Recent progress towards **BPL** vs. **L**

Gil Cohen (Tel Aviv University)

Computational Complexity of Discrete Problems, Dagstuhl

March 15, 2023

The **BPL** vs. **L** Problem

The Problem

Derandomize with low space overhead.

The **BPL** vs. **L** Problem

The Problem

Derandomize with low space overhead.

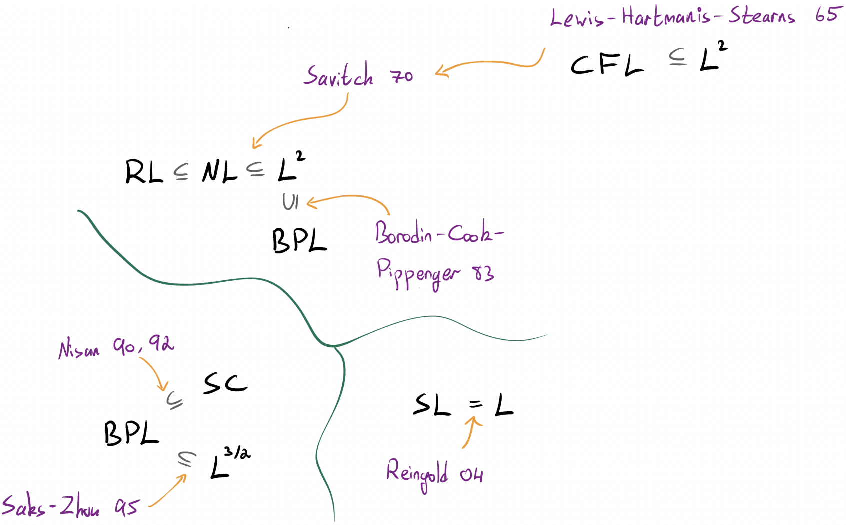
Space s randomized algorithm

$\downarrow$

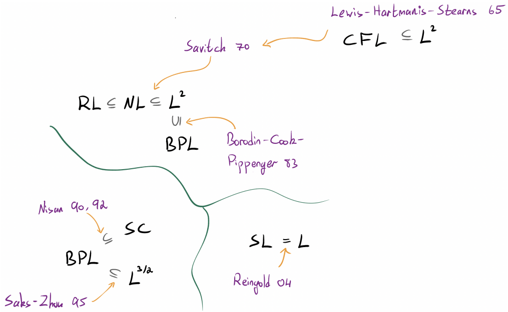
Space s' deterministic algorithm

Hopefully, $s' = O(s)$.

Where are we now?



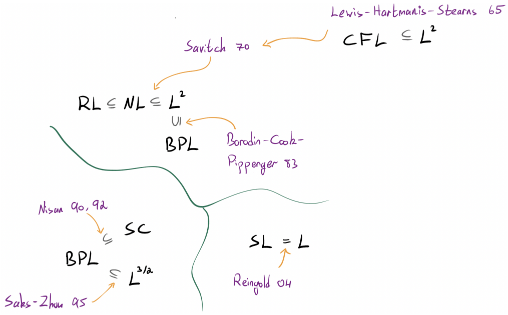
Where are we now?



Several other milestones:

- Nisan-Zuckerman (STOC'93)
- Impagliazzo-Nisan-Wigderson (STOC'94)

Where are we now?

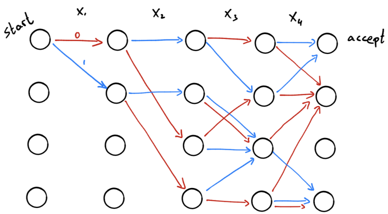


Several other milestones:

- Nisan-Zuckerman (STOC'93)
- Impagliazzo-Nisan-Wigderson (STOC'94)

Exciting advances in recent years (see Hoza's survey'22, STOC'20 workshop).

