: 0, 1
2. 4 degenerate functions: x, $\bar{x}$, y, $\bar{y}$.
3. 2 XOR-type functions: $x \oplus y \oplus a$, where $a \in \{0, 1\}$.
4. 8 AND-type functions: $(x \oplus a)(y \oplus b) \oplus c$, where $a, b, c \in \{0, 1\}$.

AND-type Gates vs XOR-type Gates

Binary functions

The set B_2 of all binary functions contains 16 functions $f(x, y)$:

1. 2 constants: 0, 1
2. 4 degenerate functions: x, $\bar{x}$, y, $\bar{y}$.
3. 2 XOR-type functions: $x \oplus y \oplus a$, where $a \in \{0, 1\}$.
4. 8 AND-type functions: $(x \oplus a)(y \oplus b) \oplus c$, where $a, b, c \in \{0, 1\}$.

Remark

Optimal circuits contain AND- and XOR-type gates only, as constant and degenerate gates can be easily eliminated.

AND-type Gates vs XOR-type Gates

AND-type Gates vs XOR-type Gates

AND-type Gates vs XOR-type Gates

AND-type Gates vs XOR-type Gates

- AND-type gates are easier to handle than XOR-type gates.

AND-type Gates vs XOR-type Gates

AND-type Gates vs XOR-type Gates

- AND-type gates are easier to handle than XOR-type gates.
- Let $Q(x_i, x_j) = (x_i \oplus a)(x_j \oplus b) \oplus c$ be an AND-type gate. Then by assigning $x_i = a$ or $x_j = b$ we make this gate constant. That is, we eliminate not only this gate, but also all its direct successors!

AND-type Gates vs XOR-type Gates

AND-type Gates vs XOR-type Gates

- AND-type gates are easier to handle than XOR-type gates.
- Let $Q(x_i, x_j) = (x_i \oplus a)(x_j \oplus b) \oplus c$ be an AND-type gate. Then by assigning $x_i = a$ or $x_j = b$ we make this gate constant. That is, we eliminate not only this gate, but also all its direct successors!
- While by assigning any constant to x_i, we obtain from $Q(x_i, x_j) = x_i \oplus x_j \oplus c$ either x_j or $\bar{x}_j$.

AND-type Gates vs XOR-type Gates

AND-type Gates vs XOR-type Gates

- AND-type gates are easier to handle than XOR-type gates.
- Let $Q(x_i, x_j) = (x_i \oplus a)(x_j \oplus b) \oplus c$ be an AND-type gate. Then by assigning $x_i = a$ or $x_j = b$ we make this gate constant. That is, we eliminate not only this gate, but also all its direct successors!
- While by assigning any constant to x_i, we obtain from $Q(x_i, x_j) = x_i \oplus x_j \oplus c$ either x_j or $\bar{x}_j$.
- That is why, in particular, the current record bounds for circuits over $U_2 = B_2 \setminus \{\oplus, \equiv\}$ are stronger than the bounds over B_2.

AND-type Gates vs XOR-type Gates

AND-type Gates vs XOR-type Gates

- AND-type gates are easier to handle than XOR-type gates.
- Let $Q(x_i, x_j) = (x_i \oplus a)(x_j \oplus b) \oplus c$ be an AND-type gate. Then by assigning $x_i = a$ or $x_j = b$ we make this gate constant. That is, we eliminate not only this gate, but also all its direct successors!
- While by assigning any constant to x_i, we obtain from $Q(x_i, x_j) = x_i \oplus x_j \oplus c$ either x_j or $\bar{x}_j$.
- That is why, in particular, the current record bounds for circuits over $U_2 = B_2 \setminus \{\oplus, \equiv\}$ are stronger than the bounds over B_2.
- Usually, the main bottleneck of a proof based on gate elimination is a circuit whose top contains many XOR-type gates.

Polynomials over $\mathrm{GF}(2)$

Polynomials over $\mathrm{GF}(2)$

Polynomials over $\mathrm{GF}(2)$

Polynomials over $\mathrm{GF}(2)$

- Let $\tau(f)$ denote the unique polynomial over $\mathrm{GF}(2)$ representing f.

