Shuffles and Circuits (On Lower Bounds for Modern Parallel Computation)

Paper Reading in MPC Reading Group

Authors: Roughgarden, Vassilvitskii, Wang
Speaker: Zhuming Shi
Mentor: Shaofeng Jiang

Peking University MPC Reading Group
November 20, 2022

Outline

Authors

(a) Tim Roughgarden from Columbia

(b) Sergei Vassilvitskii from Stanford

(c) Joshua R. Wang from Stanford

Figure: Authors

Main Contributions

1. Definition of s-SHUFFLE model ~~of MPC~~
2. Computation using few rounds can be represented as low-degree polynomials over the reals. Degree n requires $\lceil \log_s n \rceil$ rounds
3. Lower bound is best result under infinity or polynomial number of machines
4. Apply to MapReduce
5. New machinery for proving lower bounds on the polynomial degree of Boolean functions

High level idea

1. Firstly, define s-SHUFFLE
2. Next, simulate MapReduce in s-SHUFFLE
3. Finally, lower bound s-SHUFFLE

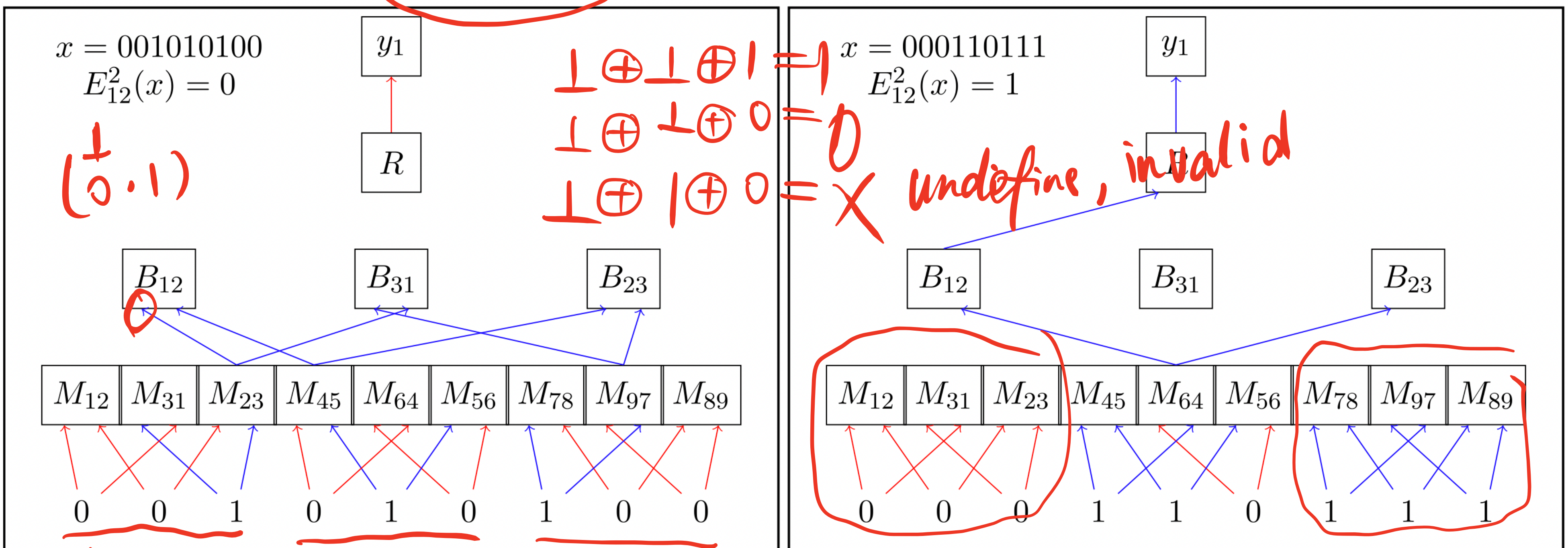
Warm Up

Example 2.1 (Silence Is Golden) Consider the Boolean function $E_{12} : \{0,1\}^3 \to \{0,1\}$ on three inputs that evaluates to 1 if and only if exactly one or two inputs are 1. Define $E_{12}^2 : \{0,1\}^9 \to \{0,1\}$ as the Boolean function that takes nine inputs, applies E_{12} to each block of three inputs, and then applies E_{12} to the results of the three blocks. For instance:

