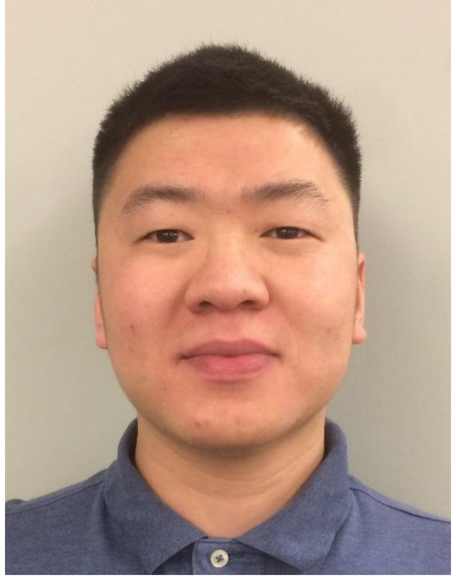
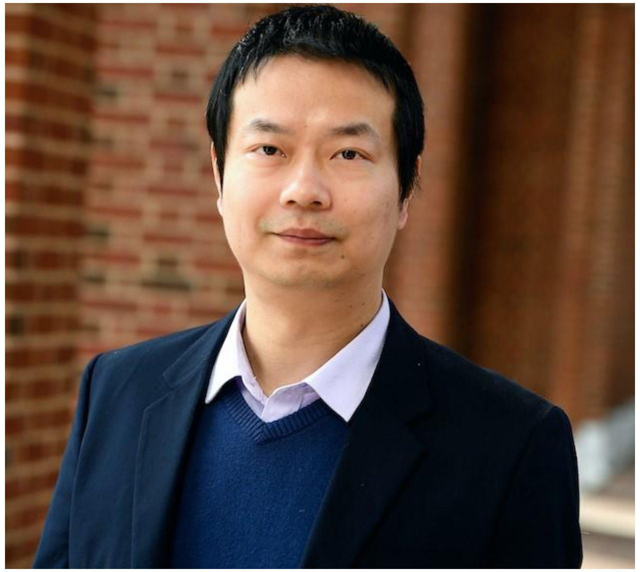
Explicit Min-wise Hash Families with Optimal Size

Xue Chen*



Shengtang Huang*

Xin Li†



* University of Science and Technology of China

† Johns Hopkins University

Min-wise Hash Family

Definition (Min-wise Hash Family [Broder, Charikar, Frieze, Mitzenmacher ’00])

- We say $\mathcal{H} = \{h : [N] \to [M]\}$ is a min-wise hash family with **multiplicative** error δ, if for any $X \subseteq [N]$ and $y \in X$,

$$\Pr_{h\sim\mathcal{H}}[h(y) < \min h(X\backslash y)] := \Pr_{h\sim\mathcal{H}}\left[h(y) < \min_{x\in X\backslash y} h(x)\right] = \frac{1 \pm \delta}{|X|}.$$

- Play a crucial role in the design of graph algorithms and streaming algorithms. Applications in similarity estimation [Cohen, Datar, Fujiwara, Gionis, Indyk, Motwani, Ullman, Yang ’01], ℓ_0-sampler [Cormode, Firmani ’14], etc.

k-min-wise Hash Family

Definition (k-min-wise Hash Family [Feigenblat, Porat, Shiftan '11])

- We say $\mathcal{H} = \{h : [N] \to [M]\}$ is a k-min-wise hash family with **multiplicative** error δ, if for any $X \subseteq [N]$ and $Y \in \binom{X}{\leq k}$,

$$\Pr_{h\sim\mathcal{H}}[\max h(Y) < \min h(X \backslash Y)] \coloneqq \Pr_{h\sim\mathcal{H}}\left[\max_{y\in Y} h(y) < \min_{x\in X\backslash Y} h(x)\right] = \frac{1 \pm \delta}{\binom{|X|}{|Y|}}.$$

- Play a crucial role in the design of graph algorithms and streaming algorithms. Applications in similarity estimation [Cohen, Datar, Fujiwara, Gionis, Indyk, Motwani, Ullman, Yang '01], ℓ_0-sampler [Cormode, Firmani '14], etc.

Problem – A Good Min-wise Hash Family

- Two ways for evaluating the quality of a hash family:

TIME

- **Fast Evaluation Time:** time complexity to compute $h(x)$ from a given input $x \in [N]$ for any hash function $h \in \mathcal{H}$.

SPACE

- **Small size** $|\boldsymbol{\mathcal{H}}|$, or equivalently $\text{seed length} = \log_2 |\mathcal{H}|$ (number of random bits used to generate a hash function).

- **Goal:** Construct an explicit (k-)min-wise hash family with short seed length.

EXPLICITNESS

- There exists an efficient ($\text{poly}(N)$-time) algorithm to compute $h(x)$ from a given input $x \in [N]$ and the seed of the hash $h \in \mathcal{H}$.

Prior Works

- [Saks, Srinivasan, Zhou, Zuckerman '00]: $O\left(\log^{3/2} N\right)$ bits for any polynomially small error.
- This was improved to $O(\log N \log\log N)$ by [Gopalan, Yehudayoff '20].

- [Indyk '01]: $O(\log(1/\delta))$-wise independence is enough to have error δ. $O(\log(1/\delta) \cdot \log N)$ seed length.
- [Feigenblat, Porat, Shiftan '11]: $O(\log(1/\delta) + k \log\log(1/\delta))$-wise independence is sufficient for k-min-wise hash.
- [Pătraşcu, Thorup '16]: $\Omega(\log(1/\delta))$-wise independence is needed for min-wise hash.

- **Non-explicitly**, $O(\log(N/\delta))$ bits for min-wise and $O(k \log N + \log(1/\delta))$ bits for k-min-wise.
- **No** construction with $O(\log N)$ bits for min-wise (and $O(k \log N)$ bits for k-min-wise) and sub-constant error was known before.

Our Results

- We give an explicit min-wise hash family with seed length $O(\log N)$ and error $2^{-O\left(\frac{\log N}{\log\log N}\right)}$, as well as an explicit k-min-wise hash family with seed length $O(k\log N)$ for any $k=\log^{O(1)} N$.

- Reduce the space complexity of algorithms that use min-wise hash families.

Seed length with error $2^{-O\left(\frac{\log N}{\log\log N}\right)}$	Min-wise hash	k-min-wise hash $\left(k=\log^{O(1)} N\right)$
[Ind01] && [FPS11]	$O\left(\frac{\log^2 N}{\log\log N}\right)$	$O\left(\frac{\log^2 N}{\log\log N}+k\log N\log\log N\right)$
[SSZZ00] && [GY20]	$O(\log N\log\log N)$	$O(k\log N\log\log N)$
Our Results	$\boldsymbol{O(\log N)}$	$\boldsymbol{O(k\log N)}$

Polynomially Small Error

[Saks, Srinivasan, Zhou, Zuckerman ’00] && [Gopalan, Yehudayoff ’20]

Hash Family as Pseudorandom Generator

Definition (Pseudorandom Generator)

- $G: \{0,1\}^r \to [M]^N$ is an ε-PRG for a family of functions $\mathcal{F} = \{f: [M]^N \to \{0,1\}\}$ if for any $f \in \mathcal{F}$,
$$\left|\mathrm{E}\left[f\left(U_{[M]^N}\right)\right] - \mathrm{E}[f(G(U_r))]\right| \le \varepsilon.$$
- r is called the seed length of G.

$$\mathcal{H} = \{h: [N] \to [M]\} \;\leftrightarrow\; G: \{0,1\}^{\log_2 |\mathcal{H}|} \to [M]^N$$
$$G(s) = \left(h_s(1), h_s(2), \dots, h_s(N)\right)$$

Decompose into Sum of Combinatorial Rectangles

Definition (Combinatorial Rectangle)

- We say $f : \Sigma^d \to \{0,1\}$ is a Σ^d-combinatorial-rectangle, if there exists $A_1, \ldots, A_d \subseteq \Sigma$ such that $f(x_1, \ldots, x_d) = \prod_{i=1}^{d} 1(x_i \in A_i)$.