Polynomials over $\mathrm{GF}(2)$

Polynomials over $\mathrm{GF}(2)$

- Let $\tau(f)$ denote the unique polynomial over $\mathrm{GF}(2)$ representing f.
- E.g., $\tau(\mathrm{MOD}^3_{3,0}) = x_1x_2x_3 + (1 - x_1)(1 - x_2)(1 - x_3)$.

Polynomials over $\mathrm{GF}(2)$

Polynomials over $\mathrm{GF}(2)$

- Let $\tau(f)$ denote the unique polynomial over $\mathrm{GF}(2)$ representing f.
- E.g., $\tau(\mathrm{MOD}^3_{3,0}) = x_1x_2x_3 + (1-x_1)(1-x_2)(1-x_3)$.
- Note that $\tau(f)$ is multi-linear.

Polynomials over $\mathrm{GF}(2)$

Polynomials over $\mathrm{GF}(2)$

- Let $\tau(f)$ denote the unique polynomial over $\mathrm{GF}(2)$ representing f.
- E.g., $\tau(\mathrm{MOD}^3_{3,0}) = x_1x_2x_3 + (1-x_1)(1-x_2)(1-x_3)$.
- Note that $\tau(f)$ is multi-linear.
- It can be easily shown that, for any r, $\deg(\tau(\mathrm{MOD}^n_{4,r})) \leq 3$, while $\deg(\tau(\mathrm{MOD}^n_{3,r})) \geq n-1$.

Polynomials over $\mathrm{GF}(2)$

Polynomials over $\mathrm{GF}(2)$

- Let $\tau(f)$ denote the unique polynomial over $\mathrm{GF}(2)$ representing f.
- E.g., $\tau(\mathrm{MOD}^3_{3,0}) = x_1x_2x_3 + (1 - x_1)(1 - x_2)(1 - x_3)$.
- Note that $\tau(f)$ is multi-linear.
- It can be easily shown that, for any r, $\deg(\tau(\mathrm{MOD}^n_{4,r})) \leq 3$, while $\deg(\tau(\mathrm{MOD}^n_{3,r})) \geq n - 1$.

Lemma (Degree lower bound)

Any circuit computing f contains at least $\deg(\tau(f)) - 1$ AND-type gates.

Combined Complexity Measure

Idea

Thus, in a bottleneck case we see only XOR-type gates, however we are given several AND-type gates in advance.

Combined Complexity Measure

Idea

Thus, in a bottleneck case we see only XOR-type gates, however we are given several AND-type gates in advance. Let us increase the weight of a XOR-type gate.

Combined Complexity Measure

Idea

Thus, in a bottleneck case we see only XOR-type gates, however we are given several AND-type gates in advance. Let us increase the weight of a XOR-type gate.

Definition

For a circuit C, let $A(C)$ and $X(C)$ denote the number of AND- and XOR-type gates in C, respectively. Let also $\mu(C) = 3X(C) + 2A(C)$.

An Improved Lower Bound

Lemma

For any circuit C computing $f \in Q^n_{2,3}$, $\mu(C) = 3X(C) + 2A(C) \geq 6n - 24$.

An Improved Lower Bound

Lemma

For any circuit C computing $f \in Q^n_{2,3}$, $\mu(C) = 3X(C) + 2A(C) \geq 6n - 24$.

Proof

An Improved Lower Bound

Lemma

For any circuit C computing $f \in Q^n_{2,3}$, $\mu(C) = 3X(C) + 2A(C) \geq 6n - 24$.

Proof

- As in the previous proof, we consider a top gate $Q(x_i, x_j)$ and assume wlog that x_i feeds also another gate P.

