Finding a Bounded-Degree **Expander** Inside a Dense One

HAL-02002377

Emanuele Natale

i3s sophia antipolis

Joint work with L. Becchetti, A. Clementi, F. Pasquale and L. Trevisan

IRISA

13 March 2019

Outline

- Definitions: Graph Expansion
- Motivation for this work
- Our Results
- Crash Course on Encoding Arguments
- Some Proof Ideas

Graph Expansion I

What is a good measure of *connectedness* for a set of nodes S?

Graph Expansion I

What is a good measure of *connectedness* for a set of nodes S?

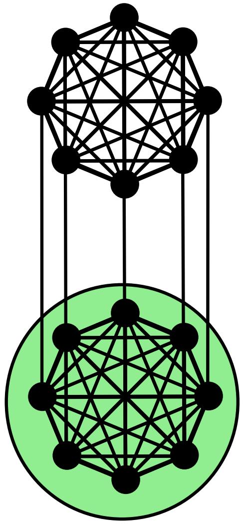
- Attempt 1. Number of edges going out of S:
  $e(S, V - S) = |\{(u, v) | u \in S, v \in V - S\}|$

Graph Expansion I

What is a good measure of *connectedness* for a set of nodes S?

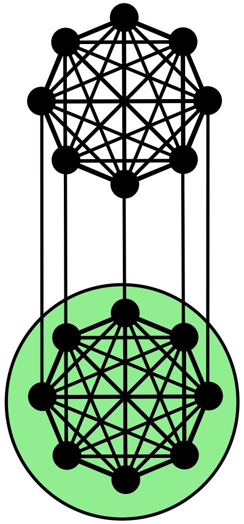
- Attempt 1. Number of edges going out of S:
  $e(S, V - S) = |\{(u, v) | u \in S, v \in V - S\}|$

  Problem: big sets are better than small ones



Graph Expansion I

What is a good measure of *connectedness* for a set of nodes S?



- Attempt 1. Number of edges going out of S: $e(S, V - S) = |\{(u, v)|u \in S, v \in V - S\}|$

  Problem: big sets are better than small ones

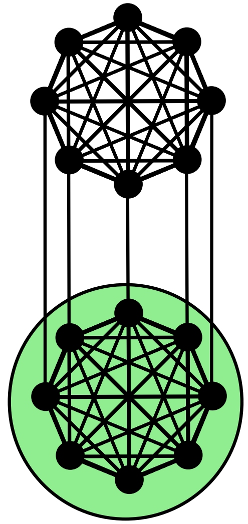
- Attempt 2. We also divide by the sum of its degrees $vol(S) = \sum_{u \in S} d_u$: $\frac{e(S,V-S)}{vol(S)}$

Graph Expansion I

What is a good measure of *connectedness* for a set of nodes S?

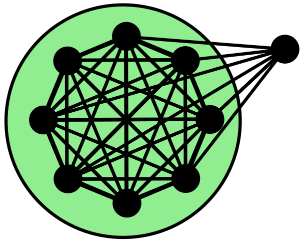
- Attempt 1. Number of edges going out of S: $e(S, V - S) = |\{(u, v) | u \in S, v \in V - S\}|$

  Problem: big sets are better than small ones



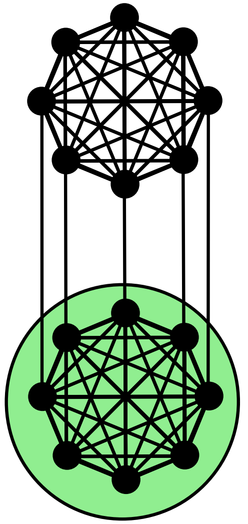
- Attempt 2. We also divide by the sum of its degrees $vol(S) = \sum_{u \in S} d_u$: $\frac{e(S, V-S)}{vol(S)}$

  Problem: Very big sets have big $vol(S)$



Graph Expansion I

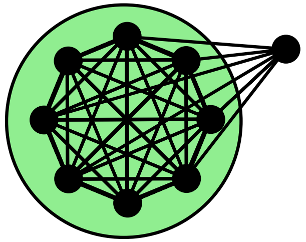
What is a good measure of *connectedness* for a set of nodes S?