- Note that (analysis in [Indyk '01], [Saks, Srinivasan, Zhou, Zuckerman '00])

$$\Pr_{h\sim\mathcal{H}}[h(y) < \min h(X\backslash y)] = \sum_{\theta=1}^{M} \Pr_{h\sim\mathcal{H}}[\underbrace{h(y) = \theta \wedge \min h(X\backslash y) > \theta}]$$

$$A_y = \{\theta\}$$

$$A_x = \{\theta+1, \cdots, M\}, \forall x \in X\backslash y$$

$[M]^{|X|}$- combinatorial-rectangle

Connection with PRG for Combinatorial Rectangles

- If $\mathcal{H}$ is an ε-PRG for $[M]^N$-combinatorial-rectangles, then $\forall X \subseteq [N]$ and $y \in X$,

$$\Pr_{h\sim\mathcal{H}}[h(y) < \min h(X\backslash y)] = \sum_{\theta=1}^{M} \Pr_{h\sim\mathcal{H}}[h(y) = \theta \wedge \min h(X\backslash y) > \theta]$$
$$= \sum_{\theta=1}^{M} \Pr_{h\sim U}[h(y) = \theta \wedge \min h(X\backslash y) > \theta] \pm M\varepsilon$$
$$= \Pr_{h\sim U}[h(y) < \min h(X\backslash y)] \pm M\varepsilon$$
$$= \Pr_{h\sim U}[h(y) < \min h(X\backslash y)] \cdot (1 \pm |X| \cdot M\varepsilon)$$

$$\Pr_{h\sim U}[h(y) < \min h(X\backslash y)] \approx 1/|X|$$
$$\text{if } M = \Omega(N/\delta)$$

Conclusion

- An ε-PRG for $[M]^N$-combinatorial-rectangles is a min-wise hash family with multiplicative error $\Theta(NM\varepsilon)$.

Connection with PRG for Combinatorial Rectangles

Theorem [Gopalan, Yehudayoff '20]

- There exists an explicit ε-PRG for $[M]^N$-combinatorial-rectangles with seed length $O(\log(M \log N / \varepsilon) \log\log(M/\varepsilon))$.

Corollary

- A min-wise hash family of $O(\log N \log\log N)$ bits for any polynomially small error.

Gap between Multiplicative and Additive Errors

$$\Pr_{h\sim\mathcal{H}}[h(y) < \min h(X\backslash y)] = \Pr_{h\sim U}[h(y) < \min h(X\backslash y)] \cdot (1 \pm \delta) \text{ if } M = \Omega(N/\delta)$$

- Min-wise hash family is a PRG with ***multiplicative*** error!
- **Problem:** additive → multiplicative, $\varepsilon \to \Theta(NM\varepsilon)$. Even for constant δ, $\varepsilon = \delta/(NM)$ would be polynomially small.
- If we want shorter seed length, like $O(\log N)$, then it is not enough to apply the PRG for combinatorial rectangles directly.

Optimal Size && Sub-constant Error

Our Approach [Chen, Huang, Li ’2025]

1st Step – Balls into Bins

- Use constant-wise independence to split X into several tiny blocks s.t. each block only needs a small size family.

2nd Step – Recycle Randomness

- Use the Nisan-Zuckerman PRG and a special extractor to recycle randomness between blocks.

3rd Step – Domain Reduction

- Use PRGs for combinatorial rectangles and the special extractor to further reduce the error.

$$\Pr_{h\sim\mathcal{H}}[h(y) < \min h(X\backslash y)] = \frac{1 \pm \delta}{|X|}$$

$$= \sum_{\theta=1}^{M} \Pr_{h\sim\mathcal{H}}[h(y) = \theta \wedge \min h(X\backslash y) > \theta]$$

Two Level Hash Structure

$h(x) = \sigma_{\varphi(x)}(x)$

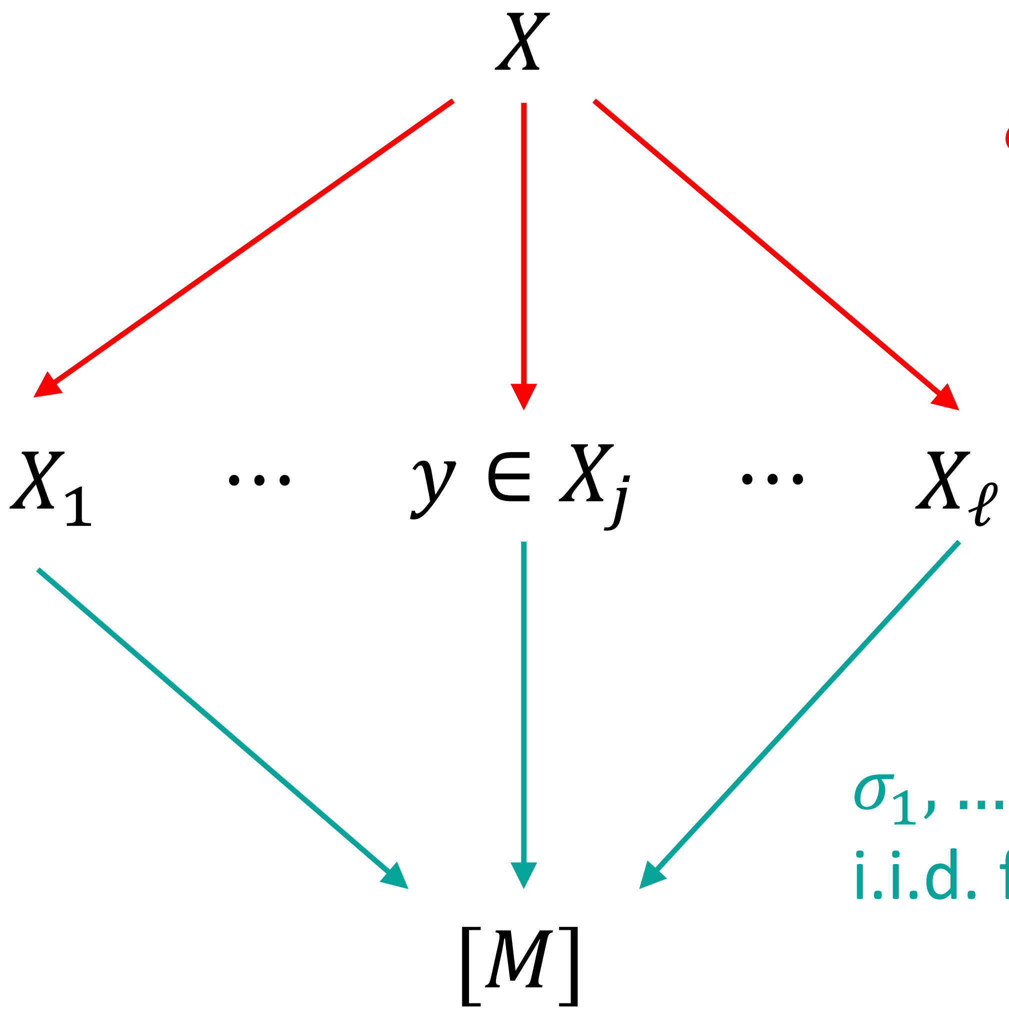
- **Balls into Bins:** small max-load, concentration.