PRGs for ROBPs

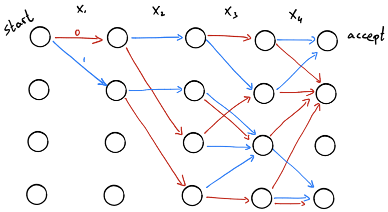


Fact

$\forall n, w, \varepsilon$ $\exists$PRG for (w, n)-ROBPs with seed length

$$s_{\text{opt}} = O\left(\log n + \log w + \log \varepsilon^{-1}\right).$$

PRGs for ROBPs



Fact

$\forall n, w, \varepsilon$ $\exists$*PRG for* (w, n)*-ROBPs with seed length*

$$s_{\text{opt}} = O\left(\log n + \log w + \log \varepsilon^{-1}\right).$$

Theorem (Nisan STOC'90)

$\forall n, w, \varepsilon$ $\exists$*space-efficient PRG for* (w, n)*-ROBPs with seed length*

$$s_{\text{Nisan}} = O\left(\log n \cdot (\log n + \log w + \log \varepsilon^{-1})\right).$$

Nisan's PRG and derandomization

Theorem (Nisan STOC'90)

$\forall n, w, \varepsilon$ $\exists$space-efficient PRG for (w,n)-ROBPs with seed length

$$s_{\text{Nisan}} = O\left(\log n \cdot \left(\log n + \log w + \log \varepsilon^{-1}\right)\right).$$

Naïve derandomization:

$$w = n^{\Theta(1)} \qquad \varepsilon = O(1),$$

and so $s_{\text{Nisan}} = O(\log^2 n)$, hence $\mathbf{BPL} \subseteq \mathbf{L}^2$.

Nisan's PRG and derandomization

Theorem (Nisan STOC'90)

$\forall n, w, \varepsilon$ $\exists$space-efficient PRG for (w, n)-ROBPs with seed length

$$s_{\text{Nisan}} = O\left(\log n \cdot \left(\log n + \log w + \log \varepsilon^{-1}\right)\right).$$

Naïve derandomization:

$$w = n^{\Theta(1)} \qquad \varepsilon = O(1),$$

and so $s_{\text{Nisan}} = O(\log^2 n)$, hence $\mathbf{BPL} \subseteq \mathbf{L}^2$.

Saks-Zhou applies Nisan's PRG in a sophisticated way in the regime

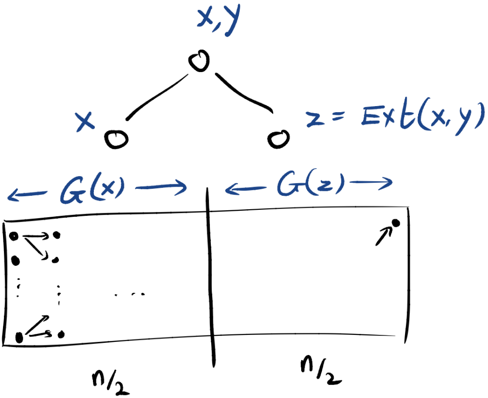
$$w, \varepsilon^{-1} = 2^{\log^2 n} \gg n$$

to conclude $\mathbf{BPL} \subseteq \mathbf{L}^{3/2}$.

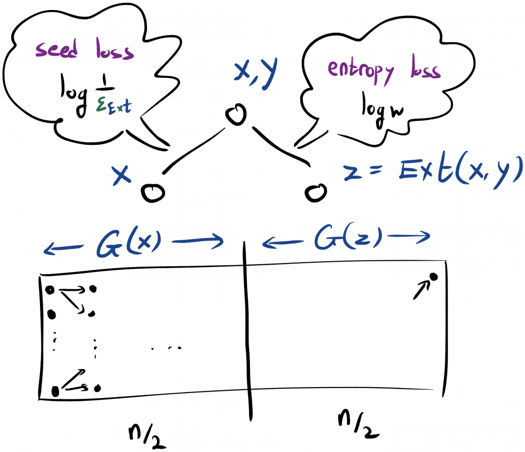
Outline

1. The BPL vs. L Problem
2. Nisan's paradigm
3. The error parameter
4. Improving Saks-Zhou for medium width
5. Summary
6. The width parameter, time permitting

