Circuit size lower bounds and #SAT upper bounds through a general framework

Alexander Golovnev[a b], Alexander Kulikov[a,],
Alexander Smal[a], Suguru Tamaki[c]

August 25, 2016

[a] St. Petersburg Department of Steklov Institute of Mathematics of RAS
[b] New York University
[c] Kyoto University

Outline

Framework

Applications

Open problems

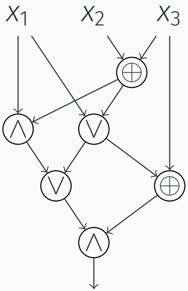
Outline

Framework

Applications

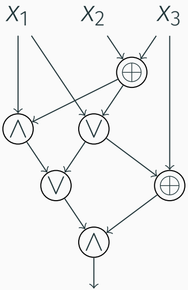
Open problems

Unbounded depth constant fan-in Boolean circuits.



Boolean circuits

Unbounded depth constant fan-in Boolean circuits.



Natural questions

- What is the smallest size of a circuit computing f?
- How many satisfiable assignments a circuit has?

What was known

Lower bounds

- Shannon: almost all functions has circuits of size $\Omega(2^n/n)$.

What was known

Lower bounds

- Shannon: almost all functions has circuits of size $\Omega(2^n/n)$.
- Worst case lower bounds:
  - $(3 + 1/86)n$ for B_2 [Find, Golovnev, Hirsch, Kulikov].
  - $5n - o(n)$ for U_2 [Iwama, Morizum].

What was known

Lower bounds

- Shannon: almost all functions has circuits of size $\Omega(2^n/n)$.
- Worst case lower bounds:
  - $(3 + 1/86)n$ for B_2 [Find, Golovnev, Hirsch, Kulikov].
  - $5n - o(n)$ for U_2 [Iwama, Morizum].
- Average case lower bounds:
  - $2.49n$ over B_2 and $2.99n$ for U_2 [Chen, Kabanets].

What was known

Lower bounds

- Shannon: almost all functions has circuits of size $\Omega(2^n/n)$.
- Worst case lower bounds:
  - $(3 + 1/86)n$ for B_2 [Find, Golovnev, Hirsch, Kulikov].
  - $5n - o(n)$ for U_2 [Iwama, Morizum].
- Average case lower bounds:
  - $2.49n$ over B_2 and $2.99n$ for U_2 [Chen, Kabanets].

Upper bounds (#SAT algorithms)

Faster than 2^n algorithms for circuits of size $2.49n$ over B_2 and $2.99n$ for U_2 [Chen, Kabanets].

Gate elimination and branching algorithms

Gate elimination

- Find substitution that eliminates many gates.
- Circuit size lower bound.

Gate elimination and branching algorithms

Gate elimination

- Find substitution that eliminates many gates.
- Circuit size lower bound.

Branching on substitutions

- Find substitution that eliminates many gates *in all branches*.
- #SAT algorithm.

Gate elimination and branching algorithms

Gate elimination

- Find substitution that eliminates many gates.
- Circuit size lower bound.

Branching on substitutions

- Find substitution that eliminates many gates *in all branches*.
- #SAT algorithm.

[Chen and Kabanets, 2014]

Branching case analysis can also be used to prove *average case circuit lower bounds (correlation bounds).*

Notation

- B_2 is a full Boolean binary basis, $U_2 = B_2 \setminus \{\oplus, \equiv\}$.
- $C_\Omega(f)$ the minimal size of a circuit over basis $\Omega \subseteq B_2$ computing function $f \in B_n$.