  $|X_i| \approx |X|/\ell, \forall i \in [\ell]$

  $\max |X_i|$ is small

- **Each Block:** Since $|X_i|$ is small, there exists $\mathcal{G} = \{\sigma: [N] \to [M]\}$ with $|\mathcal{G}| = \mathrm{poly}(N)$, s.t.

  $$\Pr_{\sigma_i \sim \mathcal{G}}[\min \sigma_i(X_i) > \theta] \approx \Pr_{h \sim U}[\min h(X_i) > \theta]$$



$\varphi: [N] \to [\ell]$

Constant-wise independence

$\sigma_1, \dots, \sigma_\ell: [N] \to [M]$ i.i.d. from $\mathcal{G}$

Split into Several Small Blocks

$$h(x) = \sigma_{\varphi(x)}(x)$$

$$\Pr_{h\sim\mathcal{H}}[h(y) = \theta \wedge \min h(X\backslash y) > \theta]$$

$$\|$$

$$\varphi: [N] \to [\ell]$$

$$\Pr_{\sigma_1\sim\mathcal{G}}[\min \sigma_1(X_1) > \theta] \cdots \Pr_{\sigma_j\sim\mathcal{G}}\left[\sigma_j(y) = \theta \wedge \min \sigma_j\left(X_j\backslash y\right) > \theta\right] \cdots \Pr_{\sigma_\ell\sim\mathcal{G}}[\min \sigma_\ell(X_\ell) > \theta]$$

$$\boxed{\begin{aligned}&\Pr_{\sigma_i\sim\mathcal{G}}[\min \sigma_i(X_i) > \theta]\\ &\approx \Pr_{h\sim U}[\min h(X_i) > \theta]\end{aligned}}$$

$$\wr\wr$$

! Require many random bits.

$\sigma_1, \ldots, \sigma_\ell: [N] \to [M]$

i.i.d. from $\mathcal{G}$

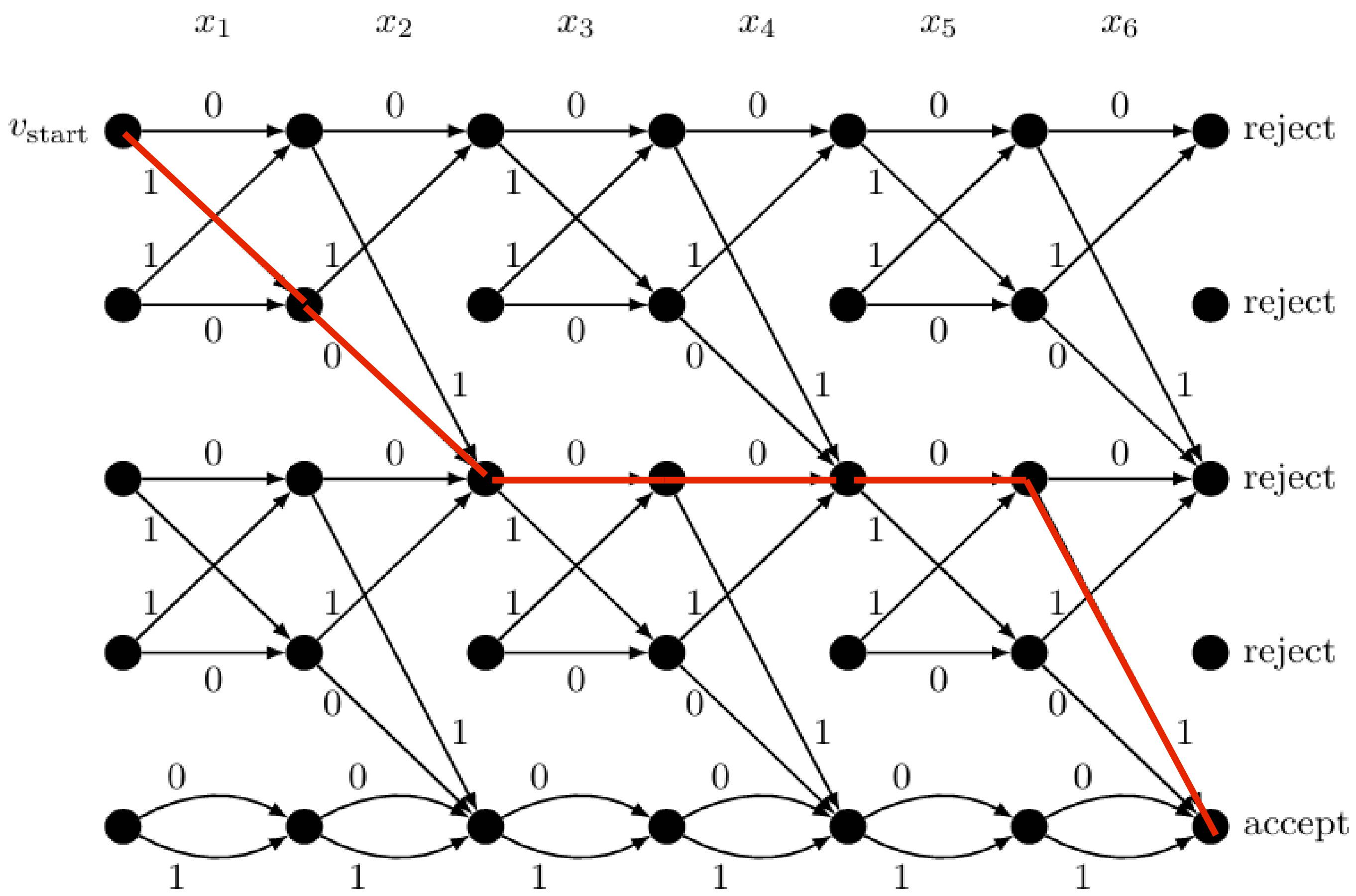
$$\Pr_{h\sim U}[\min h(X_1) > \theta] \cdots \Pr_{h\sim U}\left[h(y) = \theta \wedge \min h\left(X_j\backslash y\right) > \theta\right] \cdots \Pr_{h\sim U}[\min h(X_\ell) > \theta]$$

$$= \Pr_{h\sim U}[h(y) = \theta \wedge \min h(X\backslash y) > \theta]$$

Read-once Branching Program

Width: 5
Length: 6
Alphabet: {0,1}

$(x_1, x_2, x_3, x_4, x_5, x_6) = (1{,}0{,}0{,}0{,}0{,}1)$
$\mathrm{MAJ}(x_1 \oplus x_2, x_3 \oplus x_4, x_5 \oplus x_6) = 1$

