Explicit Abelian Lifts and Quantum LDPC Codes

Fernando G. Jeronimo, **Tushant Mittal**, Ryan O'Donnell,
Pedro Paredes, Madhur Tulsiani

ITCS 2022

- Main goal is to explicitly build symmetric expanding graphs

- Let us see why and how!

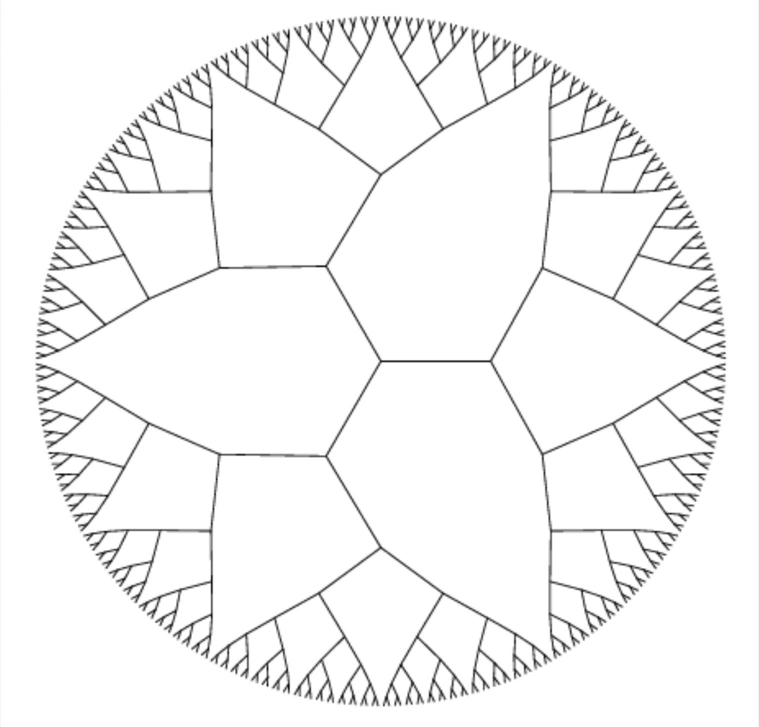


Image credits - Hoory, Linial, Wigderson ‘06

OUTLINE

1. **Introduction**

   Motivation and history

2. **Our Results**

   Statement and an application

3. **Key Contribution**

   A better count via DFS!

4. **Conclusion**

   Summary and open Problems

1.

Introduction

Here we go!

Expansion

Expansion

- A notion that captures how well-connected a graph is.

Expansion

- A notion that captures how well-connected a graph is.

Expansion

- A notion that captures how well-connected a graph is.
- Spectral notion - $\lambda(G) = \max\left(\lambda_2(A_G), |\lambda_n(A_G)| \right)$.

Expansion

- A notion that captures how well-connected a graph is.
- Spectral notion - $\lambda(G) = \max\left(\ \lambda_2(A_G), |\lambda_n(A_G)|\ \right)$.
    - The smaller $\lambda(G)$ is, the better the expander.

Expansion

- A notion that captures how well-connected a graph is.

- Spectral notion - $\lambda(G) = \max\left(\lambda_2(A_G), |\lambda_n(A_G)| \right)$.
    - The smaller $\lambda(G)$ is, the better the expander.
    - The complete graph has $\lambda(K_n) = 1$.

Expansion

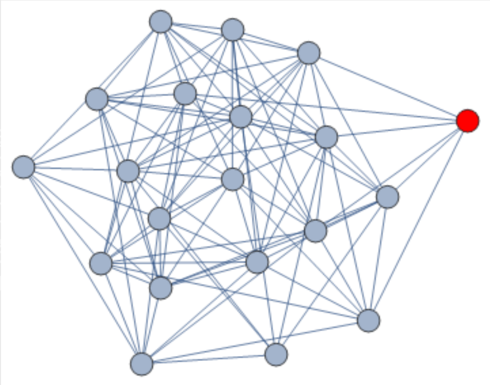
- A notion that captures how well-connected a graph is.

- Spectral notion - $\lambda(G) = \max\left(\lambda_2(A_G), |\lambda_n(A_G)| \right)$.
    - The smaller $\lambda(G)$ is, the better the expander.
    - The complete graph has $\lambda(K_n) = 1$.
    - If G is d-regular and disconnected, $\lambda(G) = d$.

Expansion

- A notion that captures how well-connected a graph is.

- Spectral notion - $\lambda(G) = \max\left(\ \lambda_2(A_G), |\lambda_n(A_G)|\ \right)$.
  - The smaller $\lambda(G)$ is, the better the expander.
  - The complete graph has $\lambda(K_n) = 1$.
  - If G is d-regular and disconnected, $\lambda(G) = d$.

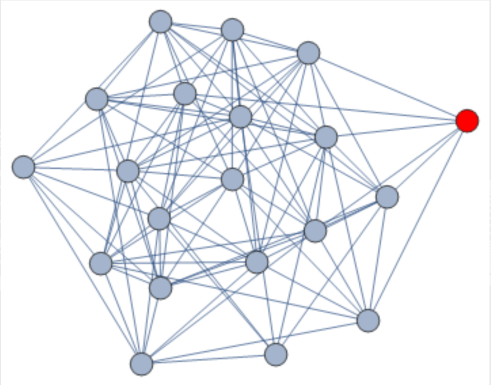


User ‘rhermans’ on Mathematica.SE

Expansion

- A notion that captures how well-connected a graph is.

- Spectral notion - $\lambda(G) = \max\left(\ \lambda_2(A_G), |\lambda_n(A_G)|\ \right)$.
  - The smaller $\lambda(G)$ is, the better the expander.
  - The complete graph has $\lambda(K_n) = 1$.
  - If G is d-regular and disconnected, $\lambda(G) = d$.



User 'rhermans' on Mathematica.SE

- Q - Given d, ε, can we construct infinite families of d-regular graphs $\{G_n\}$ with $n \to \infty$ such that $\lambda(G_n) \leq \varepsilon d$?
  - Alon-Boppanna bound says that the best possible is $2\sqrt{d-1} - o_n(1)$.

Symmetry

Symmetry

- A graph isomorphism is a bijective map $\varphi : V(G) \to V(G)$ such that for (u, v) is an edge iff $(\varphi(u), \varphi(v))$ is.

Symmetry

- A graph isomorphism is a bijective map $\varphi : V(G) \to V(G)$ such that for (u, v) is an edge iff $(\varphi(u), \varphi(v))$ is.

Symmetry

- A graph isomorphism is a bijective map $\varphi : V(G) \to V(G)$ such that for (u, v) is an edge iff $(\varphi(u), \varphi(v))$ is.
- $\mathrm{Aut}(G)$ is the group of all isomorphisms of G.

Symmetry

- A graph isomorphism is a bijective map $\varphi : V(G) \to V(G)$ such that for (u, v) is an edge iff $(\varphi(u), \varphi(v))$ is.

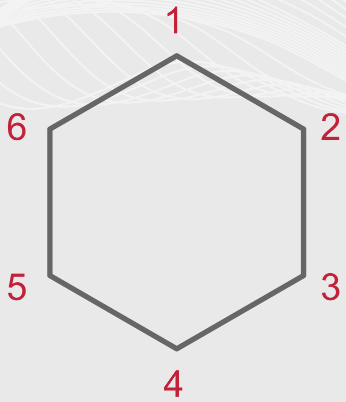
- $\mathrm{Aut}(G)$ is the group of all isomorphisms of G.

Symmetry

- A graph isomorphism is a bijective map $\varphi : V(G) \to V(G)$ such that for (u, v) is an edge iff $(\varphi(u), \varphi(v))$ is.
- $\mathrm{Aut}(G)$ is the group of all isomorphisms of G.
- Q - Given H, can we construct graphs such that $H \subseteq Aut(G)$?

Symmetry

- A graph isomorphism is a bijective map $\varphi : V(G) \to V(G)$ such that for (u, v) is an edge iff $(\varphi(u), \varphi(v))$ is.
- $\mathrm{Aut}(G)$ is the group of all isomorphisms of G.
- Q - Given H, can we construct graphs such that $H \subseteq Aut(G)$?



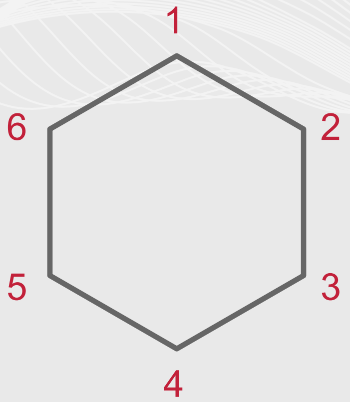
Symmetry

- A graph isomorphism is a bijective map $\varphi : V(G) \to V(G)$ such that for (u, v) is an edge iff $(\varphi(u), \varphi(v))$ is.

- $\mathrm{Aut}(G)$ is the group of all isomorphisms of G.

- Q - Given H, can we construct graphs such that $H \subseteq Aut(G)$?

- Eg - If $H = \mathbb{Z}_6$, we have the cycle graph C_6 such that
  - $\mathbb{Z}_6 \subseteq \mathrm{Aut}(C_6)$ as $i : n \to n + i \mod 6$.



Two nice graph properties

Two nice graph properties

Two nice graph properties

Expansion

Two nice graph properties

Expansion

Two nice graph properties

Expansion

- Many explicit constructions of constant degree expander graphs known.

Two nice graph properties

Expansion

- Many explicit constructions of constant degree expander graphs known.

Two nice graph properties

Expansion

Symmetry

- Many explicit constructions of constant degree expander graphs known.

Two nice graph properties

Expansion

Symmetry

- Many explicit constructions of constant degree expander graphs known.

Two nice graph properties

Expansion

- Many explicit constructions of constant degree expander graphs known.

Symmetry

- [Babai'74] For any finite group H, there exists an explicit graph X with $Aut(X) = H$.

Two nice graph properties

Expansion

- Many explicit constructions of constant degree expander graphs known.

Symmetry

- [Babai'74] For any finite group H, there exists an explicit graph X with $Aut(X) = H$.

Q - Can we have both ?