- Attempt 1. Number of edges going out of S: $e(S, V-S) = |\{(u,v) | u \in S, v \in V-S\}|$

  Problem: big sets are better than small ones

- Attempt 2. We also divide by the sum of its degrees $vol(S) = \sum_{u \in S} d_u$: $\frac{e(S,V-S)}{vol(S)}$

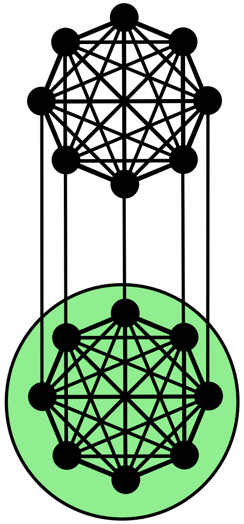
  

  Problem: Very big sets have big $vol(S)$

- Attempt 3. We consider the "worst" between S and $V-S$: $\frac{e(S,V-S)}{\min\{vol(S), vol(V-S)\}}$

Graph Expansion I

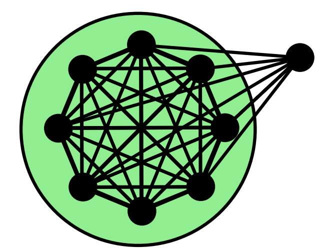
What is a good measure of *connectedness* for a set of nodes S?



- Attempt 1. Number of edges going out of S: $e(S, V-S) = |\{(u,v) | u \in S, v \in V-S\}|$

  Problem: big sets are better than small ones

- Attempt 2. We also divide by the sum of its degrees $vol(S) = \sum_{u \in S} d_u$: $\frac{e(S,V-S)}{vol(S)}$

  Problem: Very big sets have big $vol(S)$

- Attempt 3. We consider the "worst" between S and $V-S$: $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$ → conductance

  Examples: Giakkoupis et al. JACM'18 and Giakkoupis SODA'14 for another expansion measure.

Graph Expansion II

In regular graphs $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$ is equivalent to $\phi(S) = \frac{e(S,V-S)}{vol(S)}$ assuming $S \leq \frac{n}{2}$

Graph Expansion II

In regular graphs $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$ is equivalent to $\phi(S) = \frac{e(S,V-S)}{vol(S)}$ assuming $S \leq \frac{n}{2}$

Interpretation. In regular graphs, $\phi(S) = \Pr(\text{random walk on random node of } S \text{ exits it})$

Graph Expansion II

In regular graphs $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$ is equivalent to $\phi(S) = \frac{e(S,V-S)}{vol(S)}$ assuming $S \leq \frac{n}{2}$

Interpretation. In regular graphs, $\phi(S) = \Pr(\text{random walk on random node of } S \text{ exits it})$

Graph G is ϵ-expander if $\min_S \phi(S) \geq \epsilon$

Graph Expansion II

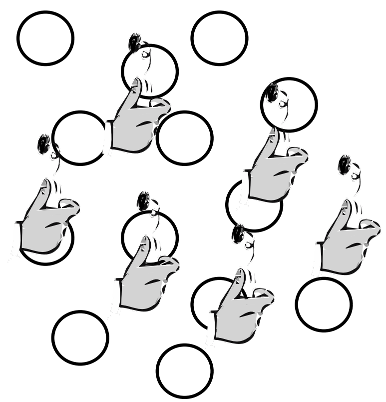
In regular graphs $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$ is equivalent to $\phi(S) = \frac{e(S,V-S)}{vol(S)}$ assuming $S \leq \frac{n}{2}$

Interpretation. In regular graphs, $\phi(S) = \Pr(\text{random walk on random node of } S \text{ exits it})$

Graph G is ϵ-expander if $\min_S \phi(S) \geq \epsilon$

Example:

In an Erős-Rényi graph $G_{n,p}$, include each edge with prob p.