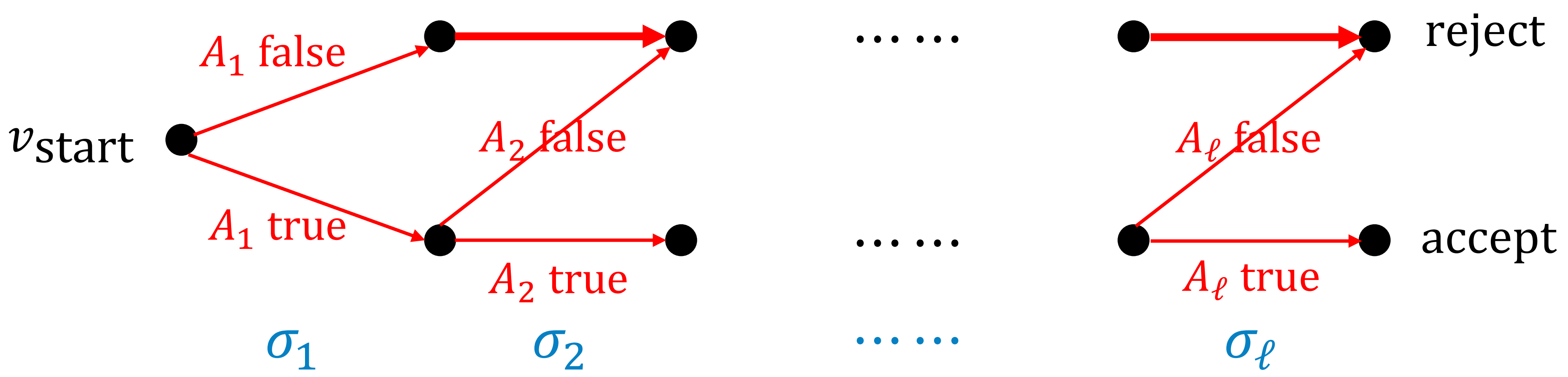


$\mathrm{MAJ}(x_1 \oplus x_2, x_3 \oplus x_4, x_5 \oplus x_6)$

Recycle Randomness

$$\underbrace{(\min \sigma_1(X_1) > \theta)}_{\text{Event } A_1} \wedge \cdots \wedge \underbrace{\left(\sigma_j(y) = \theta \wedge \min \sigma_j\left(X_j \backslash y\right) > \theta\right)}_{\text{Event } A_j} \wedge \cdots \wedge \underbrace{(\min \sigma_\ell(X_\ell) > \theta)}_{\text{Event } A_\ell}$$

$\sigma_1, \dots, \sigma_\ell$ i.i.d. from $\mathcal{G}$



Width: 2
Length: ℓ
Alphabet: $\mathcal{G}$

Recycle Randomness

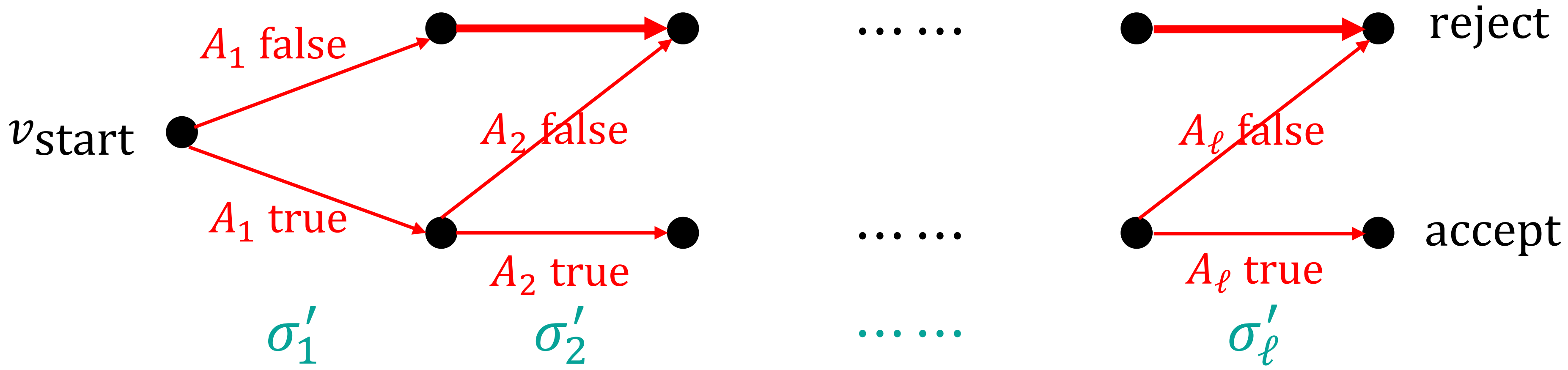
$$\underbrace{(\min \sigma_1(X_1) > \theta)}_{\text{Event } A_1} \wedge \cdots \wedge \underbrace{\left(\sigma_j(y) = \theta \wedge \min \sigma_j\left(X_j \backslash y\right) > \theta\right)}_{\text{Event } A_j} \wedge \cdots \wedge \underbrace{(\min \sigma_\ell(X_\ell) > \theta)}_{\text{Event } A_\ell}$$

$\sigma_1, \ldots, \sigma_\ell$ i.i.d. from $\mathcal{G}$

$$\approx$$

$$(\min \sigma_1'(X_1) > \theta) \wedge \cdots \wedge \left(\sigma_j'(y) = \theta \wedge \min \sigma_j'\left(X_j \backslash y\right) > \theta\right) \wedge \cdots \wedge (\min \sigma_\ell'(X_\ell) > \theta)$$

$\sigma_1', \ldots, \sigma_\ell'$ are correlated



Width: 2
Length: ℓ
Alphabet: $\mathcal{G}$

Nisan-Zuckerman PRG

Definition (Extractor)

- $\mathrm{Ext}: \{0,1\}^p \times \{0,1\}^d \to \{0,1\}^q$ is a (k,ε)-extractor, if for any random source X over $\{0,1\}^p$ with min-entropy $H_\infty(X) \geq k$, it holds that $\mathrm{Ext}(X, U_d)$ is ε-close to U_q.

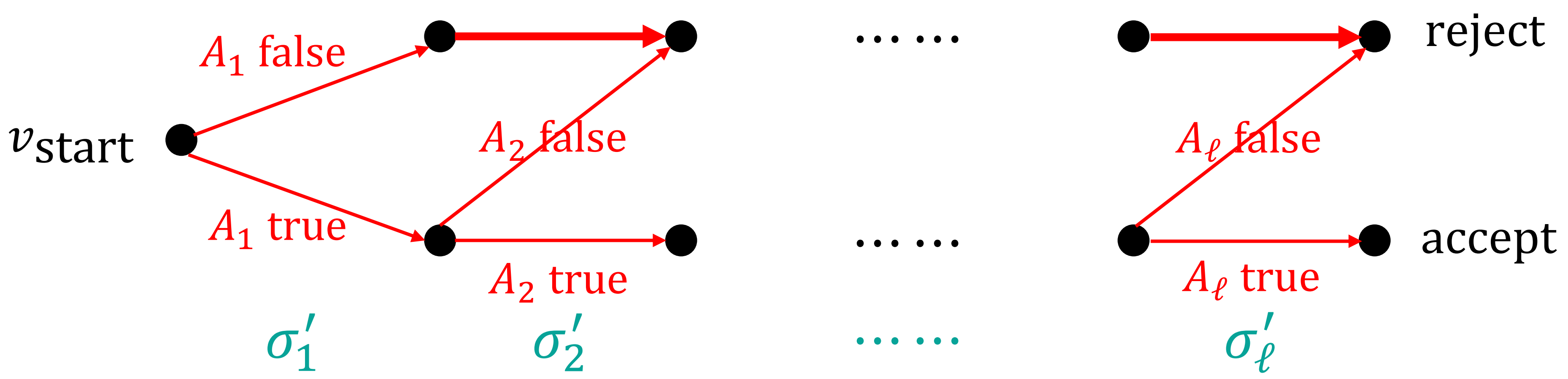
- Nisan-Zuckerman PRG:

$$\mathrm{NZPRG}(w, s_1, \dots, s_\ell) = \left(\mathrm{Ext}(w, s_1), \dots, \mathrm{Ext}(w, s_\ell)\right) \in (\{0,1\}^q)^\ell,$$
$$w \sim U_p, s_1, \dots, s_\ell \text{ i.i.d. from } U_d.$$