13 REASONS WHY

2 REASONS WHY

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -
  - Given linear codes - C_1, C_2 each with the symmetry of a group H, one can define a quantum CSS code - $C_1 \otimes_H C_2$.

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -
  - Given linear codes - C_1, C_2 each with the symmetry of a group H, one can define a quantum CSS code - $C_1 \otimes_H C_2$.

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -
  - Given linear codes - C_1, C_2 each with the symmetry of a group H, one can define a quantum CSS code - $C_1 \otimes_H C_2$.
  - $\mathbb{Z}_\ell$ - [Hastings, Haah and O'Donnell '20], [Panteleev, Kalachev '20].

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -
  - Given linear codes - C_1, C_2 each with the symmetry of a group H, one can define a quantum CSS code - $C_1 \otimes_H C_2$.
  - $\mathbb{Z}_\ell$ - [Hastings, Haah and O'Donnell '20], [Panteleev, Kalachev '20].

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -
  - Given linear codes - C_1, C_2 each with the symmetry of a group H, one can define a quantum CSS code - $C_1 \otimes_H C_2$.
  - $\mathbb{Z}_\ell$ - [Hastings, Haah and O'Donnell '20], [Panteleev, Kalachev '20].
  - $\mathrm{PSL}_2(\mathbb{F}_q)$ - [Panteleev, Kalachev'21], [Dinur, Evra, Livne, Lubotzky, Mozes'21].

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -
  - Given linear codes - C_1, C_2 each with the symmetry of a group H, one can define a quantum CSS code - $C_1 \otimes_H C_2$.
  - $\mathbb{Z}_\ell$ - [Hastings, Haah and O'Donnell '20], [Panteleev, Kalachev '20].
  - $\mathrm{PSL}_2(\mathbb{F}_q)$ - [Panteleev, Kalachev'21], [Dinur, Evra, Livne, Lubotzky, Mozes'21].

2 REASONS WHY

- Good Quantum LDPC codes and Locally testable codes -
  - Given linear codes - C_1, C_2 each with the symmetry of a group H, one can define a quantum CSS code - $C_1 \otimes_H C_2$.
  - $\mathbb{Z}_\ell$ - [Hastings, Haah and O'Donnell '20], [Panteleev, Kalachev '20].
  - $\mathrm{PSL}_2(\mathbb{F}_q)$ - [Panteleev, Kalachev'21], [Dinur, Evra, Livne, Lubotzky, Mozes'21].
- Property Testing - Interesting work by [Goldreich-Wigderson'21] builds expander graphs with $\mathrm{Aut}(X) = \{\mathrm{id}\}$ and shows applications to property testing.

Q - For a given family of groups H_n, can we explicitly construct a family of expander graphs G_n such that $H_n \subseteq \mathrm{Aut}(G_n)$?

Known Techniques

Known Techniques

- Algebraic Constructions - Specific constructions for certain groups like $\mathrm{PSL}_2(\mathbb{F}_q)$ but are highly non-elementary.

Known Techniques

- Algebraic Constructions - Specific constructions for certain groups like $\mathrm{PSL}_2(\mathbb{F}_q)$ but are highly non-elementary.

Known Techniques

- Algebraic Constructions - Specific constructions for certain groups like $\mathrm{PSL}_2(\mathbb{F}_q)$ but are highly non-elementary.

- Random Cayley Graphs - [Alon-Roichman'01] - For a random $S \subseteq H$, such that $|S| = \Theta(\log |H|)$, $Cay(H, S)$ is an expanding graph.

Known Techniques

- Algebraic Constructions - Specific constructions for certain groups like $\mathrm{PSL}_2(\mathbb{F}_q)$ but are highly non-elementary.
- Random Cayley Graphs - [Alon-Roichman'01] - For a random $S \subseteq H$, such that $|S| = \Theta(\log |H|)$, $Cay(H, S)$ is an expanding graph.
  - The degree is logarithmic and the bound is tight when H is abelian.

Known Techniques

- Algebraic Constructions - Specific constructions for certain groups like $\mathrm{PSL}_2(\mathbb{F}_q)$ but are highly non-elementary.
- Random Cayley Graphs - [Alon-Roichman'01] - For a random $S \subseteq H$, such that $|S| = \Theta(\log |H|)$, $Cay(H, S)$ is an expanding graph.
  - The degree is logarithmic and the bound is tight when H is abelian.

Known Techniques

- Algebraic Constructions - Specific constructions for certain groups like $\mathrm{PSL}_2(\mathbb{F}_q)$ but are highly non-elementary.

- Random Cayley Graphs - [Alon-Roichman'01] - For a random $S \subseteq H$, such that $|S| = \Theta(\log|H|)$, $Cay(H, S)$ is an expanding graph.
    - The degree is logarithmic and the bound is tight when H is abelian.

- Group-based lifts (Covering maps) - A generic technique introduced by Bilu, Linial'06 in context of graphs. A special case of the topological notion of covering maps.

Known Techniques

- Algebraic Constructions - Specific constructions for certain groups like $\mathrm{PSL}_2(\mathbb{F}_q)$ but are highly non-elementary.

- Random Cayley Graphs - [Alon-Roichman'01] - For a random $S \subseteq H$, such that $|S| = \Theta(\log|H|)$, $Cay(H, S)$ is an expanding graph.
  - The degree is logarithmic and the bound is tight when H is abelian.

- Group-based lifts (Covering maps) - A generic technique introduced by Bilu, Linial'06 in context of graphs. A special case of the topological notion of covering maps.
  - Used extensively to construct expanders.

(H, ℓ) lift of a graph

(H, ℓ) lift of a graph

G

(H, ℓ) lift of a graph

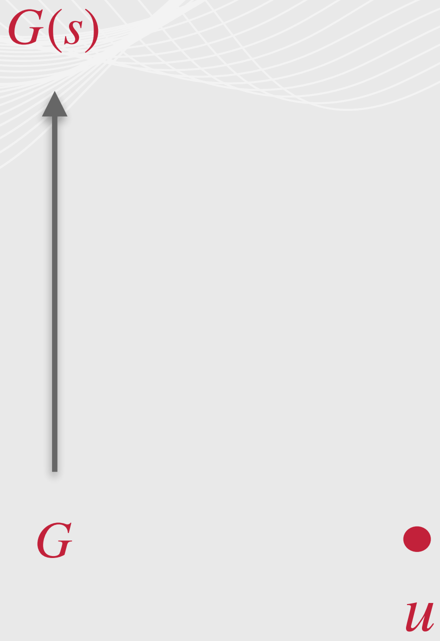
G

(H, ℓ) lift of a graph

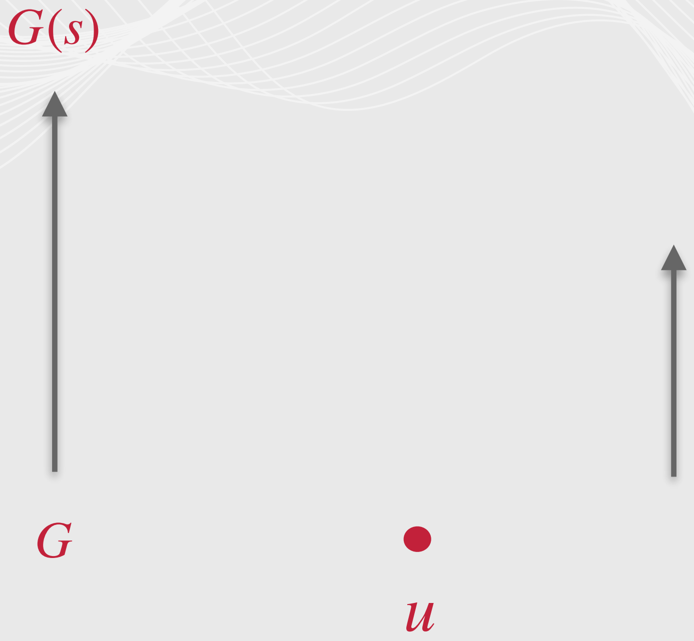
$$\begin{tikzcd} G(s) \\ G \arrow[u] \end{tikzcd}$$