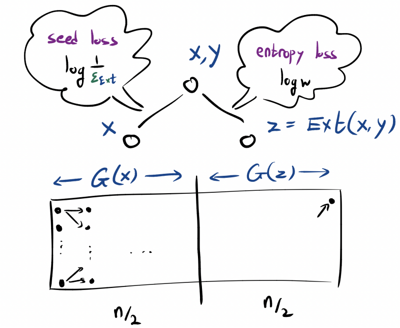
Nisan's paradigm



Nisan's paradigm

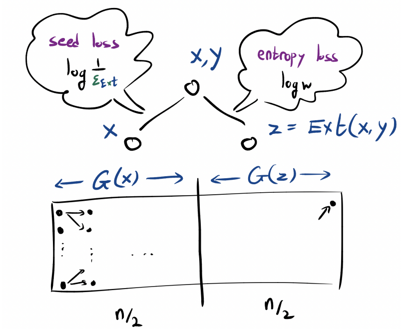


Nisan's paradigm



$$s(n) = \log n \cdot \left(\log w + \log \varepsilon_{\mathrm{Ext}}^{-1}\right)$$

Nisan's paradigm

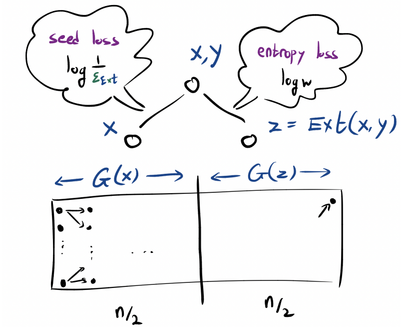


$$s(n) = \log n \cdot \left(\log w + \log \varepsilon_{\mathrm{Ext}}^{-1}\right)$$

Error evolves as

$$\varepsilon(n) = 2\varepsilon(n/2) + \varepsilon_{\mathrm{Ext}} \qquad \Longrightarrow \qquad \varepsilon_{\mathrm{final}} = \varepsilon(n) = n \cdot \varepsilon_{\mathrm{Ext}}$$

Nisan's paradigm



$$s(n) = \log n \cdot \left(\log w + \log \varepsilon_{\text{Ext}}^{-1}\right)$$

Error evolves as

$$\varepsilon(n) = 2\varepsilon(n/2) + \varepsilon_{\text{Ext}} \qquad \Longrightarrow \qquad \varepsilon_{\text{final}} = \varepsilon(n) = n \cdot \varepsilon_{\text{Ext}}$$

Hence,

$$s(n) = O\left(\log n \cdot \left(\log w + \log n + \log \varepsilon_{\text{final}}^{-1}\right)\right)$$

Outline

A tale of three parameters

Observation 1. A space-efficient PRG with seed length

$$s = O\left(\log n \cdot \log n + \log w + \log \varepsilon^{-1}\right),$$

when used in the Saks-Zhou framework, would yield $\mathbf{BPL} \subseteq \mathbf{L}^{4/3}$.

A tale of three parameters

Observation 1. A space-efficient PRG with seed length

$$s = O\left(\log n \cdot \log n + \log w + \log \varepsilon^{-1}\right),$$

when used in the Saks-Zhou framework, would yield $\mathbf{BPL} \subseteq \mathbf{L}^{4/3}$.

Raz-Reingold (STOC'99) suggested a beautiful idea towards obtaining seed length

$$s_{\mathrm{RR}} = O\left(\log n \cdot (\log n + \log \varepsilon^{-1}) + \log w\right).$$

A tale of three parameters

Observation 1. A space-efficient PRG with seed length

$$s = O\left(\log n \cdot \log n + \log w + \log \varepsilon^{-1}\right),$$

when used in the Saks-Zhou framework, would yield $\mathbf{BPL} \subseteq \mathbf{L}^{4/3}$.

Raz-Reingold (STOC'99) suggested a beautiful idea towards obtaining seed length

$$s_{\mathrm{RR}} = O\left(\log n \cdot \left(\log n + \log \varepsilon^{-1}\right) + \log w\right).$$

Observation 2. The $\log n \cdot \log n$ term is due to the way that the error evolves in Nisan's paradigm. Thus, better control on the way the error evolves may solve both problems, giving

$$s_{\mathrm{dreamy}} = O\left(\log n \cdot \log w + \log \varepsilon^{-1}\right).$$

A tale of three parameters

With Braverman and Garg (STOC'18) we obtained, essentially, a PRG with seed length

$$s_{\mathrm{BCG}} = \widetilde{O}\left(\log n \cdot (\log n + \log w) + \log \varepsilon^{-1}\right).$$

More precisely, we introduced and constructed weighted PRGs.