Nisan-Zuckerman PRG

$$\underbrace{(\min \sigma_1'(X_1) > \theta)}_{\text{Event } A_1} \wedge \cdots \wedge \underbrace{\left(\sigma_j'(y) = \theta \wedge \min \sigma_j'\left(X_j \backslash y\right) > \theta\right)}_{\text{Event } A_j} \wedge \cdots \wedge \underbrace{(\min \sigma_\ell'(X_\ell) > \theta)}_{\text{Event } A_\ell}$$

$$\boxed{\begin{array}{c} \text{Ext}: \{0,1\}^p \times \{0,1\}^d \to \mathcal{G} \\ \sigma_1' = \text{Ext}(w, s_1), \ldots, \sigma_\ell' = \text{Ext}(w, s_\ell) \\ w \sim U_p, s_1, \ldots, s_\ell \text{ i.i.d. from } U_d \end{array}}$$



Width: 2
Length: ℓ
Alphabet: $\mathcal{G}$

Analysis

$$P_1[\mathrm{NZ}]$$

$$\wr\wr$$

$$(P_1[U] \pm \mathrm{ExtErr})$$

$H_\infty(w = U_p) \geq k$
$\Rightarrow \mathrm{Ext}(w, s_1) \approx_{\mathrm{ExtErr}} \mathcal{G}$

$$\begin{aligned} P_1[\mathrm{NZ}] &= P_1[\mathcal{G}] \pm \mathrm{ExtErr} \\ &\approx P_1[U] \pm \mathrm{ExtErr} \end{aligned}$$

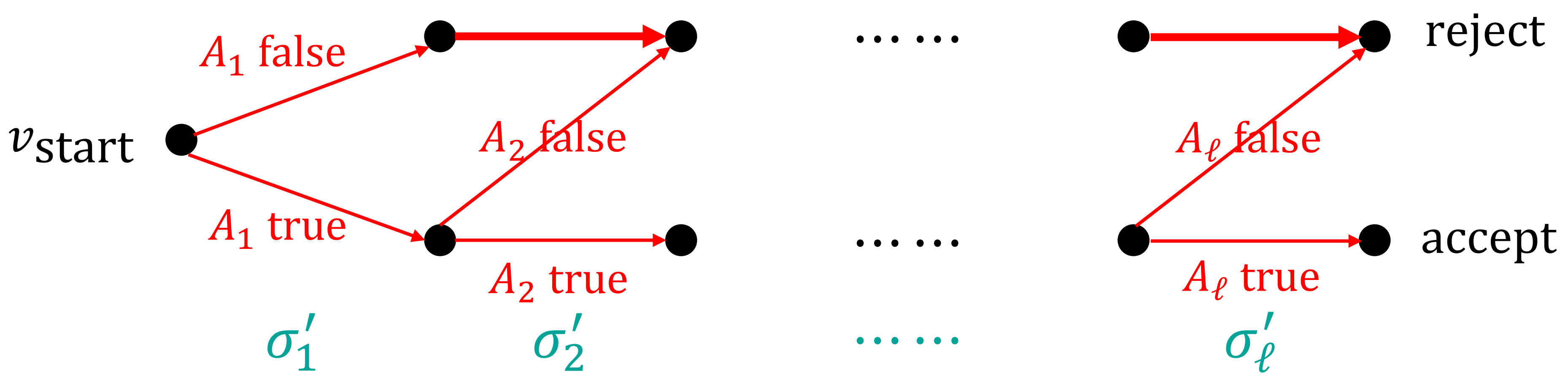
$\mathrm{Ext}: \{0,1\}^p \times \{0,1\}^d \to \mathcal{G}$
$\sigma_1' = \mathrm{Ext}(w, s_1), \dots, \sigma_\ell' = \mathrm{Ext}(w, s_\ell)$
$w \sim U_p, s_1, \dots, s_\ell$ i.i.d. from U_d

$P_i[\mathcal{G}] \approx P_i[U]$

$$P_i[\mathrm{NZ}] \coloneqq \Pr_{\sigma_i' : \mathrm{NZPRG}}[A_i]$$

$$P_i[\mathcal{G}] = \Pr_{\sigma_i \sim \mathcal{G}}[A_i]$$

$$P_i[U] \coloneqq \Pr_{h \sim U}[A_i]$$



Width: 2
Length: ℓ
Alphabet: $\mathcal{G}$

Analysis

$$P_1[\mathrm{NZ}] \cdot P_2[\mathrm{NZ}]$$

$$\wr\wr$$

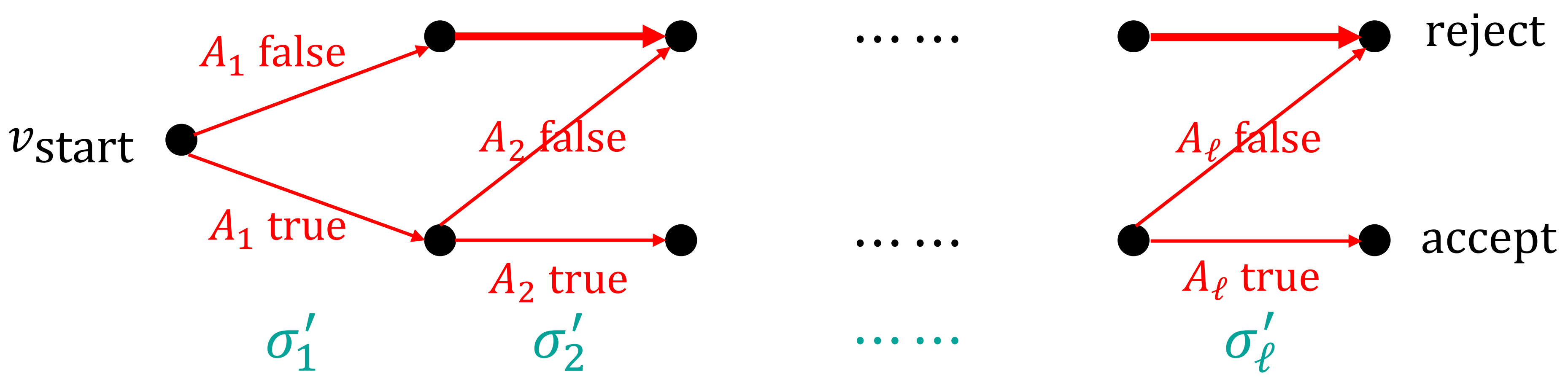
$$(P_1[U] \pm \mathrm{ExtErr}) \cdot (P_2[U] \pm \mathrm{ExtErr})$$

$$H_\infty(w \mid A_1) \geq k$$
$$\Rightarrow \mathrm{Ext}(w \mid A_1, s_2) \approx_{\mathrm{ExtErr}} \mathcal{G}$$

$$\boxed{\Pr[A_1 A_2] = \Pr[A_1] \cdot \Pr[A_2 \mid A_1]}$$

$$\mathrm{Ext}\colon \{0,1\}^p \times \{0,1\}^d \to \mathcal{G}$$
$$\sigma_1' = \mathrm{Ext}(w, s_1), \dots, \sigma_\ell' = \mathrm{Ext}(w, s_\ell)$$
$$w \sim U_p, s_1, \dots, s_\ell \text{ i.i.d. from } U_d$$

$$P_i[\mathcal{G}] \approx P_i[U]$$
$$P_i[\mathrm{NZ}] := \Pr_{\sigma_i' : \mathrm{NZPRG}}[A_i]$$
$$P_i[\mathcal{G}] = \Pr_{\sigma_i \sim \mathcal{G}}[A_i]$$
$$P_i[U] := \Pr_{h \sim U}[A_i]$$