(H, ℓ) lift of a graph

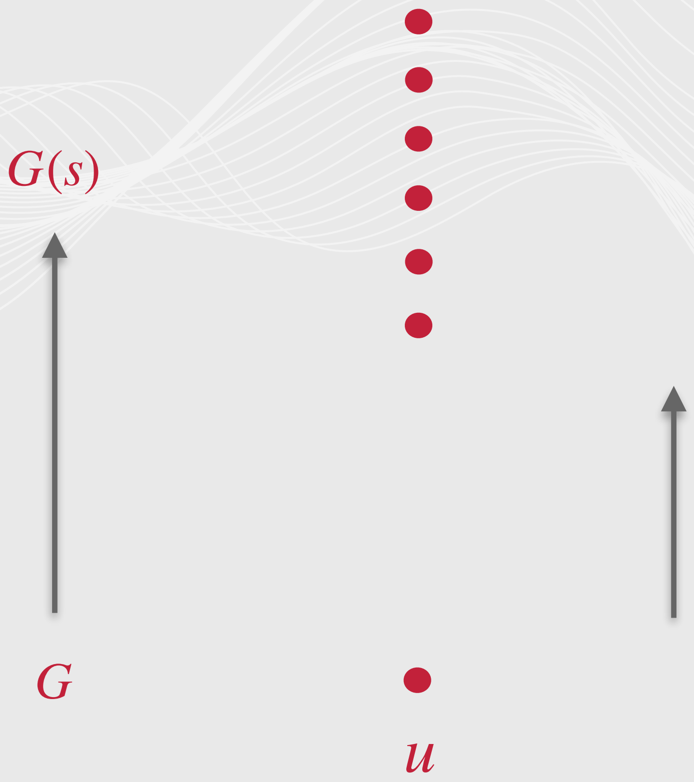
(H, ℓ) lift of a graph



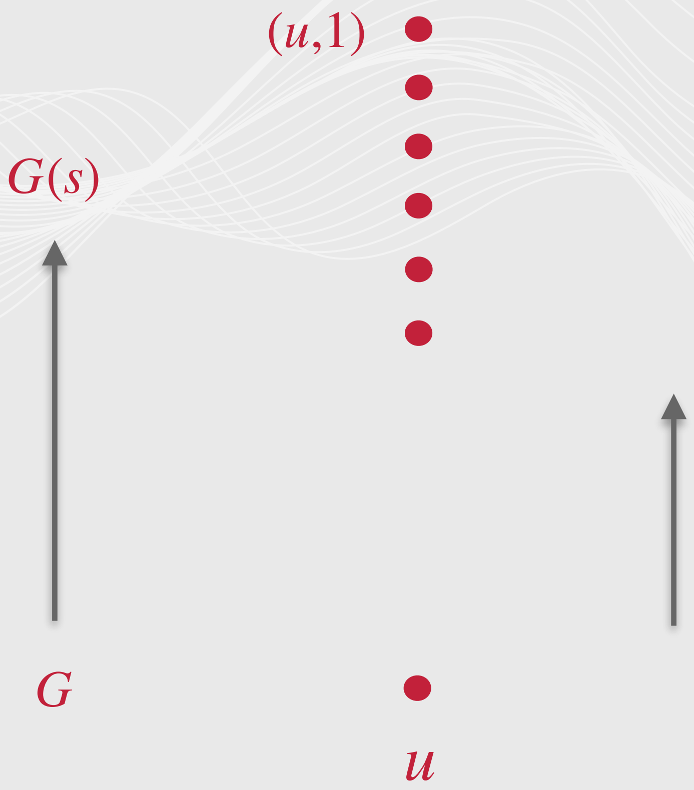
(H, ℓ) lift of a graph



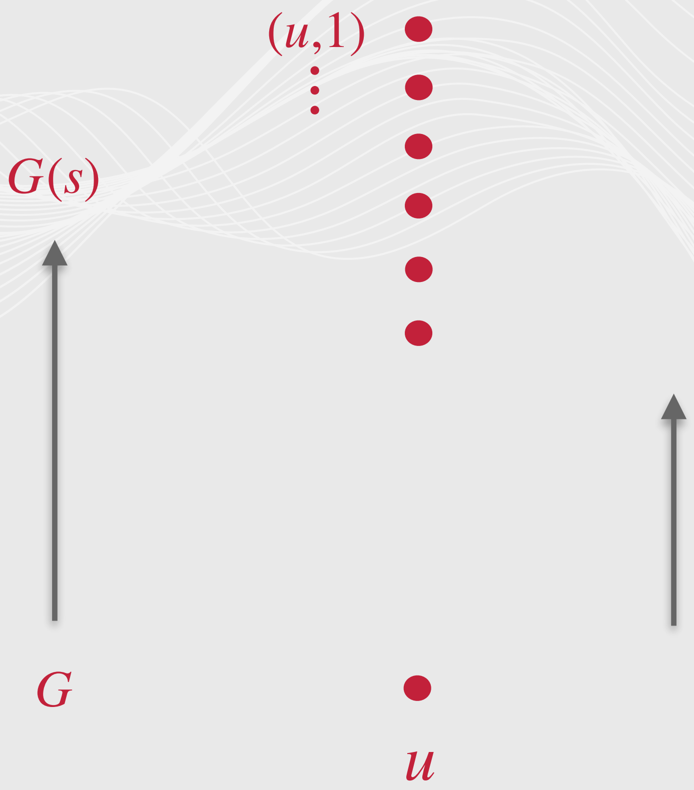
(H, ℓ) lift of a graph



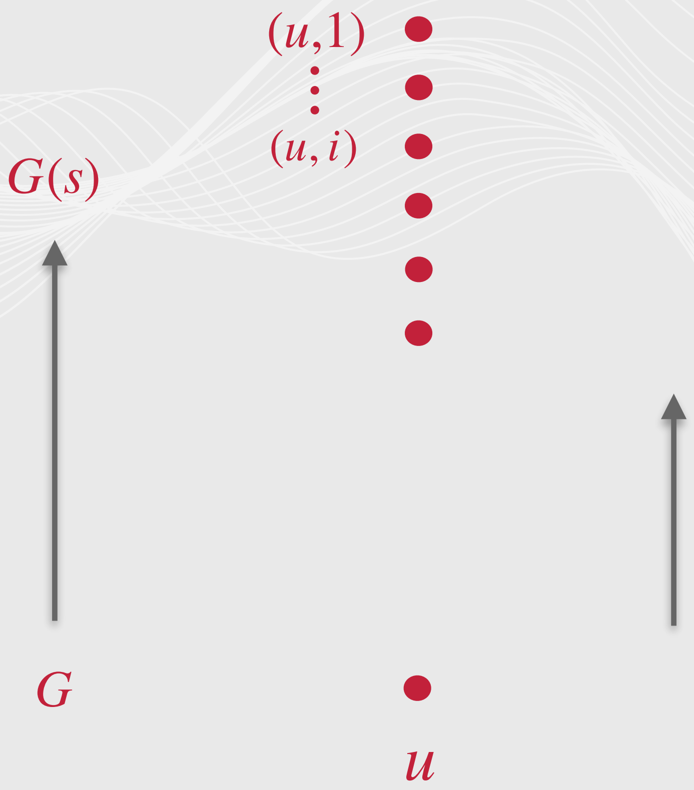
(H, ℓ) lift of a graph



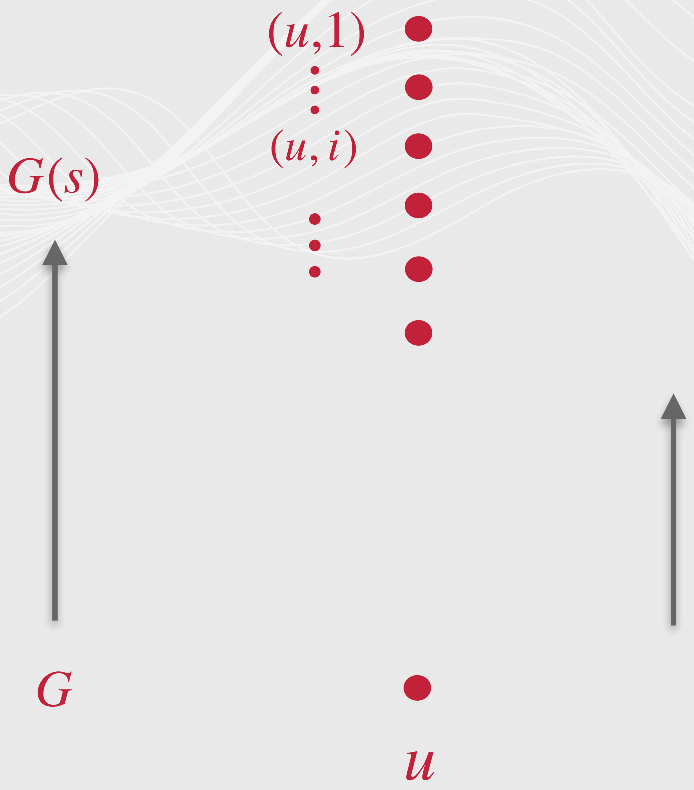
(H, ℓ) lift of a graph



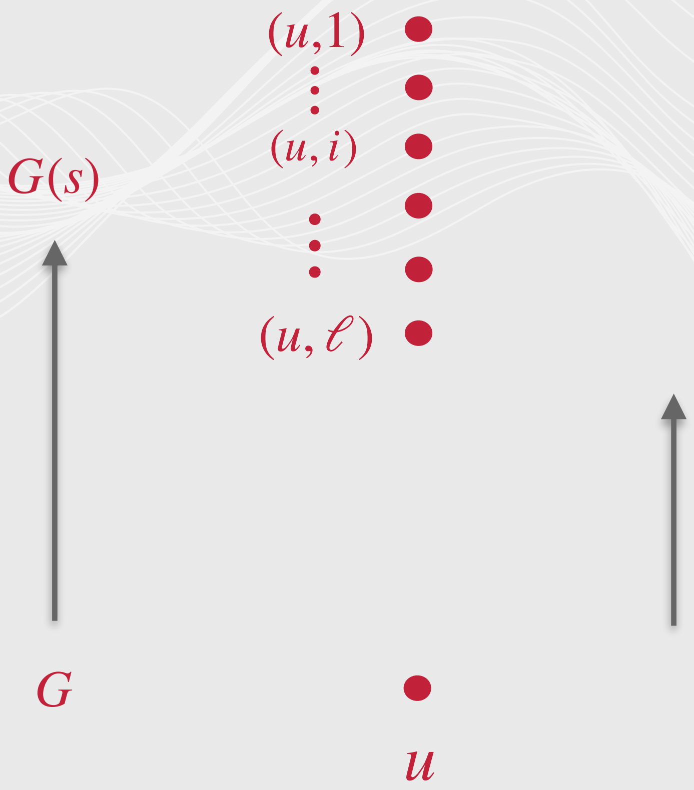
(H, ℓ) lift of a graph



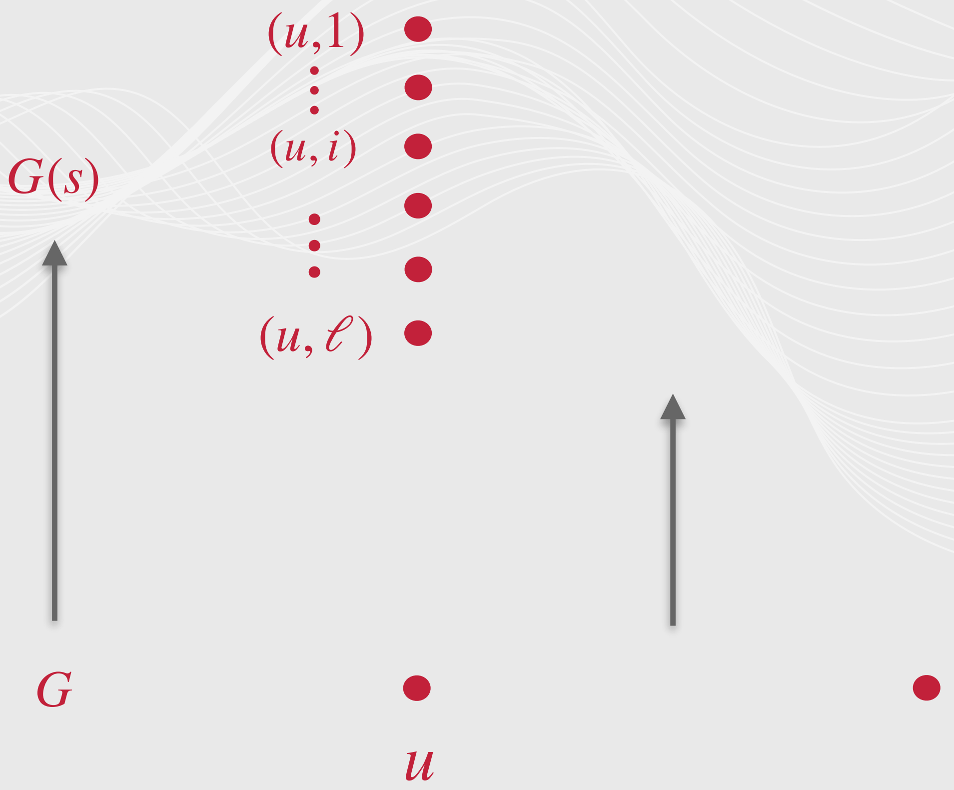