Notation

- B_2 is a full Boolean binary basis, $U_2 = B_2 \setminus \{\oplus, \equiv\}$.
- $C_\Omega(f)$ the minimal size of a circuit over basis $\Omega \subseteq B_2$ computing function $f \in B_n$.
- For functions $f, g \in B_n$, the correlation between them

$$\mathrm{Cor}(f, g) = \left| \Pr_x[f(x) = g(x)] - \Pr_x[f(x) \neq g(x)] \right| = \left| 1 - 2 \Pr_x[f(x) \neq g(x)] \right| .$$

Notation

- B_2 is a full Boolean binary basis, $U_2 = B_2 \setminus \{\oplus, \equiv\}$.
- $C_\Omega(f)$ the minimal size of a circuit over basis $\Omega \subseteq B_2$ computing function $f \in B_n$.
- For functions $f, g \in B_n$, the correlation between them

$$\mathrm{Cor}(f,g) = \left|\Pr_x[f(x) = g(x)] - \Pr_x[f(x) \neq g(x)]\right| = \left|1 - 2\Pr_x[f(x) \neq g(x)]\right|.$$

- For $0 \leq \varepsilon \leq 1$, $C_\Omega(f, \varepsilon)$ is the minimal size of a circuit over $\Omega \subseteq B_2$ computing function g such that $\mathrm{Cor}(f,g) \geq \varepsilon$.

A proof in the framework

1. Fix parameters:
    - a class of circuits $\mathcal{C}$,
    - a circuit complexity measure μ,
    - a set of allowed substitutions $\mathcal{S}$.

A proof in the framework

1. Fix parameters:
   - a class of circuits $\mathcal{C}$,
   - a circuit complexity measure μ,
   - a set of allowed substitutions $\mathcal{S}$.
2. Case analysis: for any circuit $C \in \mathcal{C}$ there is a substitution in $\mathcal{S}$ that reduces $\mu(C)$ by sufficient amount.

Framework

A proof in the framework

1. Fix parameters:
   - a class of circuits $\mathcal{C}$,
   - a circuit complexity measure μ,
   - a set of allowed substitutions $\mathcal{S}$.
2. Case analysis: for any circuit $C \in \mathcal{C}$ there is a substitution in $\mathcal{S}$ that reduces $\mu(C)$ by sufficient amount.
3. #SAT upper bound: branching algorithm.

Framework

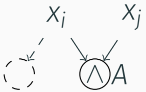
A proof in the framework

1. Fix parameters:
    - a class of circuits $\mathcal{C}$,
    - a circuit complexity measure μ,
    - a set of allowed substitutions $\mathcal{S}$.
2. Case analysis: for any circuit $C \in \mathcal{C}$ there is a substitution in $\mathcal{S}$ that reduces $\mu(C)$ by sufficient amount.
3. #SAT upper bound: branching algorithm.
4. Circuit size lower bounds for a function that survives under sufficiently many allowed substitutions.

Circuit complexity measures

We focus on the following two circuit complexity measures:

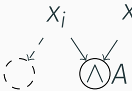
- $\mu(C) = s(C) + \alpha \cdot i(C)$ where $\alpha \geq 0$;



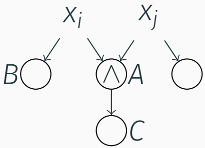
Circuit complexity measures

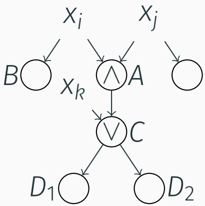
We focus on the following two circuit complexity measures:

- $\mu(C) = s(C) + \alpha \cdot i(C)$ where $\alpha \geq 0$;



- $\mu(C) = s(C) + \alpha \cdot i(C) - \sigma \cdot i_1(C)$ where $\alpha \geq 0, \sigma \leq 1$.





Sets of substitutions

1. Bit fixing substitutions: $\{x_i \leftarrow c\}$.
2. Projections: $\{x_i \leftarrow c, x_i \leftarrow x_j \oplus c\}$.
3. Affine substitutions: $\{x_i \leftarrow \bigoplus_{j \in J} x_j \oplus c\}$.
4. Quadratic substitutions: $\{x_i \leftarrow p : \deg(p) \leq 2\}$.