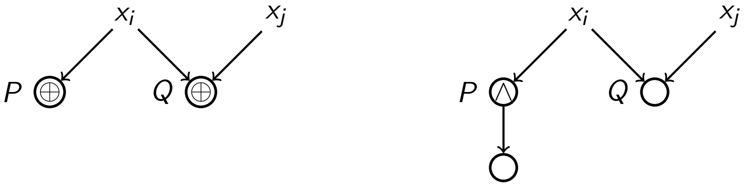
An Improved Lower Bound

Lemma

For any circuit C computing $f \in Q_{2,3}^n$, $\mu(C) = 3X(C) + 2A(C) \geq 6n - 24$.

Proof

- As in the previous proof, we consider a top gate $Q(x_i, x_j)$ and assume wlog that x_i feeds also another gate P.
- There are two cases:



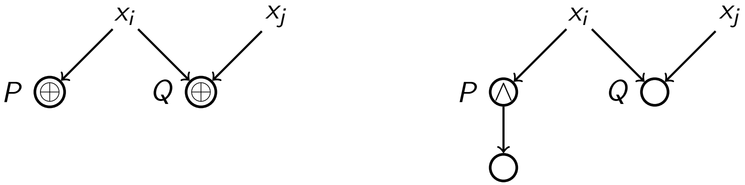
An Improved Lower Bound

Lemma

For any circuit C computing $f \in Q^n_{2,3}$, $\mu(C) = 3X(C) + 2A(C) \geq 6n - 24$.

Proof

- As in the previous proof, we consider a top gate $Q(x_i, x_j)$ and assume wlog that x_i feeds also another gate P.
- There are two cases:



- In both cases, we can assign x_i a constant such that μ is reduced at least by 6. □

$7n/3$ Lower Bound

Lemma

Let $f \in Q^n_{2,3}$ *and* $\deg(\tau(f)) \geq n - c$*, then* $C(f) \geq 7n/3 - c'$*.*

$7n/3$ Lower Bound

Lemma

Let $f \in Q^n_{2,3}$ *and* $\deg(\tau(f)) \geq n - c$, *then* $C(f) \geq 7n/3 - c'$.

Proof

Let C be an optimal circuit computing f.

$7n/3$ Lower Bound

Lemma

Let $f \in Q^n_{2,3}$ and $\deg(\tau(f)) \geq n - c$, then $C(f) \geq 7n/3 - c'$.

Proof

Let C be an optimal circuit computing f.

$$\frac{\begin{aligned} 3X(C) + 2A(C) &\geq 6n - 24 \\ A(C) &\geq n - c - 1 \end{aligned}}{3C(f) = 3X(C) + 3A(C) \geq 7n - 25 - c}$$

□

One-way permutations w.r.t circuit complexity

S_{2^n} is the subset of $B_{n,n}$ (the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}^n$) containing all $2^n!$ invertible functions.
Any sequence $f_1, f_2, \ldots$ of functions $f_i \in S_{2^i}$ — a family of permutations denoted by $\{f_n\}$.

One-way permutations w.r.t circuit complexity

S_{2^n} is the subset of $B_{n,n}$ (the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}^n$) containing all $2^n!$ invertible functions.
Any sequence $f_1, f_2, \ldots$ of functions $f_i \in S_{2^i}$ — a family of permutations denoted by $\{f_n\}$.
Let's consider the following measure of feeble one-wayness:

$$M_F(f_n) = C(f_n^{-1})/C(f_n)$$

One-way permutations w.r.t circuit complexity

S_{2^n} is the subset of $B_{n,n}$ (the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}^n$) containing all $2^n!$ invertible functions.
Any sequence $f_1, f_2, \ldots$ of functions $f_i \in S_{2^i}$ — a family of permutations denoted by $\{f_n\}$.
Let's consider the following measure of feeble one-wayness:

$$M_F(f_n) = C(f_n^{-1})/C(f_n)$$

This can be compared with the measure of practical one-wayness:

$$M_P(f_n) = \log_2[C(f_n^{-1})]/\log_2[C(f_n)]$$

One-way permutations w.r.t circuit complexity

A family of permutations $\{f_n\}$ is said to be feebly-one-way of order k, for some constant $k > 1$, if

$$C(f_n) = \omega(1) \qquad \textit{and} \qquad M_F(f_n) \sim k$$

One-way permutations w.r.t circuit complexity

A family of permutations $\{f_n\}$ is said to be feebly-one-way of order k, for some constant $k > 1$, if