Weighted PRGs

Definition

A weighted PRG with error ε against a class of functions $\mathcal{C}$ is a function

$$(\mathrm{G}, \omega) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$$

s.t. $\forall f \in \mathcal{C}$,

$$\left| \mathbb{E}[f(U_n)] - \sum_{\sigma \in \{0,1\}^s} \omega(\sigma) f(\mathrm{G}(\sigma)) \right| \le \varepsilon.$$

Weighted PRGs

Definition

A weighted PRG with error ε against a class of functions $\mathcal{C}$ is a function

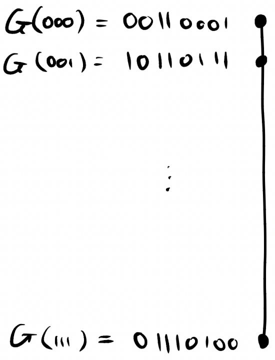
$$(\mathrm{G}, \omega) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$$

s.t. $\forall f \in \mathcal{C}$,

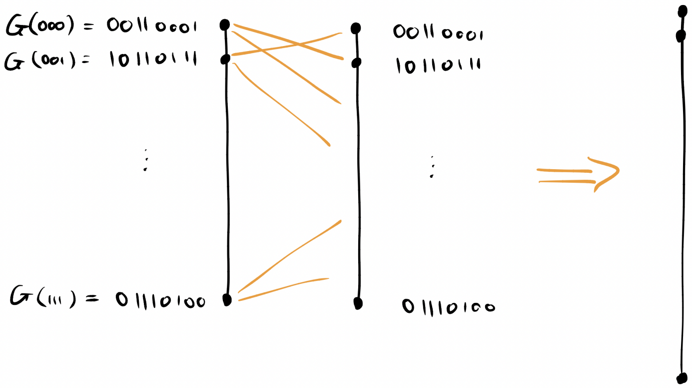
$$\left| \mathbb{E}[f(U_n)] - \sum_{\sigma \in \{0,1\}^s} \omega(\sigma) f(\mathrm{G}(\sigma)) \right| \leq \varepsilon.$$

- WPRGs are as good as PRGs for naïve derandomization and also for the Saks-Zhou framework.
- WPRGs induce hitting sets.
- Hoza and Zuckerman (FOCS'18) gave a much simplified hitting set with such parameters.