Width: 2

Length: ℓ

Alphabet: $\mathcal{G}$

Analysis

$$P_1[\mathrm{NZ}] \cdot P_2[\mathrm{NZ}] \cdots P_\ell[\mathrm{NZ}]$$

$$\wr\wr$$

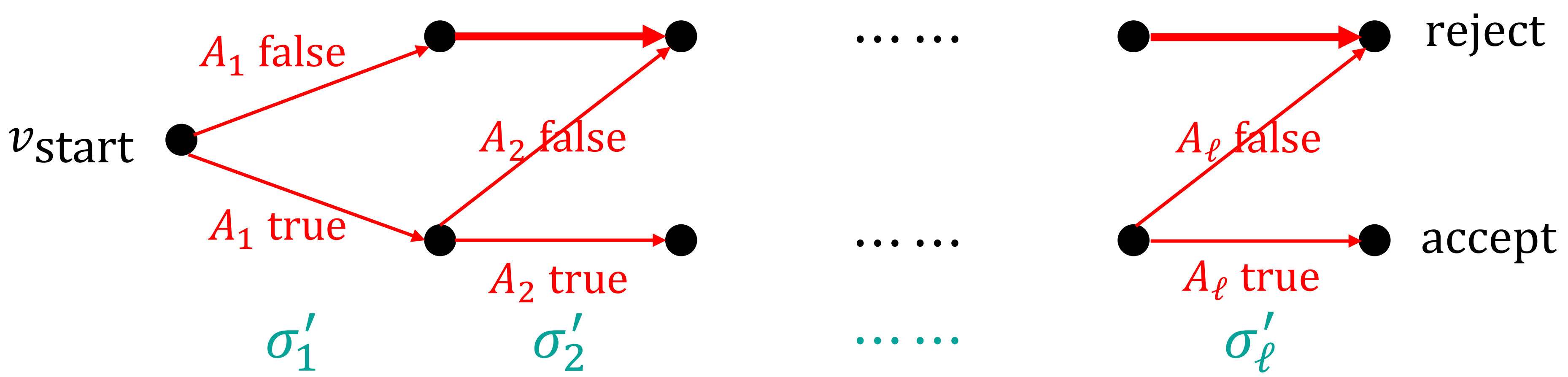
$$(P_1[U] \pm \mathrm{ExtErr}) \cdot (P_2[U] \pm \mathrm{ExtErr}) \cdots (P_\ell[U] \pm \mathrm{ExtErr})$$

$$\mathrm{Ext}\colon \{0,1\}^p \times \{0,1\}^d \to \mathcal{G}$$
$$\sigma_1' = \mathrm{Ext}(w, s_1), \dots, \sigma_\ell' = \mathrm{Ext}(w, s_\ell)$$
$$w \sim U_p, s_1, \dots, s_\ell \text{ i.i.d. from } U_d$$

$$P_i[\mathcal{G}] \approx P_i[U]$$
$$P_i[\mathrm{NZ}] \coloneqq \Pr_{\sigma_i' : \mathrm{NZPRG}}[A_i]$$
$$P_i[\mathcal{G}] = \Pr_{\sigma_i \sim \mathcal{G}}[A_i]$$
$$P_i[U] \coloneqq \Pr_{h \sim U}[A_i]$$

$$H_\infty(w \mid A_1 \cdots A_{\ell-1}) \geq k$$
$$\Rightarrow \mathrm{Ext}(w \mid A_1 \cdots A_{\ell-1}, s_\ell) \approx_{\mathrm{ExtErr}} \mathcal{G}$$

$$\boxed{\Pr[A_1 \cdots A_\ell] = \Pr[A_1 \cdots A_{\ell-1}] \cdot \Pr[A_\ell \mid A_1 \cdots A_{\ell-1}]}$$



Width: 2
Length: ℓ
Alphabet: $\mathcal{G}$

One bin is Sensitive to Multiplicative Error

$h(x) = \sigma'_{\varphi(x)}(x)$

$$\Pr_{h\sim\mathcal{H}}[h(y) = \theta \wedge \min h(X\backslash y) > \theta]$$

$$\parallel$$

$\varphi : [N] \to [\ell]$

$$P_1[\mathrm{NZ}] \cdots P_j[\mathrm{NZ}] \cdots P_\ell[\mathrm{NZ}]$$

$$\wr\wr$$

$\sigma'_1, \ldots, \sigma'_\ell : [N] \to [M]$ from NZPRG, $\mathrm{ExtErr} = N^{-o(1)}$

$P_i[\mathcal{G}] \approx P_i[U]$

$$P_i[\mathrm{NZ}] = \Pr_{\sigma'_i:\mathrm{NZPRG}}[A_i]$$

$$P_i[\mathcal{G}] := \Pr_{\sigma_i\sim\mathcal{G}}[A_i]$$

$$P_i[U] := \Pr_{h\sim U}[A_i]$$

? We hope that this special bin does not need to pay for the error from extractor.

$$(P_1[U] \pm \mathrm{ExtErr}) \cdots (P_j[U] \cancel{\pm \mathrm{ExtErr}}) \cdots (P_\ell[U] \pm \mathrm{ExtErr})$$

$P_j[U] = \Pr_{h\sim U}[h(y) = \theta \wedge \min h(X_j\backslash y) > \theta] \leq 1/M \implies P_j[U] \pm \mathrm{ExtErr} = P_j[U] \cdot (1 \pm M \cdot \mathrm{ExtErr})$

! $P_j[U]$ is very small, and it makes this bin very sensitive to multiplicative error.

Change the Order of Inputs

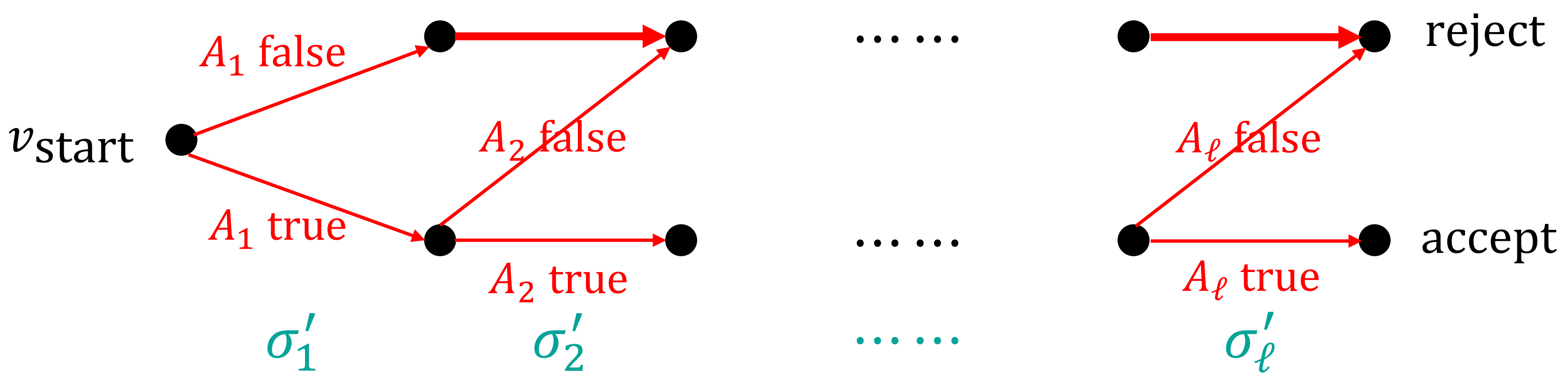
$$\underbrace{(\min \sigma_1'(X_1) > \theta)}_{\text{Event } A_1} \wedge \cdots \wedge \underbrace{\left(\sigma_j'(y) = \theta \wedge \min \sigma_j'\left(X_j \backslash y\right) > \theta\right)}_{\text{Event } A_j} \wedge \cdots \wedge \underbrace{(\min \sigma_\ell'(X_\ell) > \theta)}_{\text{Event } A_\ell}$$