Sets of substitutions

1. Bit fixing substitutions: $\{x_i \leftarrow c\}$.
2. Projections: $\{x_i \leftarrow c, x_i \leftarrow x_j \oplus c\}$.
3. Affine substitutions: $\{x_i \leftarrow \bigoplus_{j \in J} x_j \oplus c\}$.
4. Quadratic substitutions: $\{x_i \leftarrow p : \deg(p) \leq 2\}$.

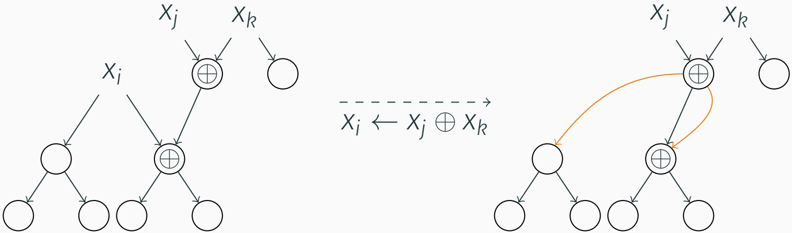
Dispersers and extractors

- $(\mathcal{S}, n, r)$-disperser f survives after $n - r$ substitutions.
- $(\mathcal{S}, n, r, \varepsilon)$-extractor f survives after $n - r$ substitutions

$$\left| \Pr_x [f_{subst}(x) = 1] - 1/2 \right| \leq \varepsilon.$$

- There are *explicit* constructions of dispersers and extractors allowing $n - o(n)$ substitutions *except for quadratic substitutions.*

Substitution replaces some variable by a function that is computed by some subcircuit of a given circuit.



Then we normalize the circuit removing trivialized gates.

Splitting vectors

Consider one step of branching algorithm A on circuit C.

- Choose k variables $x_1, \ldots, x_k$.
- Choose k functions $f_1, \ldots, f_k$ computed by gates of C.

Splitting vectors

Consider one step of branching algorithm A on circuit C.

- Choose k variables $x_1, \ldots, x_k$.
- Choose k functions $f_1, \ldots, f_k$ computed by gates of C.
- For all $c_1, \ldots, c_k \in \{0, 1\}$ substitute $x_1 \leftarrow f_1 \oplus c_1, \ldots,$ $x_k \leftarrow f_k \oplus c_k$ in C and call A on it.

$$\begin{tikzcd}
 & & A(C) \arrow[dll] \arrow[dl] \arrow[drr] & & \\
A(C_1) & A(C_2) & & \cdots & A(C_{2^k})
\end{tikzcd}$$

Splitting vectors

Consider one step of branching algorithm A on circuit C.

- Choose k variables $x_1, \dots, x_k$.
- Choose k functions $f_1, \dots, f_k$ computed by gates of C.
- For all $c_1, \dots, c_k \in \{0, 1\}$ substitute $x_1 \leftarrow f_1 \oplus c_1, \dots,$ $x_k \leftarrow f_k \oplus c_k$ in C and call A on it.

$$\begin{tikzcd}[column sep=small]
 & & A(C) \arrow[dll] \arrow[dl] \arrow[drr] & & \\
A(C_1) & A(C_2) & & \cdots & A(C_{2^k})
\end{tikzcd}$$

- Vector $v = (a_1, \dots, a_{2^k}) \in \mathbb{R}^{2^k}$ is a splitting vector w.r.t. measure μ if for all $i \in [2^k]$, $\mu(C) - \mu(C_i) \geq a_i > 0$.

Splitting numbers

For a splitting vector $v = (a_1, \ldots, a_{2^k})$ the splitting number $\tau(v)$ is the unique positive root of the equation $\sum_{i \in [2^k]} x^{-a_i} = 1$ (solution of $T(n) = T(n - a_1) + T(n - a_2) + \cdots + T(n - a_{2^k})$).