Graph Expansion II

In regular graphs $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$ is equivalent to $\phi(S) = \frac{e(S,V-S)}{vol(S)}$ assuming $S \leq \frac{n}{2}$

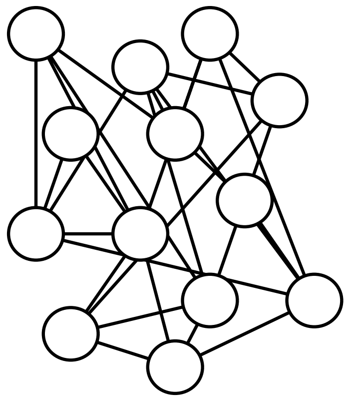
Interpretation. In regular graphs, $\phi(S) = \Pr(\text{random walk on random node of } S \text{ exits it})$

Graph G is ϵ-expander if $\min_S \phi(S) \geq \epsilon$

Example:

In an Erős-Rényi graph $G_{n,p}$, include each edge with prob p.

For any $p \gg \frac{\log n}{n}$, they are good expanders with high probability.



Expander Mixing Lemma

Expanders can be studied using **linear algebra**
(*Spectral Graph Theory*)

Expander Mixing Lemma

Expanders can be studied using **linear algebra** (*Spectral Graph Theory*)

Lemma. For any subset S of nodes of a Δ-regular graph with 2nd-largest eigenvalue of adjecency matrix λ :

$$e(S,S) \leq |S|(\frac{|S|}{2}\frac{\Delta}{n} + \frac{\lambda}{2})$$

Expander Mixing Lemma

Expanders can be studied using **linear algebra** (*Spectral Graph Theory*)

Lemma. For any subset S of nodes of a Δ-regular graph with 2nd-largest eigenvalue of adjecency matrix λ :

$$e(S,S) \leq |S|(\frac{|S|}{2}\frac{\Delta}{n} + \frac{\lambda}{2})$$

Proof. A adjacency matrix, 1_S indicator vector of S, J all-1 matrix. We observe

$$2e(S,S) = 1_S^T A 1_S \text{ and } 1_S^T(\frac{\Delta}{n}J)1_S = \frac{\Delta}{n}|S|$$

Expander Mixing Lemma

Expanders can be studied using **linear algebra** (*Spectral Graph Theory*)

Lemma. For any subset S of nodes of a Δ-regular graph with 2nd-largest eigenvalue of adjecency matrix λ :

$$e(S,S) \leq |S|(\frac{|S|}{2}\frac{\Delta}{n} + \frac{\lambda}{2})$$

Proof. A adjacency matrix, 1_S indicator vector of S, J all-1 matrix. We observe

$$2e(S,S) = 1_S^T A 1_S \text{ and } 1_S^T(\frac{\Delta}{n}J)1_S = \frac{\Delta}{n}|S|$$

Hence

$$2e(S,S) - \frac{\Delta}{n}|S| = 1_S^T(A - \frac{\Delta}{n}J)1_S \leq \lambda||1_S||^2 = \lambda|S|$$

↓

λ is the largest eigenvalue

Motivations for this Work I

Distributed construction of constant-degree expanders

- Corollary of Marcus-Spielman-Srivastava proof's of the Kadison-Singer conjecture [Ann. of Math. '15]:
  Every dense expander has a *constant-degree subgraph* which is also an expander.

Motivations for this Work I

Distributed construction of constant-degree expanders

- Corollary of Marcus-Spielman-Srivastava proof's of the Kadison-Singer conjecture [Ann. of Math. '15]:
  Every dense expander has a *constant-degree subgraph* which is also an expander.

  But the proof is non-constructive.

Motivations for this Work I

Distributed construction of constant-degree expanders

- Corollary of Marcus-Spielman-Srivastava proof's of the Kadison-Singer conjecture [Ann. of Math. '15]:
  Every dense expander has a *constant-degree subgraph* which is also an expander.

  But the proof is non-constructive.

- Several works propose complicated distributed construction of expanders:
  - Law and Siu [INFOCOM'03]: incremental construction using Hamiltonian cycles

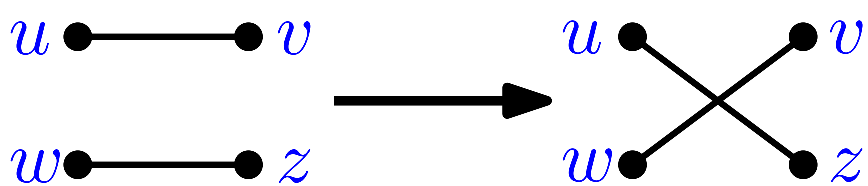
Motivations for this Work I

Distributed construction of constant-degree expanders

- Corollary of Marcus-Spielman-Srivastava proof's of the Kadison-Singer conjecture [Ann. of Math. '15]:
  Every dense expander has a *constant-degree subgraph* which is also an expander.