$\mathrm{Ext}\colon \{0,1\}^p \times \{0,1\}^d \to \mathcal{G}$

$\sigma_1' = \mathrm{Ext}(w, s_1), \ldots, \sigma_\ell' = \mathrm{Ext}(w, s_\ell)$

$w \sim U_p, s_1, \ldots, s_\ell$ i.i.d. from U_d

★ $s_1, \ldots s_\ell$ (or correspondingly $\sigma_1', \ldots, \sigma_\ell'$) are symmetric.

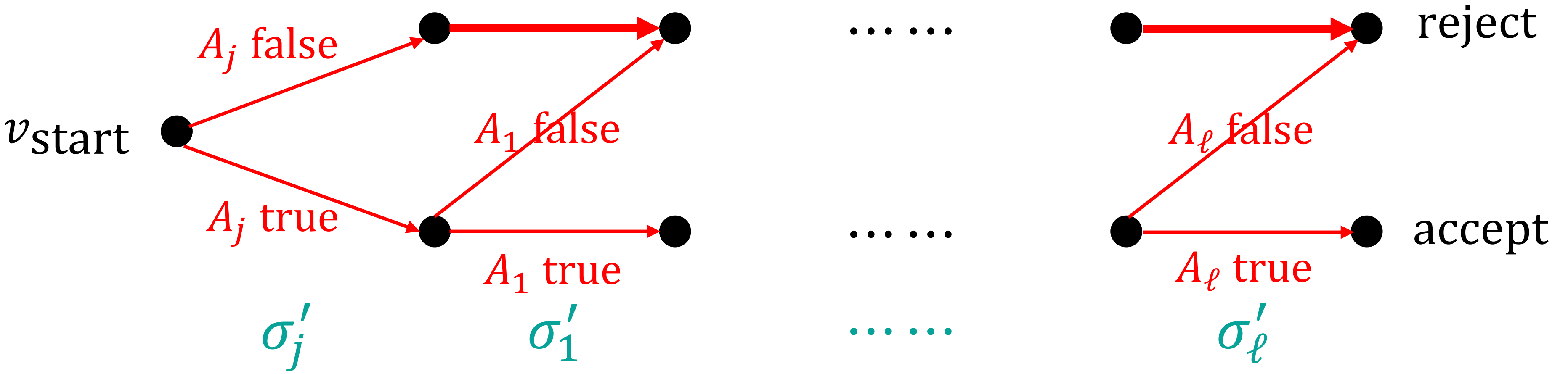


Width: 2
Length: ℓ
Alphabet: $\mathcal{G}$

Change the Order of Inputs

$$\underbrace{\left(\sigma_j'(y) = \theta \wedge \min \sigma_j'\left(X_j \backslash y\right) > \theta\right)}_{\text{Event } A_j} \wedge \underbrace{\left(\min \sigma_1'(X_1) > \theta\right)}_{\text{Event } A_1} \wedge \cdots \wedge \underbrace{\left(\min \sigma_\ell'(X_\ell) > \theta\right)}_{\text{Event } A_\ell}$$

$$\boxed{\begin{gathered} \text{Ext}: \{0,1\}^p \times \{0,1\}^d \to \mathcal{G} \\ \sigma_j' = \text{Ext}(w, s_j), \sigma_1' = \text{Ext}(w, s_1), \ldots, \sigma_\ell' = \text{Ext}(w, s_\ell) \\ w \sim U_p, s_j, s_1, \ldots, s_\ell \text{ i.i.d. from } U_d \end{gathered}}$$



Width: 2
Length: ℓ
Alphabet: $\mathcal{G}$

Change the Order of Inputs

$$P_j[\mathrm{NZ}] \cdot P_1[\mathrm{NZ}] \cdots P_\ell[\mathrm{NZ}]$$

$$\wr\wr$$

$$(P_j[U] \pm \mathrm{ExtErr}) \cdot (P_1[U] \pm \mathrm{ExtErr}) \cdots (P_\ell[U] \pm \mathrm{ExtErr})$$

$\mathrm{Ext}\colon \{0,1\}^p \times \{0,1\}^d \to \mathcal{G}$

$\sigma_j' = \mathrm{Ext}(w, s_j), \sigma_1' = \mathrm{Ext}(w, s_1), \dots, \sigma_\ell' = \mathrm{Ext}(w, s_\ell)$

$w \sim U_p, s_j, s_1, \dots, s_\ell$ i.i.d. from U_d

$H_\infty(w = U_p) = p \geq k$

$\Rightarrow \mathrm{Ext}(w, s_j) \approx_{\mathrm{ExtErr}} \mathcal{G}$

★ w has no entropy loss at the beginning.

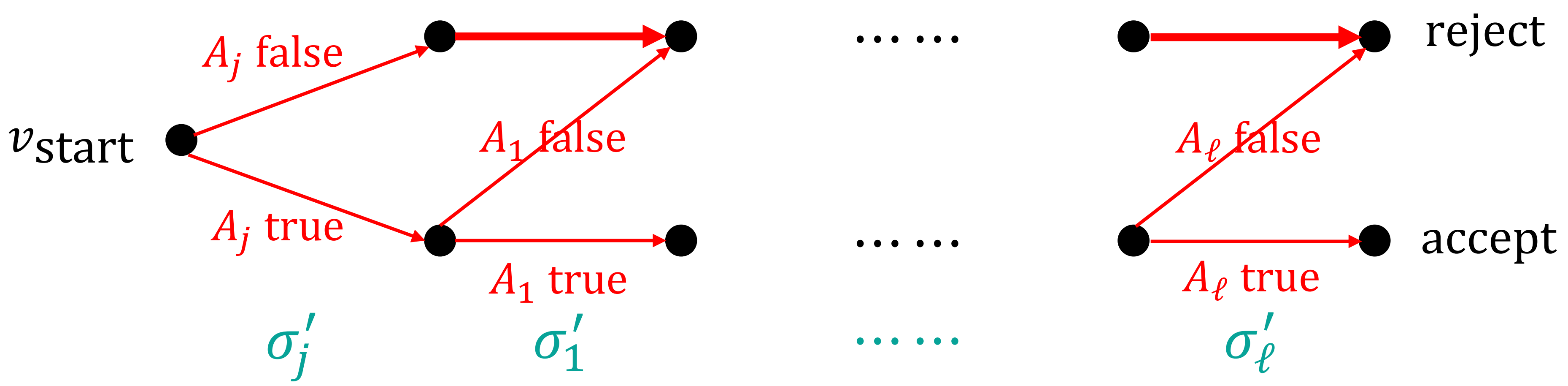
? We expect strong properties for $\mathrm{Ext}(w, s_j)$ than other $\mathrm{Ext}(w, s_i)$.