Splitting numbers

For a splitting vector $v = (a_1, \ldots, a_{2^k})$ the splitting number $\tau(v)$ is the unique positive root of the equation $\sum_{i \in [2^k]} x^{-a_i} = 1$ (solution of $T(n) = T(n - a_1) + T(n - a_2) + \cdots + T(n - a_{2^k})$).

Properties

- If splitting number is at most τ then running time is bounded by $O^*(\tau^{\mu(C)})$.
- Balanced splitting vectors (a, a) have better splitting numbers than unbalanced $(a + b, a - b)$.
- $2^{1/a} = \tau(a, a) < \tau(a + b, a - b)$ for $0 < b < a$.

Splitting

For a class of circuits Ω (e.g., $\Omega = B_2$ or $\Omega = U_2$), a set of substitutions $\mathcal{S}$, and a circuit complexity measure μ, we write

$$\text{Splitting}(\Omega, \mathcal{S}, \mu) \preceq \{v_1, \ldots, v_m\}$$

as a shortcut for the following statement:

- For any normalized circuit C from the class Ω
- one can find in time poly($|C|$) a substitution from $\mathcal{S}$ whose splitting vector w.r.t. measure μ belongs to $\{v_1, \ldots, v_m\}$
- or a substitution that trivializes the output gate of C.

Note: a substitution always trivializes at least one gate and eliminates at least one variable.

Main theorem

If $\text{Splitting}(\Omega, \mathcal{S}, \mu) \preceq \{v_1, \ldots, v_m\}$ *and the longest splitting vector has length* 2^k*, then*

1. *There exists an algorithm solving #SAT for circuits over* Ω *in time* $O^*(\gamma^{\mu(C)})$*, where* $\gamma = \max_{i \in [m]}\{\tau(v_i)\}$.
2. *For* $(\mathcal{S}, n, r)$*-disperser f,*
$$\mu(f) \geq \beta_w(\{v_i\}) \cdot (r - k + 1).$$
3. *For* $(\mathcal{S}, n, r, \varepsilon)$*-extractor f,*
$$\mu(f, \delta) \geq \beta_a(\{v_i\}) \cdot r, \quad \delta = \varepsilon + \exp\left(-g(r, \{v_i\})\right).$$

Weak limitation theorem

If for some $\mathcal{S}$, $\text{Splitting}(\Omega, \mathcal{S}, s + \alpha i) \preceq \{(a_1, b_1), \ldots, (a_m, b_m)\}$*, such that* $\min_{i \in [m]} \max\{a_i, b_i\} = \omega(1)$ *then #SAT for circuits over bases Ω can be solved in time $O^*(2^{o(s)})$.*

Corollary

Due to the Sparsification Lemma such an algorithm even over the bases U_2 contradicts the Exponential Time Hypothesis.

Framework

Applications

Open problems

Lemma

For $0 \leq \sigma \leq 1/2$,

$$\text{Splitting}(U_2, \{x_i \leftarrow c, x_i \leftarrow x_j \oplus c\}, s + \alpha i - \sigma i_1) \preceq \{\ldots\}.$$

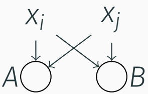
Corollary

1. $(2 - \delta(\epsilon))^n$ #SAT algorithm for circuits of size $(3.25 - \epsilon)n$.
2. $C_{U_2}(f) \geq 3.5n - \log^{O(1)}(n)$ for projections disperser f.
3. $C_{U_2}(f, \delta) \geq 3.25n - t$ for projections extractor f.

   $\text{Cor}(f, C)$ is negligible for C of size $3.25n - \omega(\sqrt{n \log n})$.

U_2: projections

There is only one case where projection $x_i \leftarrow x_j \oplus c$ is necessary (other cases are handled by bit fixing substitutions).



Let gates A and B compute Boolean functions
$f_A(x_i, x_j) = (x_i \oplus a_A)(x_j \oplus b_A) \oplus c_A$ and
$f_B(x_i, x_j) = (x_i \oplus a_B)(x_j \oplus b_B) \oplus c_B$ respectively.