(H, ℓ) lift of a graph



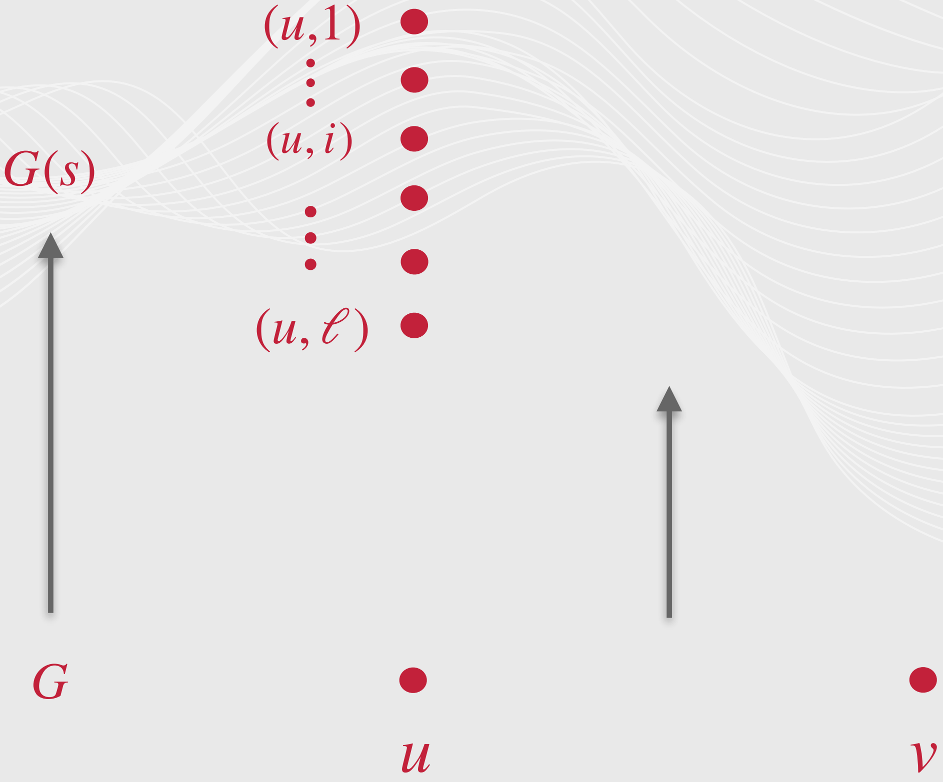
(H, ℓ) lift of a graph



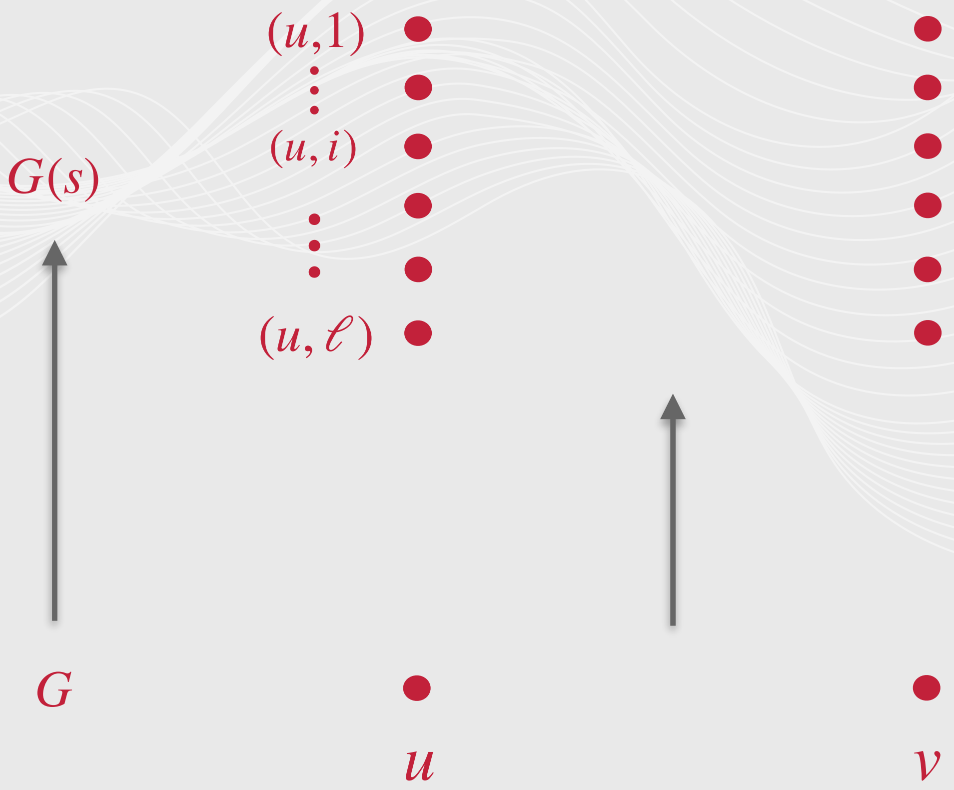
(H, ℓ) lift of a graph



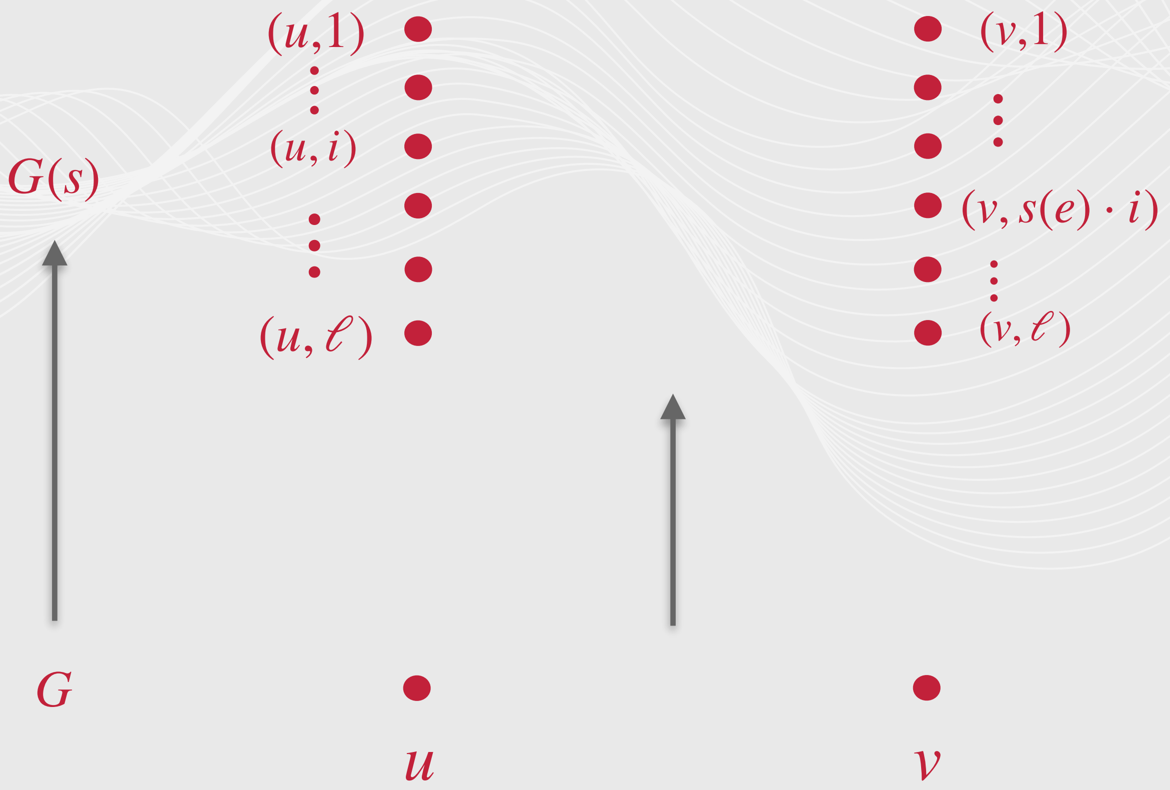
(H, ℓ) lift of a graph



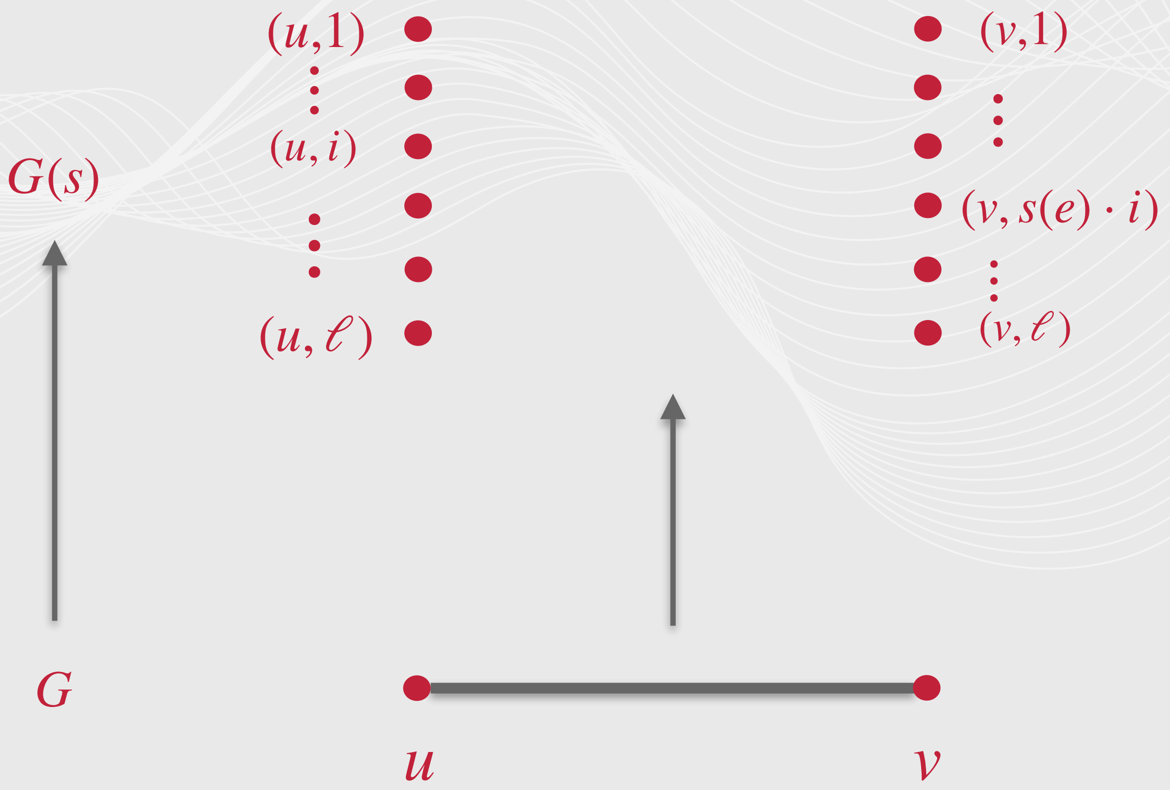
(H, ℓ) lift of a graph



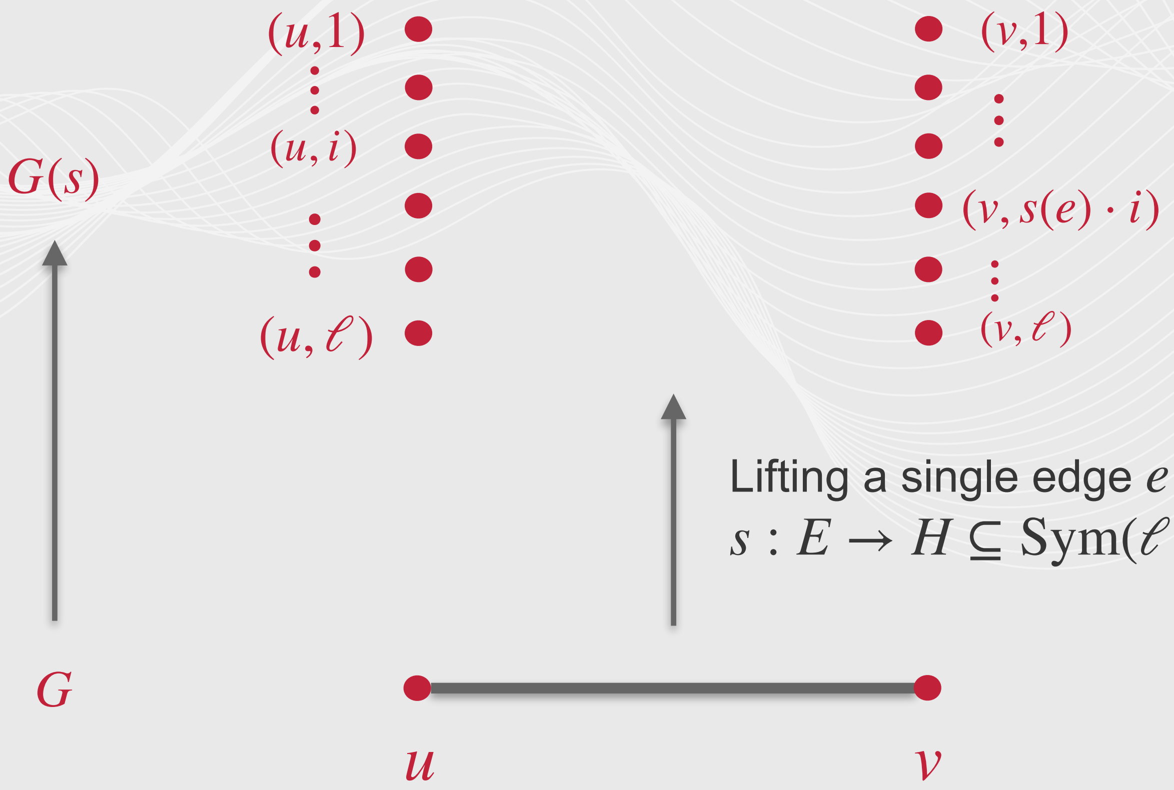
(H, ℓ) lift of a graph



(H, ℓ) lift of a graph

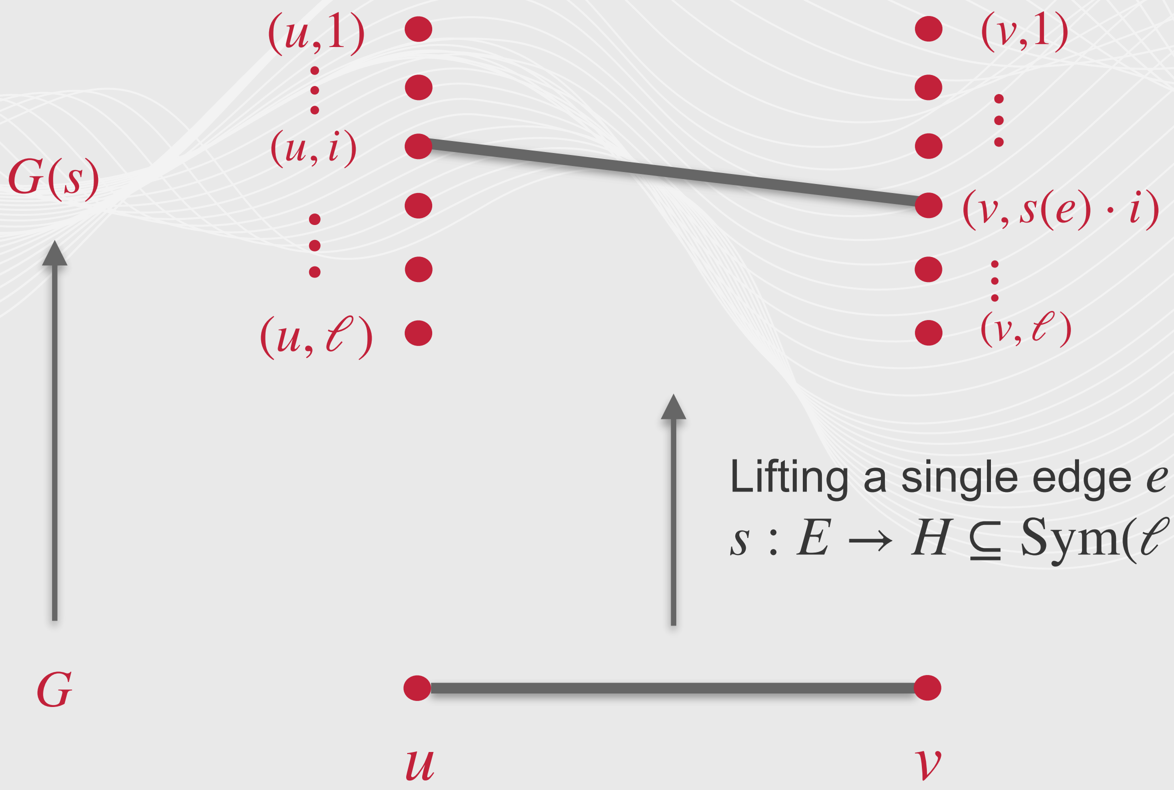