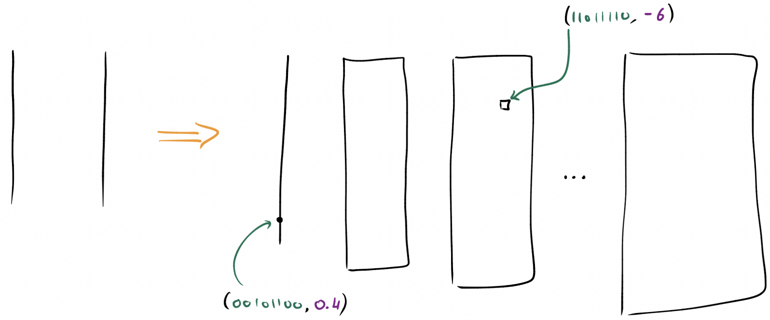
The idea underlying BCG



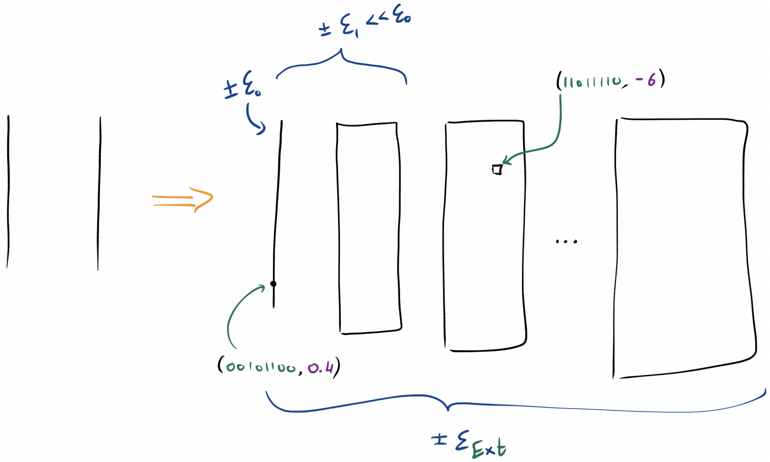
The idea underlying BCG



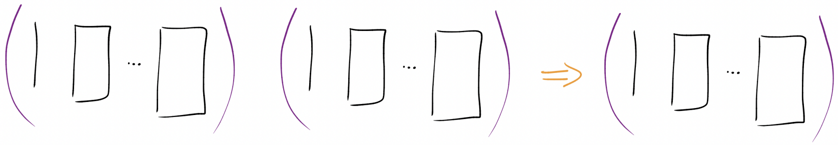
The idea underlying BCG



The idea underlying BCG



The idea underlying BCG



Error reduction via Richardson iterations

Several simplifications to BCG were introduced, most notably, Chattopadhyay-Liao (CCC'20).

With Doron, Renard, Sberlo, and Ta-Shma (CCC'21), we obtained a substantial simplification, in fact, an error reduction procedure

$$n^{-1}\text{-error PRG} \quad \rightarrow \quad \varepsilon\text{-error weighted PRG}$$

with essentially optimal seed length overhead of $\approx \log \varepsilon^{-1}$.

Error reduction via Richardson iterations

Several simplifications to BCG were introduced, most notably, Chattopadhyay-Liao (CCC'20).

With Doron, Renard, Sberlo, and Ta-Shma (CCC'21), we obtained a substantial simplification, in fact, an error reduction procedure

$$n^{-1}\text{-error PRG} \quad \rightarrow \quad \varepsilon\text{-error weighted PRG}$$

with essentially optimal seed length overhead of $\approx \log \varepsilon^{-1}$.

The result was concurrently and independently obtained by Pyne and Vadhan (CCC'21). Hoza (RANDOM'21) got rid of all $\log\log$ factors.

Error reduction via Richardson iterations

Let $\mathbf{A}$ be the random walk operator corresponding to a ROBP. We wish to approximate $\mathbf{A}^n$. Note that

$$(\mathbf{I}-\mathbf{A})^{-1} = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^n + \cdots$$

To avoid this "interference" of all powers we can consider the tensor with the directed path graph. E.g.,

$$\mathbf{P}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \mathbf{P}_4 \otimes \mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ 0 & \mathbf{A} & 0 & 0 \\ 0 & 0 & \mathbf{A} & 0 \end{pmatrix}$$

Error reduction via Richardson iterations

$$(\mathbf{I} - \mathbf{P}_4 \otimes \mathbf{A})^{-1} = \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ \mathbf{A} & \mathbf{I} & 0 & 0 \\ \mathbf{A}^2 & \mathbf{A} & \mathbf{I} & 0 \\ \mathbf{A}^3 & \mathbf{A}^2 & \mathbf{A} & \mathbf{I} \end{pmatrix}.$$

$\mathbf{L} = \mathbf{I} - \mathbf{P}_{n+1} \otimes \mathbf{A}$ is the Laplacian of the directed graph $\mathbf{P}_{n+1} \otimes \mathbf{A}$.

Error reduction via Richardson iterations

$$(\mathbf{I} - \mathbf{P}_4 \otimes \mathbf{A})^{-1} = \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ \mathbf{A} & \mathbf{I} & 0 & 0 \\ \mathbf{A}^2 & \mathbf{A} & \mathbf{I} & 0 \\ \mathbf{A}^3 & \mathbf{A}^2 & \mathbf{A} & \mathbf{I} \end{pmatrix}.$$

$\mathbf{L} = \mathbf{I} - \mathbf{P}_{n+1} \otimes \mathbf{A}$ is the Laplacian of the directed graph $\mathbf{P}_{n+1} \otimes \mathbf{A}$.