$P_i[\mathcal{G}] \approx P_i[U]$

$$P_i[\mathrm{NZ}] \coloneqq \Pr_{\sigma_i' : \mathrm{NZPRG}}[A_i]$$

$$P_i[\mathcal{G}] = \Pr_{\sigma_i \sim \mathcal{G}}[A_i]$$

$$P_i[U] \coloneqq \Pr_{h \sim U}[A_i]$$



Width: 2

Length: ℓ

Alphabet: $\mathcal{G}$

Change the Order of Inputs

$$P_j[\mathrm{NZ}] \cdot P_1[\mathrm{NZ}] \cdots P_\ell[\mathrm{NZ}]$$

$$\approx$$

$$(P_j[U] \cancel{\pm \mathrm{ExtErr}}) \cdot (P_1[U] \pm \mathrm{ExtErr}) \cdots (P_\ell[U] \pm \mathrm{ExtErr})$$

$\mathrm{Ext}: \{0,1\}^p \times \{0,1\}^d \to \mathcal{G}$

$\sigma_j' = \mathrm{Ext}(w, s_j), \sigma_1' = \mathrm{Ext}(w, s_1), \dots, \sigma_\ell' = \mathrm{Ext}(w, s_\ell)$

$w \sim U_p, s_j, s_1, \dots, s_\ell$ i.i.d. from U_d

$H_\infty(w = U_p) = p \geq k$

$\Rightarrow$ ~~$\mathrm{Ext}(w, s_j) \approx_{\mathrm{ExtErr}} \mathcal{G}$~~

$\mathrm{Ext}(w, s_j) = \mathcal{G}$

★ w has no entropy loss at the beginning.

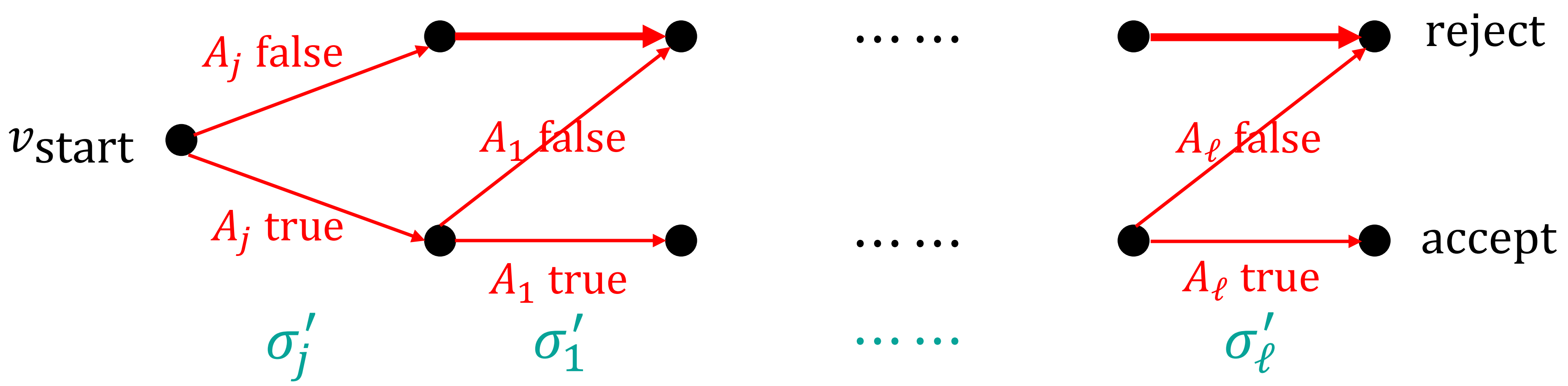
? We expect strong properties for $\mathrm{Ext}(w, s_j)$ than other $\mathrm{Ext}(w, s_i)$.

$P_i[\mathcal{G}] \approx P_i[U]$

$P_i[\mathrm{NZ}] := \Pr_{\sigma_i' : \mathrm{NZPRG}} [A_i]$

$P_i[\mathcal{G}] = \Pr_{\sigma_i \sim \mathcal{G}} [A_i]$

$P_i[U] := \Pr_{h \sim U} [A_i]$



Width: 2

Length: ℓ

Alphabet: $\mathcal{G}$

Special Extractor

Lemma

- Given any p and $k < p$, for any error ε, there exists an explicit (k, ε)-extractor $\mathrm{Ext}\colon \{0,1\}^p \times \{0,1\}^d \to \{0,1\}^q$ with $q = k/2$ and $d = O(\log(p/\varepsilon))$. And Ext satisfies an extra property: $\boxed{\mathrm{Ext}(U_p, s) = U_q}$ for any fixed seed s.

$$\sigma_j' = \mathrm{Ext}\left(w, s_j\right) = \mathrm{Ext}(U_p, U_d) \sim \mathcal{G} \Longrightarrow X_j \text{ has no error from extractor}$$

$$P_j[\mathrm{NZ}] \cdot P_1[\mathrm{NZ}] \cdots P_\ell[\mathrm{NZ}]$$

$$\approx$$

$$\boldsymbol{P_j[U]} \cdot (P_1[U] \pm \mathrm{ExtErr}) \cdots (P_\ell[U] \pm \mathrm{ExtErr})$$

Domain Reduction

$$\mathrm{NZPRG}(w, s_1, \ldots, s_\ell) = \left(\mathrm{Ext}(w, s_1), \ldots, \mathrm{Ext}(w, s_\ell)\right)$$

- Use a PRG for $\left(\{0,1\}^d\right)^\ell$-combinatorial rectangles to generate $s_1, \ldots, s_\ell$, instead of i.i.d. sampling.
- Also need the special property of our extractor.
- Help us to further reduce the error from $1/\log^{O(1)} N$ to $2^{-O\left(\frac{\log N}{\log\log N}\right)}$.

Summary

Main Theorem 1 (Optimal Size && Sub-constant Error, Min-wise Hash)

- There exists an explicit min-wise hash family with seed length $O(\log N)$ and error $2^{-O\left(\frac{\log N}{\log\log N}\right)}$.

Main Theorem 2 (Optimal Size && Sub-constant Error, k-min-wise Hash)

- There exists an explicit k-min-wise hash family with seed length $O(k\log N)$ and error $2^{-O\left(\frac{\log N}{\log\log N}\right)}$, for any $k = \log^{O(1)} N$.

Main Technical Lemma (Extractor with Extra Properties)

- Given any p and $k < p$, for any error ε, there exists an explicit (k, ε)-extractor $\mathrm{Ext}: \{0,1\}^p \times \{0,1\}^d \to \{0,1\}^q$ with $q = k/2$ and $d = O(\log(p/\varepsilon))$. Moreover, Ext satisfies an extra property: $\mathrm{Ext}(U_p, s) = U_q$ for any fixed seed s.

Open Problems

Smaller error with optimal seed length?

Extending the result for larger k (like $\sqrt{N}$) on k-min-wise hash?

Faster evaluation time with optimal seed length?

Lower bound on t-wise independence for k-min-wise hash?

More applications of the powerful Nisan-Zuckerman framework?

Thank you for listening!