(H, ℓ) lift of a graph



Lifting a single edge e via a *signing* $s : E \to H \subseteq \mathrm{Sym}(\ell)$

(H, ℓ) lift of a graph

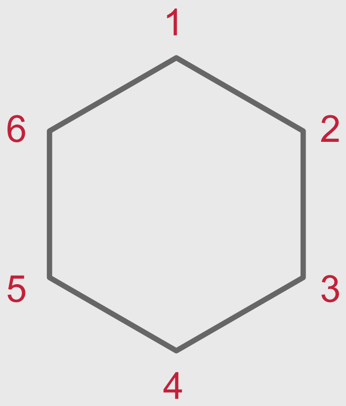


Lifting a single edge e via a *signing* $s : E \to H \subseteq \mathrm{Sym}(\ell)$

Example (revisited)

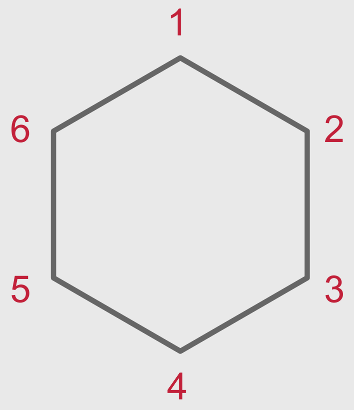
Example (revisited)

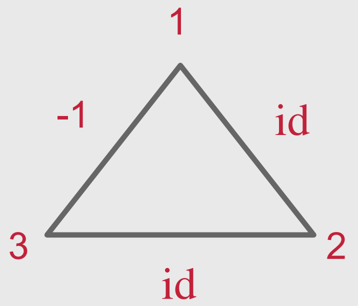
Can this cycle graph be seen as a lift of a smaller graph?