Define

$$\mathbf{L}_k = \sum_{i=0}^{k} (\mathbf{I} - \widetilde{\mathbf{L}^{-1}}\mathbf{L})^i \widetilde{\mathbf{L}^{-1}}.$$

It is easy to verify that

$$\|\mathbf{I} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}\| \leq \varepsilon_0 \quad \Longrightarrow \quad \|\mathbf{I} - \mathbf{L}_k\mathbf{L}\| \leq \varepsilon_0^{k+1},$$

Error reduction via Richardson iterations

Thus, to obtain a good ε approximation of $\mathbf{A}^n$, we

1. Compute a modest ε_0 approximation $\widetilde{\mathbf{A}^i}$ of $\mathbf{A}^i$ for $1 \le i \le n$. Namely, $\|\widetilde{\mathbf{A}^i} - \mathbf{A}^i\| \le \varepsilon_0$.
2. Construct

$$\widetilde{\mathbf{L}^{-1}} = \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 & 0 \\ \widetilde{\mathbf{A}} & \mathbf{I} & 0 & 0 & 0 \\ \widetilde{\mathbf{A}^2} & \widetilde{\mathbf{A}} & \mathbf{I} & 0 & 0 \\ \vdots & \vdots & \widetilde{\mathbf{A}} & \mathbf{I} & 0 \\ \widetilde{\mathbf{A}^n} & \widetilde{\mathbf{A}^{n-1}} & \dots & \widetilde{\mathbf{A}} & \mathbf{I} \end{pmatrix}.$$

3. Compute $\mathbf{L}_k = \sum_{i=0}^{k} (\mathbf{I} - \widetilde{\mathbf{L}^{-1}}\mathbf{L})^i \widetilde{\mathbf{L}^{-1}}$ for $k = \frac{\log \varepsilon^{-1}}{\log \varepsilon_0^{-1}}$.
4. Return the bottom-left block of $\mathbf{L}_k$.

Example $k = 1, n = 3$

$$\mathbf{L}_1 = \sum_{i=0}^{k=1} (\mathbf{I} - \widetilde{\mathbf{L}^{-1}}\mathbf{L})^i \widetilde{\mathbf{L}^{-1}}, ,$$

where recall

$$\widetilde{\mathbf{L}^{-1}} = \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ \widetilde{\mathbf{A}} & \mathbf{I} & 0 & 0 \\ \widetilde{\mathbf{A}^2} & \widetilde{\mathbf{A}} & \mathbf{I} & 0 \\ \widetilde{\mathbf{A}^3} & \widetilde{\mathbf{A}^2} & \widetilde{\mathbf{A}} & \mathbf{I} \end{pmatrix} \qquad \mathbf{L} = \mathbf{I} - \mathbf{P}_4 \otimes \mathbf{A} = \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ -\mathbf{A} & \mathbf{I} & 0 & 0 \\ 0 & -\mathbf{A} & \mathbf{I} & 0 \\ 0 & 0 & -\mathbf{A} & \mathbf{I} \end{pmatrix}$$

Then,

$$\mathbf{L}_1 = \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ \mathbf{A} & \mathbf{I} & 0 & 0 \\ \widetilde{\mathbf{A}}\mathbf{A} + \mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^2} & \mathbf{A} & \mathbf{I} & 0 \\ \widetilde{\mathbf{A}^2}\mathbf{A} - \widetilde{\mathbf{A}^2}\widetilde{\mathbf{A}} + \widetilde{\mathbf{A}}\mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{A}^2} + \mathbf{A}\widetilde{\mathbf{A}^2} & \widetilde{\mathbf{A}}\mathbf{A} + \mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^2} & \mathbf{A} & \mathbf{I} \end{pmatrix}.$$

Example $k = 1, n = 4$

$$
\begin{pmatrix}
\mathbf{I} & 0 & 0 & 0 \\
\mathbf{A} & \mathbf{I} & 0 & 0 \\
\widetilde{\mathbf{A}}\mathbf{A} + \mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^2} & \mathbf{A} & \mathbf{I} & 0 \\
\widetilde{\mathbf{A}^2}\mathbf{A} - \widetilde{\mathbf{A}^2}\widetilde{\mathbf{A}} + \widetilde{\mathbf{A}}\mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{A}^2} + \mathbf{A}\widetilde{\mathbf{A}^2} & \widetilde{\mathbf{A}}\mathbf{A} + \mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^2} & \mathbf{A} & \mathbf{I}
\end{pmatrix}.
$$