- $E_{12}^2(\overbrace{0,0,1}^{1},\overbrace{0,1,0}^{1},\overbrace{1,0,0}^{1}) = E_{12}(1,1,1) = 0$;
- $E_{12}^2(\underbrace{0,0,0}_{0},\underbrace{1,1,0}_{1},\underbrace{1,1,1}_{0}) = E_{12}(0,1,0) = 1.$

Warm Up

$\Leftarrow E_{12}(E_{12}(1,2,3), E_{12}(4,5,6), E_{12}(7,8,9))$



$\overset{\perp}{(0,1)}$

$\perp \oplus \perp \oplus 1 = 1$

$\perp \oplus \perp \oplus 0 = 0$

$\perp \oplus 1 \oplus 0 = \times$ undefine, invalid

1 2 3 4 5 6 7 8 9

$[0,1] M_j, \square$

s-SHUFFLE model

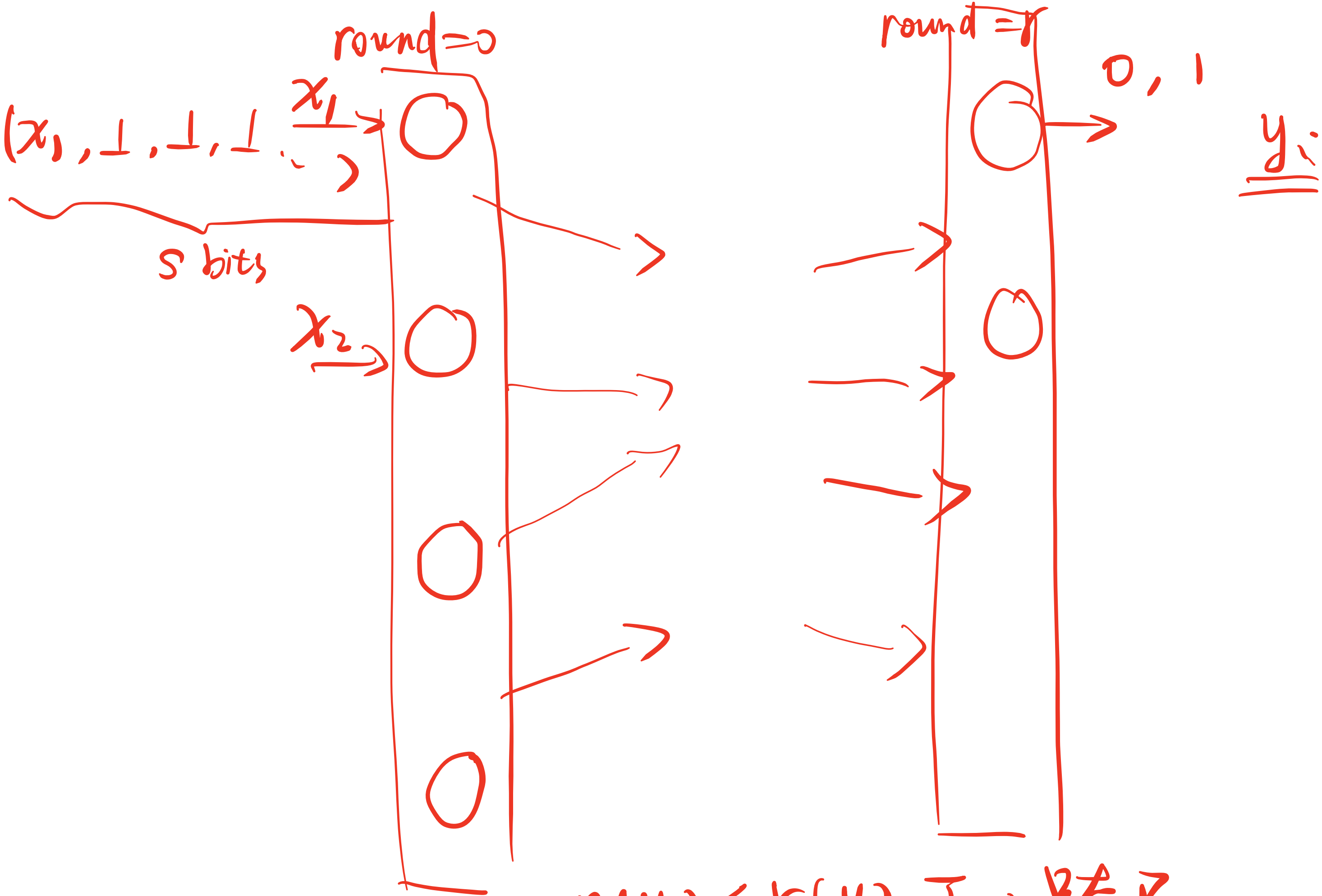
Definition 2.2 ($\perp$-sum) The $\perp$-*sum* of $z_1, z_2, \ldots, z_m \in \{0, 1, \perp\}$ is:

- 1 if exactly one z_i is 1 and the rest are $\perp$;
- 0 if exactly one z_i is 0 and the rest are $\perp$;
- $\perp$ if every z_i is $\perp$;
- undefined (or invalid) otherwise.

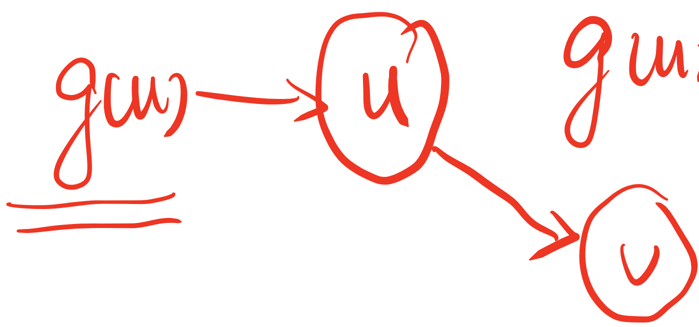
The $\perp$-sum of m s-tuples $a_1, \ldots, a_m$ is the entry-by-entry $\perp$-sum, denoted $\bigodot_{i=1}^{m} a_i$.

Definition 2.3 (**s-SHUFFLE Computation**) An *R-round s-SHUFFLE computation with inputs $x_1, \ldots, x_n$ and outputs $y_1, \ldots, y_k$* has the following ingredients:

1. A set V of *machines*, which includes one machine for each input bit x_i and each output bit y_i.
2. An assignment of a *round* $r(v)$ to each machine $v \in V$. Machines corresponding to input bits have round 0. Machines corresponding to output bits have round $R + 1$. All other machines have a round in $\{1, 2, \ldots, R\}$.
3. For each pair (u, v) of machines with $r(u) < r(v)$, a function α_{uv} from $\{0, 1, \perp\}^s$ to $\{0, 1, \perp\}^s$.

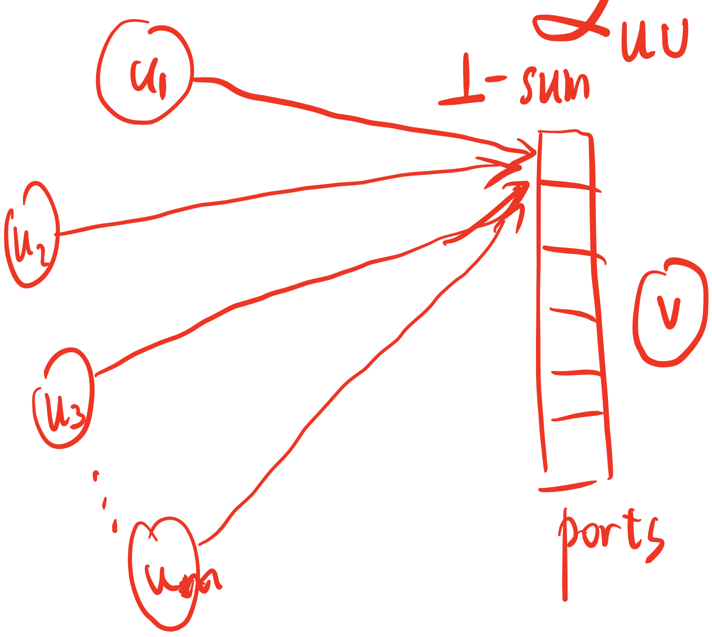


$r(u) < r(v)$ 可以跨层



$g(u)$: u机的input

$\alpha_{uv}(g(u))$: u给v的output



Definition 2.4 (Result of an s-SHUFFLE Computation) The *result* of an s-SHUFFLE computation assigns a value $g(v) \in \{0, 1, \perp\}^s$ to every machine $v \in V$, and is defined inductively as follows.