Example (revisited)

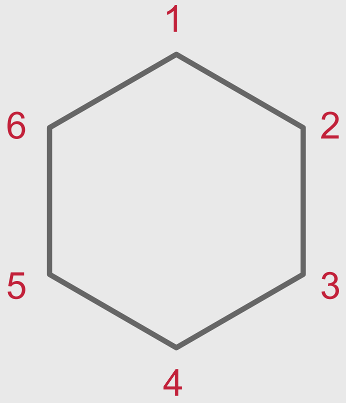
Can this cycle graph be seen as a lift of a smaller graph?

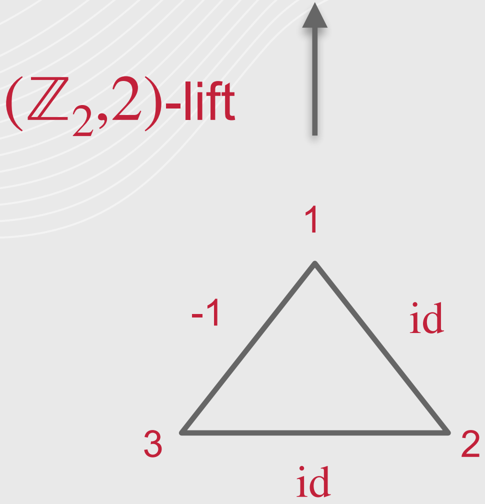




Example (revisited)

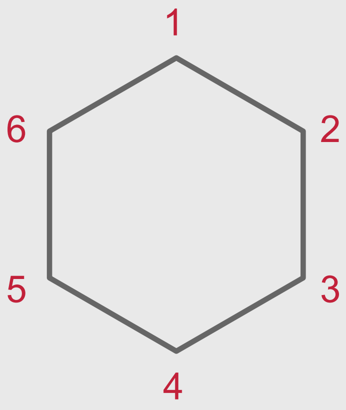
Can this cycle graph be seen as a lift of a smaller graph?

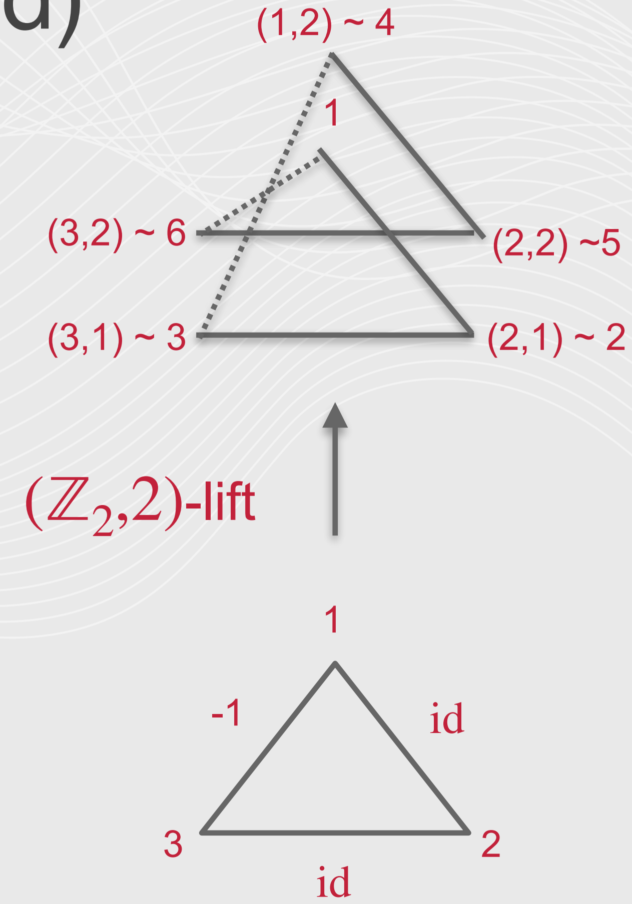




Example (revisited)

Can this cycle graph be seen as a lift of a smaller graph?





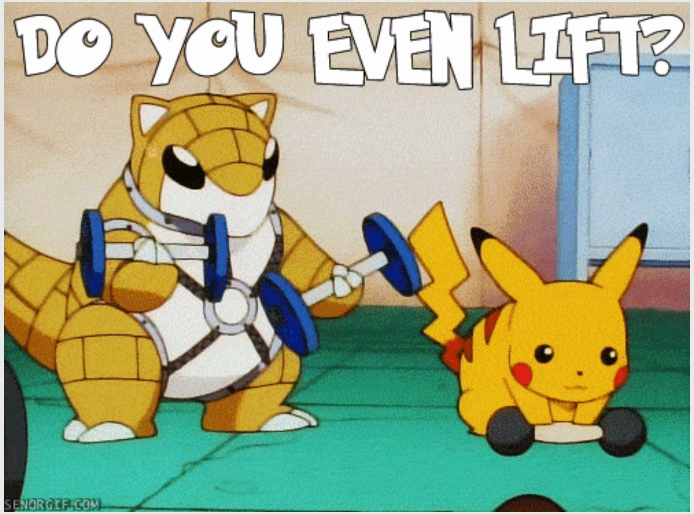
Properties of lifting

- Explicit characterization of the spectrum of lifted graph, $G(s)$.
- Preserves degree.
- If H is abelian, it possesses symmetries of H i.e.,
  - $H \subseteq \mathrm{Aut}(G(s))$.
- If G is an expander and s is random, $G(s)$ is known to be an expander*. Challenge is to explicitly construct such a signing s.

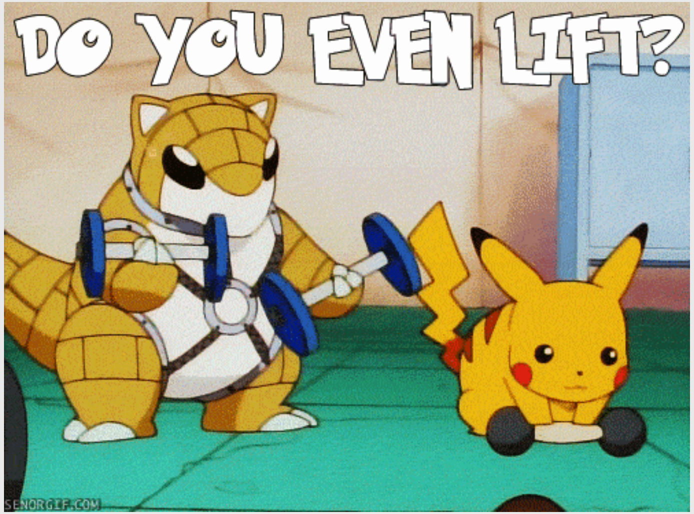
Quick history of lifting

Technique	Authors	Lift	$\lambda(G)$	Explicit
Discrepancy	[Bilu, Linial '06]	2-lift	$\sqrt{d}\log^{1.5} d$	Yes
	[Agrawal, Chandrashekharan, Kolla, Madan '16]	$(\mathbb{Z}_\ell, \ell)$	$O(\sqrt{d})$	No
Method of interlacing polynomials	[Marcus, Spielman, Srivastava '13] [Cohen '16]	2-lift	$2\sqrt{d-1}$	Yes
	[Hall, Puder, Sawin '15]	(H, ℓ) for some non-abelian		No?
Trace Power Method	[Mohanty, O'Donnell, Paredes '20]	2-lift	$2\sqrt{d-1}+\varepsilon$	Yes

Can we lift more?



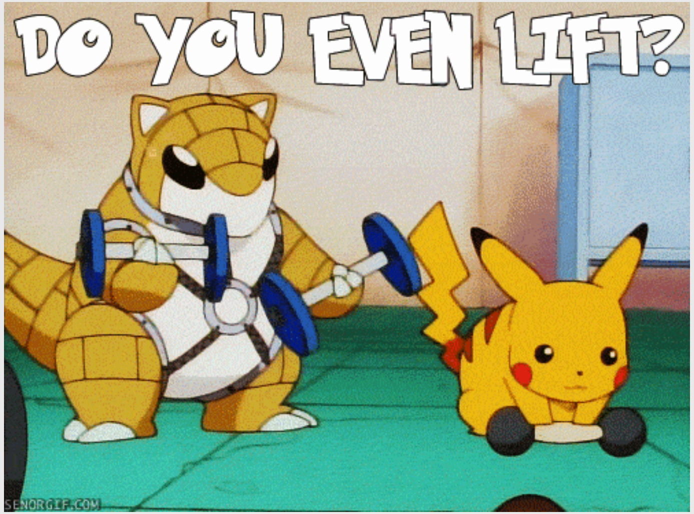
Can we lift more?



Can we lift more?

- [ACKM'16] showed that for $\mathbb{Z}_\ell$, there exists good signings for $\ell \le 2^{n/d^3}$.
  - They further show that for any abelian group H, no lift of size $\ell > \exp(nd)$ is expanding.

- The goal now is to construct $(\mathbb{Z}_\ell, \ell)$ lifts for $3 \le \ell \le 2^{nd}$.



2.

Our Results

Yes, we do lift!
And that too explicitly!

Main Result (Simplified)

Main Result (Simplified)

Theorem - For any $d \geq 3$, large enough n and "nice" $\ell(n)$, we have an explicit family of d-regular expanding graphs $\{G_n\}$ such that G_n is a $(\mathbb{Z}_{\ell(n)}, \ell(n))$-lift* of some base graph.

Main Result (Simplified)

Theorem - For any $d \geq 3$, large enough n and "nice" $\ell(n)$, we have an explicit family of d-regular expanding graphs $\{G_n\}$ such that G_n is a $(\mathbb{Z}_{\ell(n)}, \ell(n))$-lift* of some base graph.

* The result can be generalized from $\mathbb{Z}_\ell$ to any transitive abelian subgroup of $\mathrm{Sym}(\ell)$

Main Result (Simplified)

Theorem - For any $d \geq 3$, large enough n and "nice" $\ell(n)$, we have an explicit family of d-regular expanding graphs $\{G_n\}$ such that G_n is a $(\mathbb{Z}_{\ell(n)}, \ell(n))$-lift* of some base graph.