Example $k = 1, n = 4$

$$\begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ \mathbf{A} & \mathbf{I} & 0 & 0 \\ \widetilde{\mathbf{A}}\mathbf{A} + \mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^2} & \mathbf{A} & \mathbf{I} & 0 \\ \widetilde{\mathbf{A}^2}\mathbf{A} - \widetilde{\mathbf{A}^2}\widetilde{\mathbf{A}} + \widetilde{\mathbf{A}}\mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{A}^2} + \mathbf{A}\widetilde{\mathbf{A}^2} & \widetilde{\mathbf{A}}\mathbf{A} + \mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^2} & \mathbf{A} & \mathbf{I} \end{pmatrix} \cdot$$

$$\begin{pmatrix} \mathbf{I} & 0 & 0 & 0 & 0 \\ \mathbf{A} & \mathbf{I} & 0 & 0 & 0 \\ \widetilde{\mathbf{A}}\mathbf{A} + \mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^2} & \mathbf{A} & \mathbf{I} & 0 & 0 \\ \widetilde{\mathbf{A}^2}\mathbf{A} - \widetilde{\mathbf{A}^2}\widetilde{\mathbf{A}} + \widetilde{\mathbf{A}}\mathbf{A}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{A}^2} + \mathbf{A}\widetilde{\mathbf{A}^2} & \nwarrow & \mathbf{A} & \mathbf{I} & 0 \\ \widetilde{\mathbf{A}^3}\mathbf{A} + \widetilde{\mathbf{A}}\widetilde{\mathbf{A}^2}\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}^3}\widetilde{\mathbf{A}} + \widetilde{\mathbf{A}}\mathbf{A}\widetilde{\mathbf{A}^2} - \widetilde{\mathbf{A}^2}\widetilde{\mathbf{A}^2} + \mathbf{A}\widetilde{\mathbf{A}^3} - \widetilde{\mathbf{A}}\widetilde{\mathbf{A}^3} & \nwarrow & \nwarrow & \mathbf{A} & \mathbf{I} \end{pmatrix}$$

Outline

Improving Saks-Zhou for medium width

Ignoring ε, in matrix-language, Saks-Zhou give a space

$$O\left(\sqrt{\log n} \cdot (\log n + \log w)\right)$$

algorithm for approximating $\mathbf{A}^n$ and, more generally, the product of n stochastic $w \times w$ matrices.

Improving Saks-Zhou for medium width

Ignoring ε, in matrix-language, Saks-Zhou give a space

$$O\left(\sqrt{\log n} \cdot (\log n + \log w)\right)$$

algorithm for approximating $\mathbf{A}^n$ and, more generally, the product of n stochastic $w \times w$ matrices.

Joint with Doron, Sberlo and Ta-Shma (STOC'23), we reduce the space down to

$$\widetilde{O}\left(\log n + \sqrt{\log n} \cdot \log w\right).$$

This is nearly optimal for width up to $w = 2^{\sqrt{\log n}}$.

Improving Saks-Zhou for medium width

Ignoring ε, in matrix-language, Saks-Zhou give a space

$$O\left(\sqrt{\log n}\cdot(\log n+\log w)\right)$$

algorithm for approximating $\mathbf{A}^n$ and, more generally, the product of n stochastic $w \times w$ matrices.

Joint with Doron, Sberlo and Ta-Shma (STOC'23), we reduce the space down to

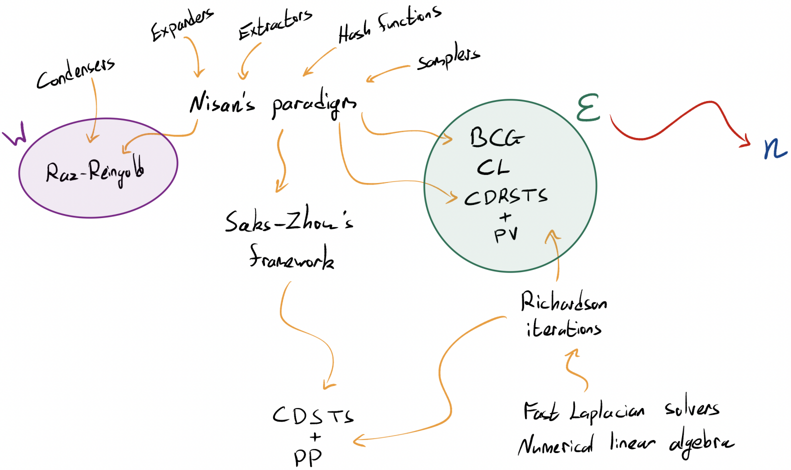
$$\widetilde{O}\left(\log n+\sqrt{\log n}\cdot\log w\right).$$

This is nearly optimal for width up to $w = 2^{\sqrt{\log n}}$.