$$C(f_n) = \omega(1) \qquad \textit{and} \qquad M_F(f_n) \sim k$$

A family of permutations $\{f_n\}$ is said to be practically-one-way of order k, for some constant $k > 1$, if

$$C(f_n) = \omega(1) \qquad \textit{and} \qquad M_P(f_n) \sim k$$

One-way permutations w.r.t circuit complexity

A family of permutations $\{f_n\}$ is said to be feebly-one-way of order k, for some constant $k > 1$, if

$$C(f_n) = \omega(1) \qquad \textit{and} \qquad M_F(f_n) \sim k$$

A family of permutations $\{f_n\}$ is said to be practically-one-way of order k, for some constant $k > 1$, if

$$C(f_n) = \omega(1) \qquad \textit{and} \qquad M_P(f_n) \sim k$$

These definitions imply $C(f_n^{-1}) \sim k \cdot C(f_n)$ and $C(f_n^{-1}) = [C(f_n)]^{k \pm o(1)}$ respectively.

A linear family with feeble one-wayness of order $\frac{3}{2}$

Let's define ϕ_n, for $n \geq 3$ as a linear function:

$$\phi_n([x_1, \ldots, x_n]) = [y_1, \ldots, y_n]$$

where

$$y_i(x) = x_i \oplus x_{i+1} \qquad \text{for i} \neq n$$

$$y_n(x) = x_1 \oplus x_{\lceil n/2 \rceil} \oplus x_n$$

A linear family with feeble one-wayness of order $\frac{3}{2}$

Let's define ϕ_n, for $n \geq 3$ as a linear function:

$$\phi_n([x_1, \ldots, x_n]) = [y_1, \ldots, y_n]$$

where

$$y_i(x) = x_i \oplus x_{i+1} \qquad \text{for i} \neq n$$

$$y_n(x) = x_1 \oplus x_{\lceil n/2 \rceil} \oplus x_n$$

The inverse function ϕ_n^{-1} is given by:

$$x_i(y) = (y_1 \oplus \cdots \oplus y_{i-1}) \oplus (y_{\lceil n/2 \rceil} \oplus \cdots \oplus y_n) \qquad i \leq \lceil n/2 \rceil$$

$$x_i(y) = (y_1 \oplus \cdots \oplus y_{\lceil n/2 \rceil - 1}) \oplus (y_i \oplus \cdots \oplus y_n) \qquad i > \lceil n/2 \rceil$$

A linear family with feeble one-wayness of order $\frac{3}{2}$

Theorem

For all $n \geq 5$, the functions ϕ_n satisfy

$$C(\phi_n) = n + 1 \quad \textit{and} \quad C(\phi_n^{-1}) = \lfloor \frac{3}{2}(n-1) \rfloor$$

A linear family with feeble one-wayness of order $\frac{3}{2}$

Theorem

For all $n \geq 5$, the functions ϕ_n satisfy

$$C(\phi_n) = n + 1 \quad \textit{and} \quad C(\phi_n^{-1}) = \lfloor \frac{3}{2}(n-1) \rfloor$$

Proof

- *By considering independent realizations of the component function we get $C(\phi_n) \leq n + 1$.*

A linear family with feeble one-wayness of order $\frac{3}{2}$

Theorem

For all $n \geq 5$, the functions ϕ_n satisfy

$$C(\phi_n) = n + 1 \quad \text{and} \quad C(\phi_n^{-1}) = \lfloor \frac{3}{2}(n-1) \rfloor$$

Proof

- *By considering independent realizations of the component function we get $C(\phi_n) \leq n + 1$.*
- *By noticing that each $x_i(y)$ is a sum of at least $\lceil n/2 \rceil$ of the y_k's we get $C(\phi_n) \geq \lfloor \frac{3}{2}(n-1) \rfloor$*