1. For a round-0 machine v, corresponding to an input bit x_i, the value $g(v)$ is the s-tuple $(x_i, \perp, \perp, \ldots, \perp)$.
2. Given the value $g(u)$ assigned to every machine u with $r(u) < q$, the value assigned to a machine v with $r(v) = q$ is the $\perp$-sum, over all machines u with $r(u) < r(v)$, of the message $\alpha_{uv}(g(u))$ sent to v by u:

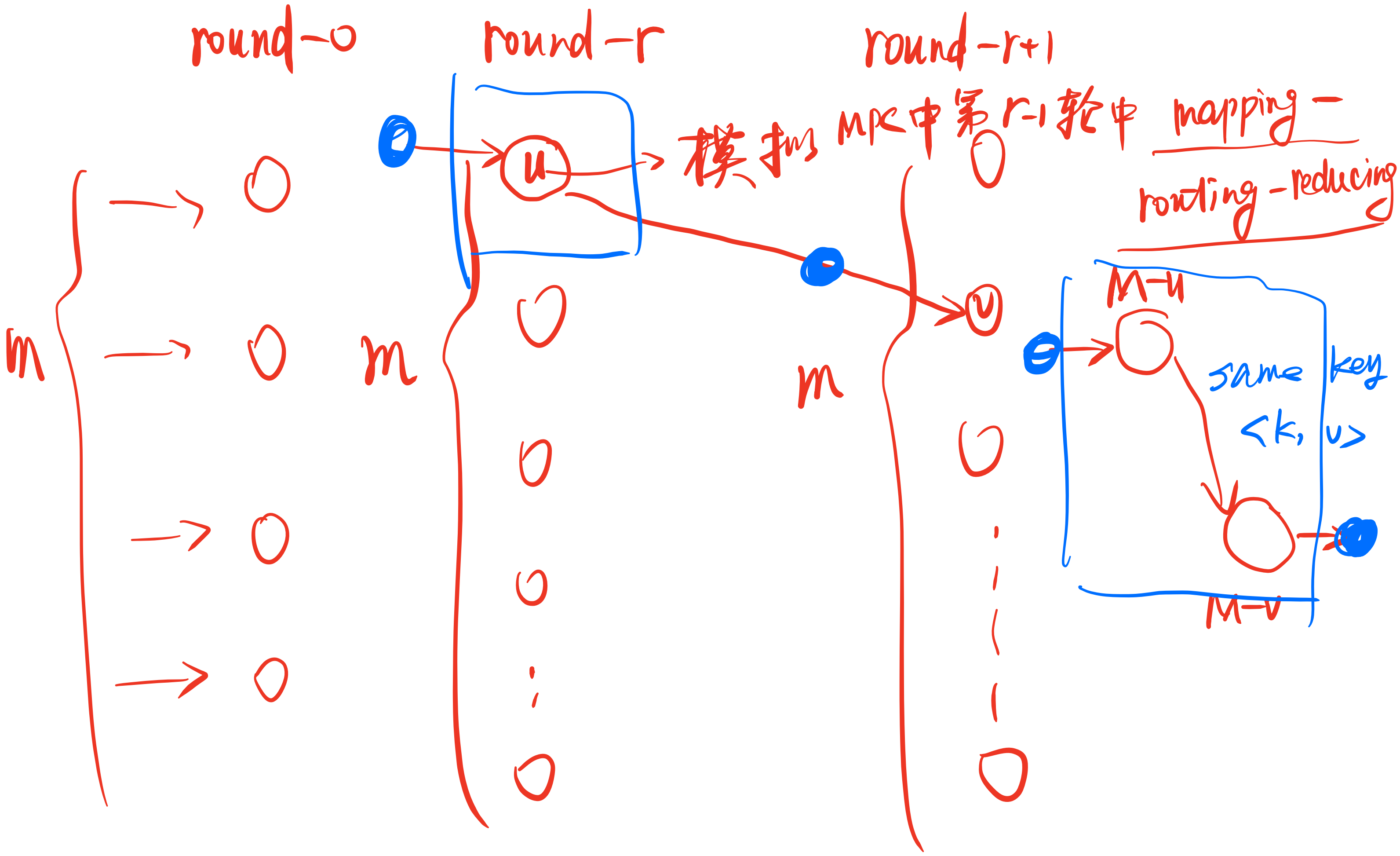
$$g(v) := \odot_{u\,:\,r(u)<r(v)} \alpha_{uv}(g(u)). \tag{1}$$