  But the proof is non-constructive.

- Several works propose complicated distributed construction of expanders:
  - Law and Siu [INFOCOM'03]: incremental construction using Hamiltonian cycles
  - Allen-Zhu et al. [SODA'16]: start with a $\Omega(\log n)$-regular graph and increase its expansion

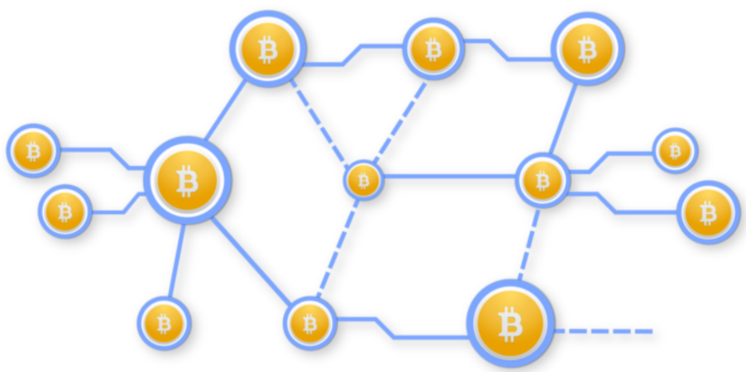


Bonus Motivation II

- Parallel algorithms for *sparsifying* a graph don't achieve sublogarithmic degree and assume weighted edges

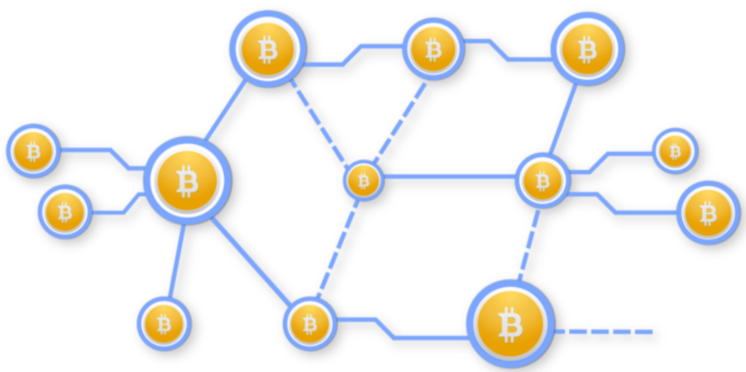
Bonus Motivation II

- Parallel algorithms for *sparsifying* a graph don't achieve sublogarithmic degree and assume weighted edges
- Model the way nodes create bounded-degree overlay networks in real distributed protocols, such as in peer-to-peer protocols (BitTorrent) or in distributed ledger protocols (Bitcoin)



Bonus Motivation II

- Parallel algorithms for *sparsifying* a graph don't achieve sublogarithmic degree and assume weighted edges
- Model the way nodes create bounded-degree overlay networks in real distributed protocols, such as in peer-to-peer protocols (BitTorrent) or in distributed ledger protocols (Bitcoin)



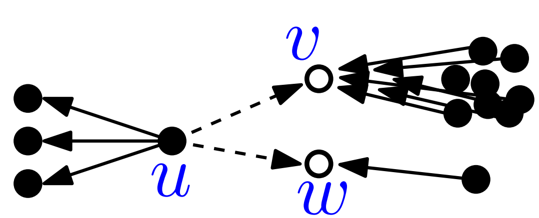
- A distributed construction of a constant-degree graph implies a *constant-load balancing* algorithm. Previous works obtain almost-tight load balancing in polynomial time (Berenbrink et al., SPAA'14)

Our Algorithm: RAES

Algorithm 1 RAES(G, d, c)