A linear family with feeble one-wayness of order $\frac{3}{2}$

Theorem

For all $n \geq 5$, the functions ϕ_n satisfy

$$C(\phi_n) = n + 1 \quad \text{and} \quad C(\phi_n^{-1}) = \lfloor \frac{3}{2}(n-1) \rfloor$$

Proof

- *By considering independent realizations of the component function we get $C(\phi_n) \leq n + 1$.*
- *By noticing that each $x_i(y)$ is a sum of at least $\lceil n/2 \rceil$ of the y_k's we get $C(\phi_n) \geq \lfloor \frac{3}{2}(n-1) \rfloor$*
- *It can be easily verified that the previous two bounds are exact.*

Nonlinear family with feeble one-wayness of order 2

Remark

It is easy to modify the previous family to make it one-way of order 2 (still being linear). However, it is even simipler to construct a non-linear family.

Construction

The family ν_n results from composition $\beta_n(\alpha_n(x))$ of linear permutation $\alpha_n([x_1, \ldots, x_n]) = (z_1, \ldots, z_n)$ with a nonlinear permutation $\beta_n([z_1, \ldots, z_n]) = (y_1, \ldots, y_n)$, where:

$$z_i(x) = x_i \oplus x_{i+1} \quad \text{for i} \neq n; \quad z_n(x) = x_n$$

$$y_i(z) = z_i \quad \text{for i} \neq n; \quad y_n(z) = z_n \oplus [\overline{(z_1 \oplus \cdots \oplus z_{n-2})} \wedge z_{n-1}]$$

Nonlinear family with feeble one-wayness of order 2

Construction

The inverse permutations $\beta_n^{-1}([y_1, \ldots, y_n]) = (z_1, \ldots, z_n)$ and $\alpha_n^{-1}([z_1, \ldots, z_n]) = (x_1, \ldots, x_n)$ will be:

$$z_i(y) = y_i \quad \text{for i} \neq n; \quad z_n(y) = y_n \oplus [\overline{(y_1 \oplus \cdots \oplus y_{n-2})} \wedge y_{n-1}]$$

$$x_i(z) = z_i \oplus \cdots \oplus z_n \quad \text{for i} \neq n; \quad x_n(z) = z_n$$

Nonlinear family with feeble one-wayness of order 2

Construction

The composition of α_n and β_n yields $\nu_n(x) = \beta_n(\alpha_n(x))[y_1, \dots, y_n]$, and $\nu_n^{-1}(y) = \alpha_n^{-1}(\beta_n^{-1}(y)) = [x_1, \dots, x_n]$ where:

$$y_i(x) = x_i \oplus x_{i+1} \quad \text{for i} \neq n; \quad y_n(x) = x_n \oplus [\overline{(x_1 \oplus x_{n-1})} \wedge (x_{n-1} \oplus x_n)]$$

$$x_i(y) = (y_i \oplus \cdots \oplus y_n) \oplus [\overline{(y_1 \oplus \cdots \oplus y_{n-2})} \wedge y_{n-1}] \quad \text{for i} \neq n$$

$$x_n(y) = y_n \oplus [\overline{(y_1 \oplus \cdots \oplus y_{n-2})} \wedge y_{n-1}]$$

Theorem

For all $n \geq 4$, the functions ν_n satisfy

$$C(\nu_n) = n + 2 \quad \textit{and} \quad C(\nu_n^{-1}) = 2(n-1)$$

Conclusion

- The results described in the first two sections of this talk were obtained together with Alexander S. Kulikov and Arist Kojevnikov.

Conclusion

- The results described in the first two sections of this talk were obtained together with Alexander S. Kulikov and Arist Kojevnikov.
- Now we are working on improving the results of the last section (obtained by Alain Hiltgen) together with my advisor Edward A. Hirsch.

Conclusion

- The results described in the first two sections of this talk were obtained together with Alexander S. Kulikov and Arist Kojevnikov.
- Now we are working on improving the results of the last section (obtained by Alain Hiltgen) together with my advisor Edward A. Hirsch.
- It is not easy to improve the constant 2 in the last section, because you need to prove a nontrivial lower bound to do this.

Thank you for your attention!