MapReduce simulated by s-SHUFFLE(Σ)

M-machine

S-machine

Proposition 2.7 (Simulating MapReduce). *Every r-round MapReduce computation with m machines and space s per machine can be simulated by an $(r+1)$-round s–*Shuffle*(Σ) computation with* $\boxed{(r+1)m}$ *machines and word size s.*


round-0
round-r
round-r+1
m
m
m
u
v
模拟MPC中第r-1轮中 mapping-routing-reducing
M-u
same key
<k, v>
M-v

Poly representation

low-round s-Shuffle computations with small s can only compute Boolean functions that can be represented as low-degree polynomials inframeMapReduce simulated by s-SHUFFLE(Σ)

Theorem 3.1 *Suppose that an s-*SHUFFLE *computation computes the function $f : \{0,1\}^n \to \{0,1\}^k$ in r rounds. Then there are k polynomials $\{p_i(x_1, \dots, x_n)\}_{i=1}^k$ of degree at most s^r such that $p_i(\mathbf{x}) = f(\mathbf{x})_i$ for all $i \in \{1, 2, \dots, k\}$ and $\mathbf{x} \in \{0,1\}^n$.*

The proof splits into some parts.

结论：$f: \{0,1\}^n \to \{0,1\}^k$，$\vec{x} \mapsto f(\vec{x})$

如果 f 不能表示为 degree $< d$ 的形式，

则至少在 S-SHUFFLE 上用 $\lceil \log_s d \rceil$ 轮计算

（无所谓机器数）

引理：给定机器 v，一个 string $z = \{0,1,\perp\}^s$

$\exists$ 多项式 $P_{v,z}(\vec{x}) = \begin{cases} 1 & \text{在 } \vec{x} \text{ input 下 } g(v) == z \\ 0 & \text{else} \end{cases}$