* The result can be generalized from $\mathbb{Z}_\ell$ to any transitive abelian subgroup of $\mathrm{Sym}(\ell)$

Main Result

Technique	Authors	Lift	$\lambda(G)$	Explicit
Discrepancy	[BL06]	$(\mathbb{Z}_2, 2)$	$\tilde{O}\left(\sqrt{d}\right)$	Yes
	[ACKM16]	$(\mathbb{Z}_\ell, \ell)$ $\ell \le \exp(n/d^3)$	$O\left(\sqrt{d}\right)$	No
	This work	$(\mathbb{Z}_\ell, \ell)$ $\ell = \exp(\Theta(n))$	$\tilde{O}\left(\sqrt{d}\right)$	Yes
Trace Power Method	[MOP20]	$(\mathbb{Z}_2, 2)$	$2\sqrt{d-1} + \varepsilon$	Yes
	This work	$(\mathbb{Z}_\ell, \ell)$ $\ell \le \exp(n^{\delta(d,\varepsilon)})$	$2\sqrt{d-1} + \varepsilon$	Yes
		$(\mathbb{Z}_\ell, \ell)$ $\ell \le \exp(n^{0.01})$	εd	

Application - LDPC Codes

Application - LDPC Codes

- [Panteleev-Kalachev '20] Given a d-regular graph G on $n\ell$ vertices such that it is a $(\mathbb{Z}_\ell, \ell)$-lift of a graph, one can construct

Application - LDPC Codes

- [Panteleev-Kalachev '20] Given a d-regular graph G on $n\ell$ vertices such that it is a $(\mathbb{Z}_\ell, \ell)$-lift of a graph, one can construct

Application - LDPC Codes

- [Panteleev-Kalachev '20] Given a d-regular graph G on $n\ell$ vertices such that it is a $(\mathbb{Z}_\ell, \ell)$-lift of a graph, one can construct
  - A *good* quasi-cyclic linear code with circulant size ℓ.

Application - LDPC Codes

- [Panteleev-Kalachev '20] Given a d-regular graph G on $n\ell$ vertices such that it is a $(\mathbb{Z}_\ell, \ell)$-lift of a graph, one can construct
  - A *good* quasi-cyclic linear code with circulant size ℓ.
  - An $[[n\ell, n, \ell]]$ quantum LDPC code.

Application - LDPC Codes

Application - LDPC Codes

- *Corollary* - We have explicit polynomial time construction of each of the following -

Application - LDPC Codes

- *Corollary* - We have explicit polynomial time construction of each of the following -

    - Good quasi-cyclic LDPC code of block length N and any circulant size up to $N/\text{polylog}(N)$ or $\Theta(N/\log(N))$.

Application - LDPC Codes

- *Corollary* - We have explicit polynomial time construction of each of the following -
  - Good quasi-cyclic LDPC code of block length N and any circulant size up to $N/\text{polylog}(N)$ or $\Theta(N/\log(N))$.
  - Quantum LDPC code with distance $\Omega(N/\log(N))$ and dimension $\Omega(\log(N))$.

Application - LDPC Codes

- *Corollary* - We have explicit polynomial time construction of each of the following -
  - Good quasi-cyclic LDPC code of block length N and any circulant size up to $N/\mathrm{polylog}(N)$ or $\Theta(N/\log(N))$.
  - Quantum LDPC code with distance $\Omega(N/\log(N))$ and dimension $\Omega(\log(N))$.
  - Quantum LDPC code with distance $\Omega(N^{1-\alpha})$ and dimension $\Theta(N^{\alpha})$ for every constant $0 < \alpha < 1$.

3.

Key Contribution

A better count of non-backtracking hikes

Trace Power method

Trace Power method

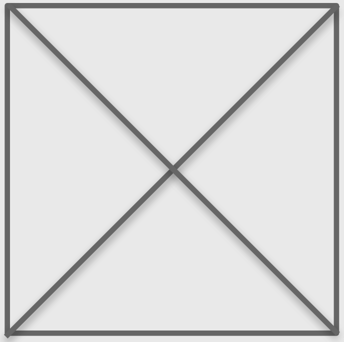
- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*

Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* $=$ *singleton-free non-backtracking walks.*

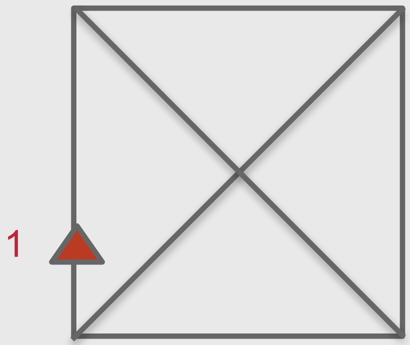
Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right| .$



Trace Power method

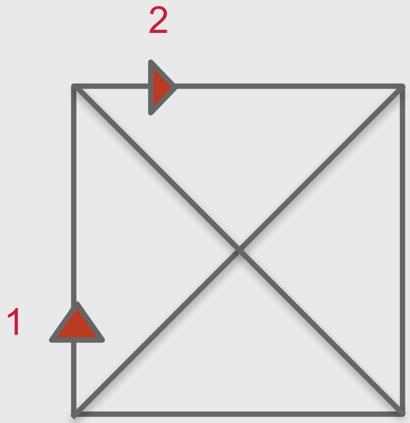
- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes = singleton-free non-backtracking walks.*
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$



Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*

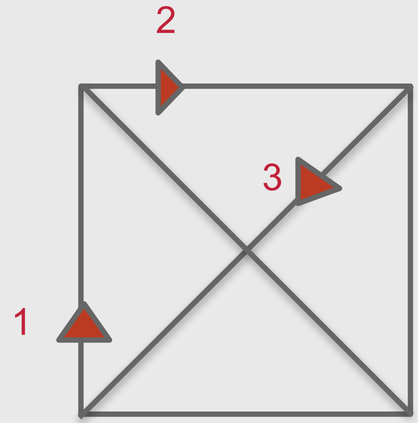
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$



Trace Power method

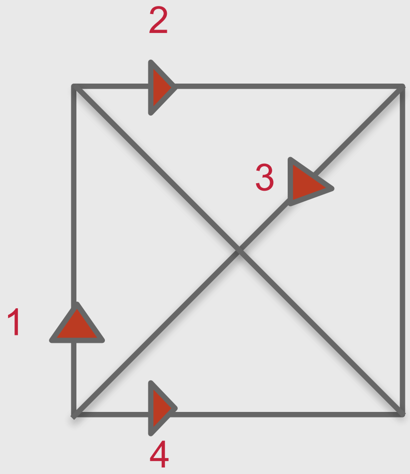
- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes = singleton-free non-backtracking walks.*

- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$



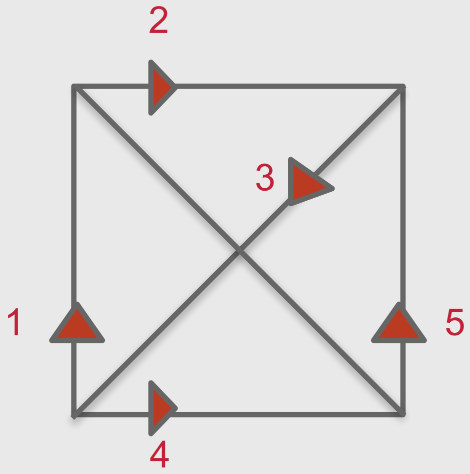
Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes = singleton-free non-backtracking walks.*
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$



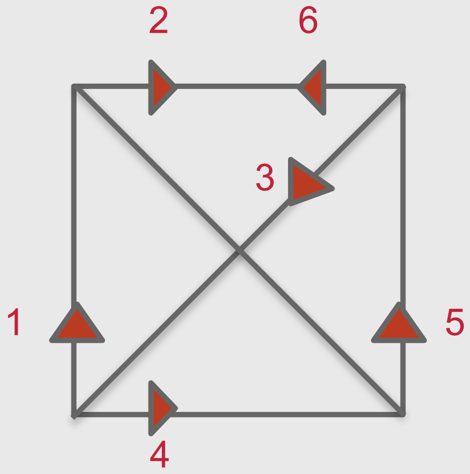
Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes = singleton-free non-backtracking walks.*
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$



Trace Power method

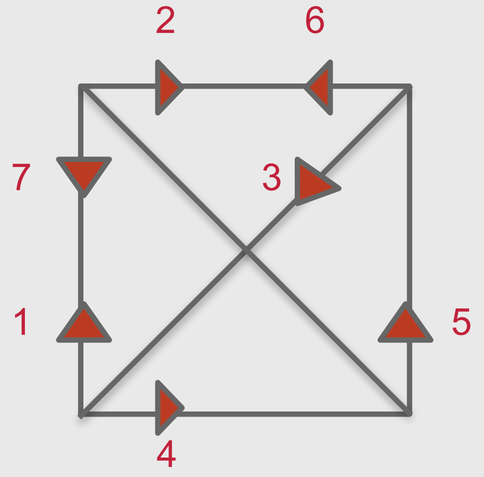
- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$



Trace Power method

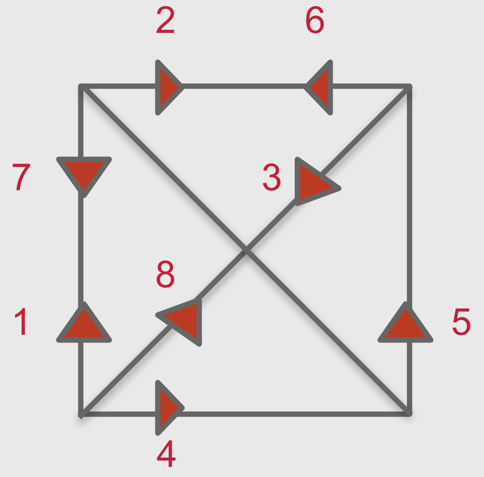
- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes = singleton-free non-backtracking walks.*

- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$



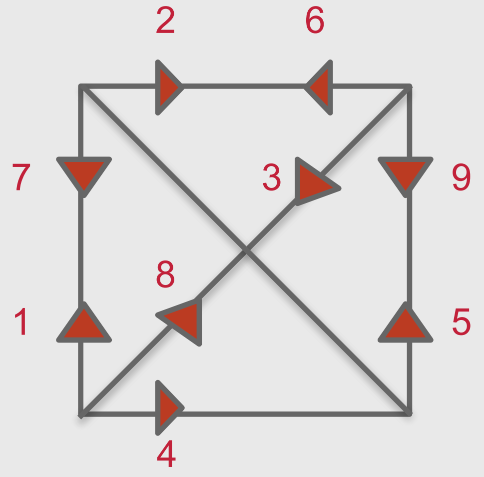
Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes = singleton-free non-backtracking walks.*
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left| \text{Hikes of length } 2k \right|.$



Trace Power method

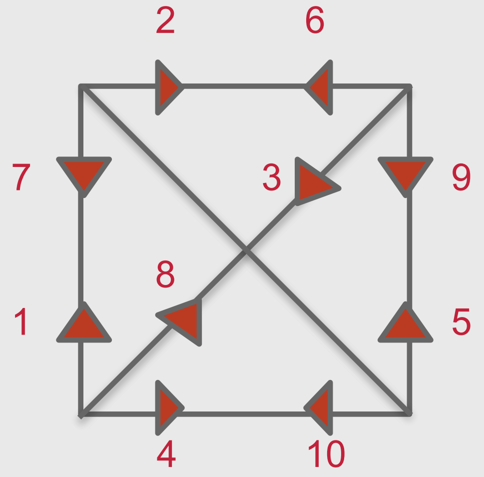
- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*
- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \le \lambda_{max}(B)^{2k} \le \mathrm{tr}((B^*)^k B^k) \le \left|\text{Hikes of length } 2k\right|.$



Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*

- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$

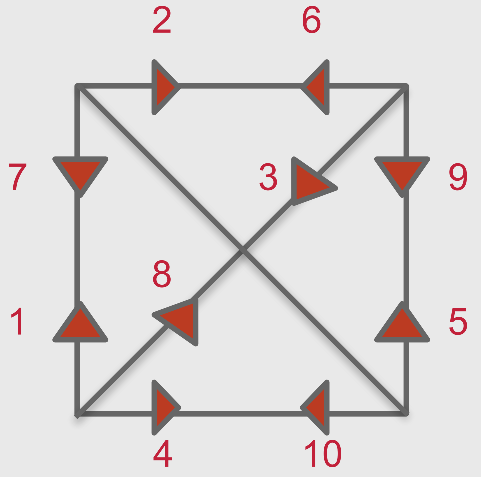


Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*

- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \le \lambda_{max}(B)^{2k} \le \mathrm{tr}((B^*)^k B^k) \le \left|\text{Hikes of length } 2k\right|.$

- Trivial Count $\sim (d-1)^{2k}$ gives a trivial eigenvalue bound of d.

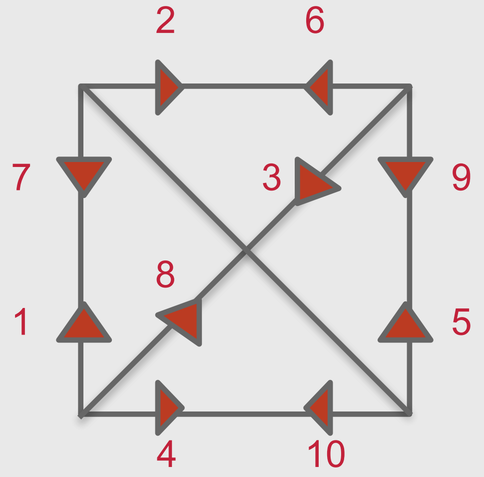


Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes = singleton-free non-backtracking walks.*

- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \le \lambda_{max}(B)^{2k} \le \mathrm{tr}((B^*)^k B^k) \le \left|\text{Hikes of length } 2k\right|.$

- Trivial Count $\sim (d-1)^{2k}$ gives a trivial eigenvalue bound of d.



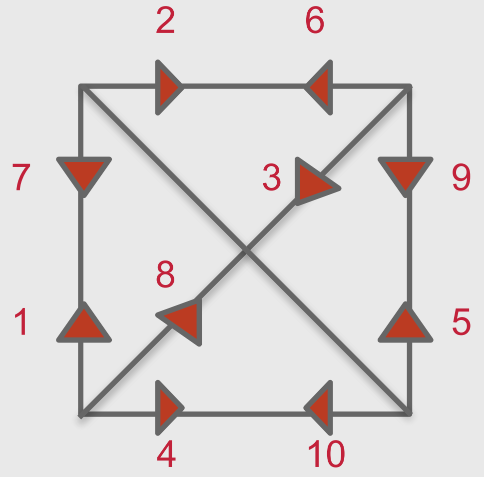
Trace Power method

- A variation of the method used by [Friedman'02], [Bordenave'19] and [MOP'20]. Define *Hikes* = *singleton-free non-backtracking walks.*

- $\frac{1}{2}\lambda_{max}(A_G)^{2k} \leq \lambda_{max}(B)^{2k} \leq \mathrm{tr}((B^*)^k B^k) \leq \left|\text{Hikes of length } 2k\right|.$

- Trivial Count $\sim (d-1)^{2k}$ gives a trivial eigenvalue bound of d.

- Ideal Count - $(d-1)^k$ would give the optimal bound of $2\sqrt{d-1}$.



Longer hikes are harder!

Longer hikes are harder!

- The length, $2k$, of the walk depends on the size of the lift ℓ.

Longer hikes are harder!

- The length, $2k$, of the walk depends on the size of the lift ℓ.

Longer hikes are harder!

- The length, $2k$, of the walk depends on the size of the lift ℓ.
- For 2-lifts, bounding walks of length $O(\log n)$ suffices which is what [MOP20] does and gives a count close to the optimal one.

Longer hikes are harder!

- The length, $2k$, of the walk depends on the size of the lift ℓ.
- For 2-lifts, bounding walks of length $O(\log n)$ suffices which is what [MOP20] does and gives a count close to the optimal one.

Longer hikes are harder!

- The length, $2k$, of the walk depends on the size of the lift ℓ.
- For 2-lifts, bounding walks of length $O(\log n)$ suffices which is what [MOP20] does and gives a count close to the optimal one.
- We extend the near-optimal bound to walks of length $O(n^{\delta(d,\varepsilon)})$ and obtain a weaker bound all the way up to $k = O(n^{0.01})$.

Counting via DFS

Counting via DFS

- Define the hike graph to be the subgraph formed by the hike.

Counting via DFS

- Define the hike graph to be the subgraph formed by the hike.
- First, count all possible hike graphs by encoding a DFS traversal.

Counting via DFS

- Define the hike graph to be the subgraph formed by the hike.
- First, count all possible hike graphs by encoding a DFS traversal.
  - Encoding 1 - At each step store whether it is a backtracking step or which of the d neighbors do we recurse to.

Counting via DFS

- Define the hike graph to be the subgraph formed by the hike.
- First, count all possible hike graphs by encoding a DFS traversal.
  - Encoding 1 - At each step store whether it is a backtracking step or which of the d neighbors do we recurse to.

Counting via DFS

- Define the hike graph to be the subgraph formed by the hike.
- First, count all possible hike graphs by encoding a DFS traversal.
  - Encoding 1 - At each step store whether it is a backtracking step or which of the d neighbors do we recurse to.
  - If $k = O\left(n^{\delta(\varepsilon)}\right)$ — Most vertices have degree 2. Compress the encoding by storing list of vertices of degree > 2.

Counting via DFS

- Define the hike graph to be the subgraph formed by the hike.
- First, count all possible hike graphs by encoding a DFS traversal.
  - Encoding 1 - At each step store whether it is a backtracking step or which of the d neighbors do we recurse to.
  - If $k = O\left(n^{\delta(\varepsilon)}\right)$ — Most vertices have degree 2. Compress the encoding by storing list of vertices of degree > 2.
- Then, count the number of hikes corresponding to a given graph.

4.

Conclusion

All good things come to an end

Summary

Summary

- We give explicit constructions of (H, ℓ)-lifted graphs for abelian H and a large range of lift sizes ℓ.

Summary

- We give explicit constructions of (H, ℓ)-lifted graphs for abelian H and a large range of lift sizes ℓ.

Summary

- We give explicit constructions of (H, ℓ)-lifted graphs for abelian H and a large range of lift sizes ℓ.

- Main method of analysis is trace power method utilizing a careful count of special walks on a large girth graph.

Summary

- We give explicit constructions of (H, ℓ)-lifted graphs for abelian H and a large range of lift sizes ℓ.

- Main method of analysis is trace power method utilizing a careful count of special walks on a large girth graph.

Summary

- We give explicit constructions of (H, ℓ)-lifted graphs for abelian H and a large range of lift sizes ℓ.
- Main method of analysis is trace power method utilizing a careful count of special walks on a large girth graph.
- As an application, we get new explicit LDPC codes — classical and quantum.

Open Problems

Open Problems

- Extend the almost-Ramanujan bound to the entire range of lift-sizes, possibly with a unified proof technique.

Open Problems

- Extend the almost-Ramanujan bound to the entire range of lift-sizes, possibly with a unified proof technique.

Open Problems

- Extend the almost-Ramanujan bound to the entire range of lift-sizes, possibly with a unified proof technique.

- Can we give strongly explicit constructions?

Open Problems

- Extend the almost-Ramanujan bound to the entire range of lift-sizes, possibly with a unified proof technique.

- Can we give strongly explicit constructions?

Open Problems

- Extend the almost-Ramanujan bound to the entire range of lift-sizes, possibly with a unified proof technique.
- Can we give strongly explicit constructions?
- Generalize the result to new families of non-abelian groups.

Thank you!

Thank you!

- Glad to hear your feedback/questions - tushant@uchicago.edu

Thank you!

- Glad to hear your feedback/questions - tushant@uchicago.edu

Thank you!

- Glad to hear your feedback/questions - tushant@uchicago.edu
- Fernando gave a (much) longer talk on the topic at IAS.

Thank you!

- Glad to hear your feedback/questions - tushant@uchicago.edu
- Fernando gave a (much) longer talk on the topic at IAS.
  - Check it out on Youtube!