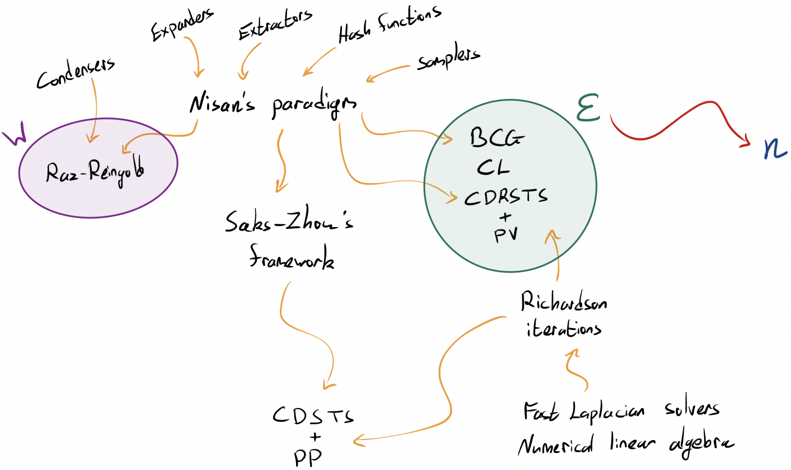
Based on our earlier manuscript on matrix powering, the case of iterated product was concurrently and independently obtained by Putterman and Pyne (STOC'23).

Outline

Summary


Expanders
Extractors
Hash functions
Condensers
samplers
Nisan's paradigm
Raz-Reingold
BCG
CL
CDRSTS
+
PV
Saks-Zhou's
framework
Richardson
iterations
CDSTS
+
PP
Fast Laplacian solvers
Numerical linear algebra

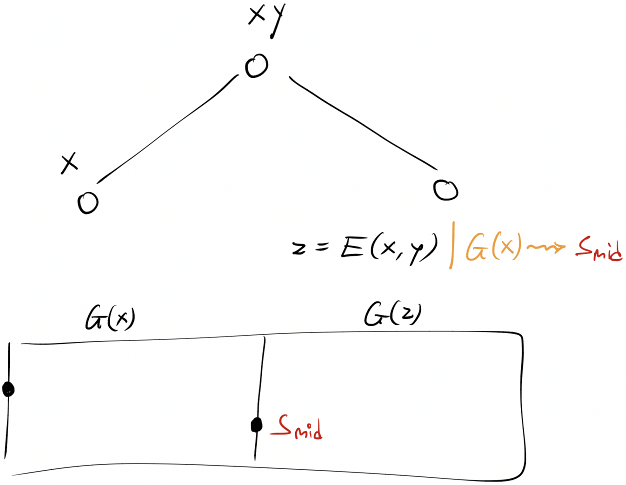
Summary



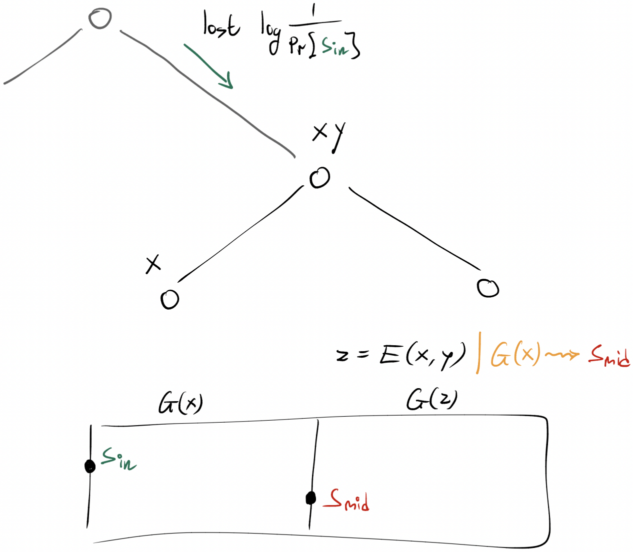
Thank you!

Outline

The width parameter



The width parameter



The width parameter