且 $P_{v,z}$ 的 degree $\leq s^{r(v)}$

证明：数学归纳法：归纳 $r(v)$ 的值（$0 \sim r$）

① 对 round-0 的机器 v，$z == (x_i, \perp, \perp, \cdots, \perp)$

if $z == (1, \perp, \perp, \cdots, \perp)$

$P_{v,z} = x_1$

if $z == (0, \perp, \perp, \cdots, \perp)$

$P_{v,z} = 1 - x_1$

degree $= 1$，成立 ✓

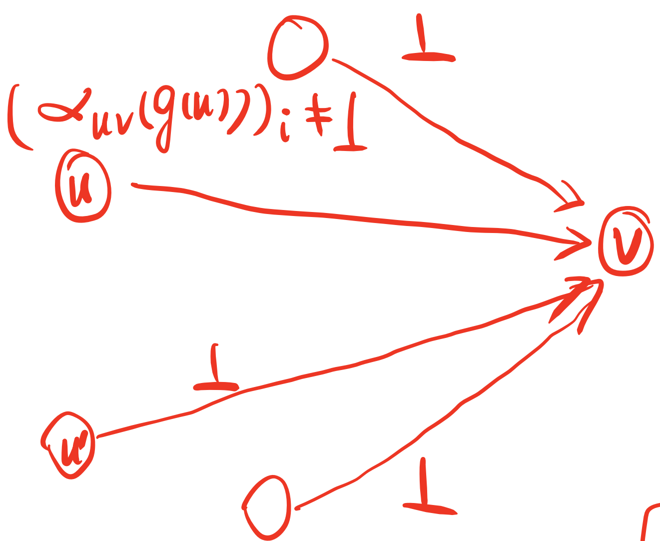
② round $\leq r(v)-1$ 的机器成立假设

2) round $= r(v)$ 的机器分析

$g(v)$ 只要判断每个 bit == z 的对应 bit

└→ 先分析 $(g(v))_i$ 单个 bit 的情况

(a) $(g(v))_i =$ none$-\perp$ 情况



$\exists\, u$, s.t. $(\alpha_{uv}(g(u)))_i \neq \perp$

$r(u) < r(v)$

原像 set: $\{\hat{z} \mid \alpha_{uv}(\hat{z}) == z_i\}$

$$\boxed{\vec{x} \Rightarrow g(u) == \hat{z}} \Rightarrow \alpha_{uv}(g(u)) == z_i$$

degree $\leq s^{r(u)} \leq s^{r(v)-1}$

$$\vec{x} \Rightarrow g(u') == \hat{z}' \Rightarrow \alpha_{u'v}(g(u')) == \perp$$

只需要 degree $\leq s^{r(v)-1}$ 的一些多项式

$P_{u,\hat{z}}(\vec{x})$，来判断 $(g(u))_i == z_i$

$$\left(\sum_{\hat{z}\text{使得}(\alpha_{uv}(\hat{z}))_i = z_i} P_{u,\hat{z}}(\vec{x}) + \sum_{\substack{r(u')<r(v) \\ \text{且 } u'\neq u}} \sum_{\substack{\hat{z}'\text{使得} \\ (\alpha_{u'v}(\hat{z}'))_i=\bot}} P_{u',\hat{z}'}(\vec{x}) \right)$$

用来 indicate $P_{v,z_i}(\vec{x})$

$(g(v))_i$ 是否 $== z_i$

且 degree $\leq s^{r(v)-1}$

(b) $(g(v))_i == \bot$

$$\sum_{r(u)<r(v)} \sum_{\substack{\hat{z}\text{使得} \\ \alpha_{uv}(\hat{z})==\bot}} P_{u,\hat{z}}(\vec{x}) \quad \text{degree} \leq s^{r(v)-1}$$

↓ 对每个 bit，都有 degree $s^{r(v)-1}$ 的判别式

则对 $\vec{g}(v) == \vec{z}$

$$P_{v,z}(\vec{x}) = \left(P_{v,z_1}(\vec{x})\right)\left(P_{v,z_2}(\vec{x})\right)\cdots\left(P_{v,z_s}(\vec{x})\right)$$

且 degree $\leq s^{r(v)}$ 成立

round-r 后：输出机 v

$P_{v,z}(\vec{x}) \Rightarrow v$ 的 output

$\downarrow$

degree $\le s^r$ 的 poly

$\downarrow$

闭 可以写出 $\forall \vec{x}$ 的 output

$\Rightarrow f: \{0,1\}^n \to \{0,1\}^k$ degree $\ge d$

至少要 $d \le s^r \Rightarrow r \ge \lceil \log_s d \rceil$ 轮

---

* 对于 $s = n^\varepsilon$, $s = \log n$.

a function fan-ins per machine, r rounds.

g degree $\le s^r \Rightarrow$ input degree n

Proof

至少 $\lceil \log_{n^\varepsilon} n \rceil = \frac{1}{\varepsilon}$ rounds

$s = \sqrt{n}$ $\lceil \log_{\sqrt{n}} n \rceil = 2$ best bound，不可改进

PROOF. We proceed by induction on the number of rounds. We claim that for every non-output machine $v \in V$ and value $\mathbf{z} \in \{0, 1, \perp\}^s$, there is a polynomial $p_{v,\mathbf{z}}(x_1, \ldots, x_n)$ that evaluates to 1 on points $\mathbf{x}$ for which the computation's assigned value $g(v)$ to v is $\mathbf{z}$ and to 0 on all other points $\mathbf{x} \in \{0, 1\}^n$. Furthermore, $p_{v,\mathbf{z}}$ has degree at most $s^{r(v)}$.

machine number - independent ↑

↓ width ⇒ 机器数量

① ∞ 机器 ⇒ tight $\lceil \log_5 n \rceil$

② 机器数 poly(n)

(lower bounded larger than $\Theta(\log_5 n)$ for P

separate NC' from P

Thank you!