- If $a_A = a_B$ ($b_A = b_B$) we assign $x_i \leftarrow a_A$ ($x_j \leftarrow b_A$).
- Otherwise, $x_i \leftarrow a_A \oplus x_j \oplus b_A \oplus 1$ makes A and B constant.

We get at least $(\alpha, 2\alpha)$ splitting vector.

B_2: quadratic substitutions

Lemma

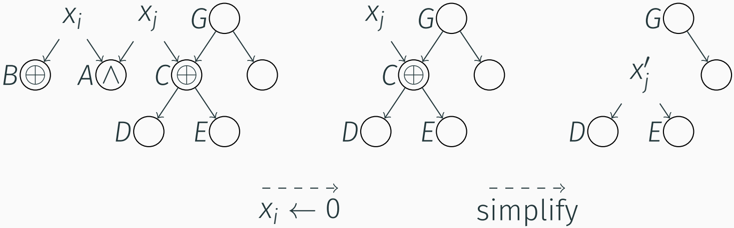
For $0 \le \sigma \le 1/5$*,*
$\mathrm{Splitting}(B_2, \{x_i \leftarrow p\colon\ \deg(p) \le 2\}, s + \alpha i - \sigma i_1) \preceq \{\dots\}$.

Corollary

1. $(2 - \delta(\epsilon))^n$ #SAT algorithms for circuits of size $(2.6 - \epsilon)n$.
2. $C_{B_2}(f) \ge 3n - o(n)$ for quadratic disperser f.
3. $C_{B_2}(f, \delta) \ge 2.6n - t$ for quadratic extractor f.

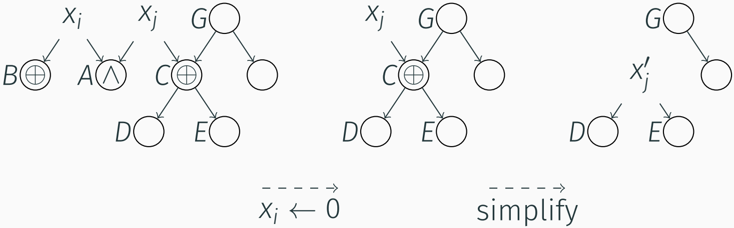
   $\mathrm{Cor}(f, C)$ is negligible for any circuit C of size $2.6n - g(n)$ for some $g(n) = o(n)$.

We can handle one of the cases in the case analysis differently.



B_2: improving #SAT algorithm

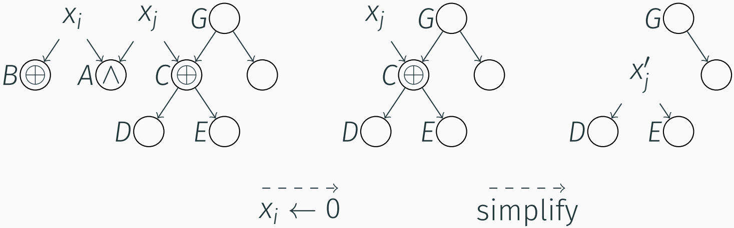
We can handle one of the cases in the case analysis differently.



$O((2-\delta(\epsilon))^n)$ #SAT algorithm for circuits of size $(3-\epsilon)n$.

B_2: improving #SAT algorithm

We can handle one of the cases in the case analysis differently.



$O((2-\delta(\epsilon))^n)$ #SAT algorithm for circuits of size $(3-\epsilon)n$.

This simplification *changes the function computed by a circuit.*

Outline

Open problems

1. Give an explicit construction of quadratic dispersers.
2. Adjust the framework to allow using natural simplification rules like replacing a xor gate fed by a 1-variable for both upper bounds and lower bounds.
3. Prove better limitation theorem.

Thanks for your attention!