$H :=$ empty directed graph over the node set V
while H has nodes of outdegree $< d$ **do**
  PHASE 1: ▷ $\mathrm{d}^{\mathrm{out}}_{\mathrm{v}}$: current outdegree of v in H
  for each node $v \in V$ **do**
    $v \in V$ picks $d - \mathrm{d}^{\mathrm{out}}_{\mathrm{v}}$ neighbors in G uniformly at random
    v submits a connection request to each of them
  end for
  PHASE 2: ▷ $\mathrm{d}^{\mathrm{in}}_{\mathrm{v}}$: current indegree of v in H
  for each node $v \in V$ **do**
    if v received $\leqslant cd - \mathrm{d}^{\mathrm{in}}_{\mathrm{v}}$ connection requests in the previous phase **then**
      v accepts all of them and the corresponding directed edges are added to H
    else
      v rejects all connection requests received in Phase 1
    end if
  end for
end while
Replace each directed edge by an undirected one
return H

Example with $d = 5$



u is missing 2 connections.
u asks to connect to w and v.
v has already cd incoming connections and refuses u's requests.

Our Result

Theorem.
For every $d \gg 1$, $0 < \alpha \leq 1$, $c \gg \frac{1}{\alpha^2}$, and αn-regular graph G, w.h.p. RAES(G, d, c) runs in $\mathcal{O}(\log n)$ parallel rounds with message complexity is $\mathcal{O}(n)$.
Moreover, if G's 2nd-largest eigenvalue λ of normalized adjacency matrix is $\leq \epsilon\alpha^2$, then w.h.p. RAES(G, d, c) creates a ϵ-expander with degrees between d and $d(c+1)$.

Our Result

Theorem.
For every $d \gg 1$, $0 < \alpha \leq 1$, $c \gg \frac{1}{\alpha^2}$, and αn-regular graph G, w.h.p. RAES(G, d, c) runs in $\mathcal{O}(\log n)$ parallel rounds with message complexity is $\mathcal{O}(n)$.
Moreover, if G's 2nd-largest eigenvalue λ of normalized adjacency matrix is $\leq \epsilon\alpha^2$, then w.h.p. RAES(G, d, c) creates a ϵ-expander with degrees between d and $d(c+1)$.

Proof Technique: *Encoding Argument*
(omitted: message complexity using martingale theory)

Encoding Lemma.
If X finite set and
$C : X \to \{0,1\}^*$ a (partial & prefix-free) encoding of X then

$$\Pr_{x \sim Unif(X)}(|C(x)| \leq \log|X| - s) \leq 2^{-s}$$

Encoding Arguments

Encoding Lemma.
If X finite set and $C : X \to \{0,1\}^*$ a (partial & prefix-free) encoding of X then

$$\Pr_{x \sim Unif(X)}(|C(x)| \leq \log|X| - s) \leq 2^{-s}$$

Proof. $\frac{2^{\log|X|-s}}{|X|} \leq 2^{-s}$.

Encoding Arguments

Encoding Lemma.
If X finite set and
$C : X \to \{0,1\}^*$ a (partial & prefix-free) encoding of X then

$$\Pr_{x \sim Unif(X)}(|C(x)| \le \log|X| - s) \le 2^{-s}$$

Proof. $\frac{2^{\log|X|-s}}{|X|} \le 2^{-s}$.

Suggested reading: P. Morin et al. *Encoding Arguments*, ACM Comp. Surveys '17.

Encoding Argument Example

Flip a coin n times: $0110010\cdots$.
Probability of $\log n + s$ consecutive heads?

Encoding Argument Example

Flip a coin n times: $0110010\cdots$.
Probability of $\log n + s$ consecutive heads?

Call B a *bad substring* of $\log n + s$ consecutive heads.
Consider encoding C_B for strings containing B:

(index i of first bit of B , all other bits of the string except those at entry $i, i+1, \ldots, i+\log n + s$)

$\log n$ bits — $n - (\log n + s)$ bits

Encoding Argument Example

Flip a coin n times: $0110010\cdots$.
Probability of $\log n + s$ consecutive heads?

Call B a *bad substring* of $\log n + s$ consecutive heads.
Consider encoding C_B for strings containing B:

(index i of first bit of B , all other bits of the string except those at entry $i, i+1, \ldots, i+\log n+s$)

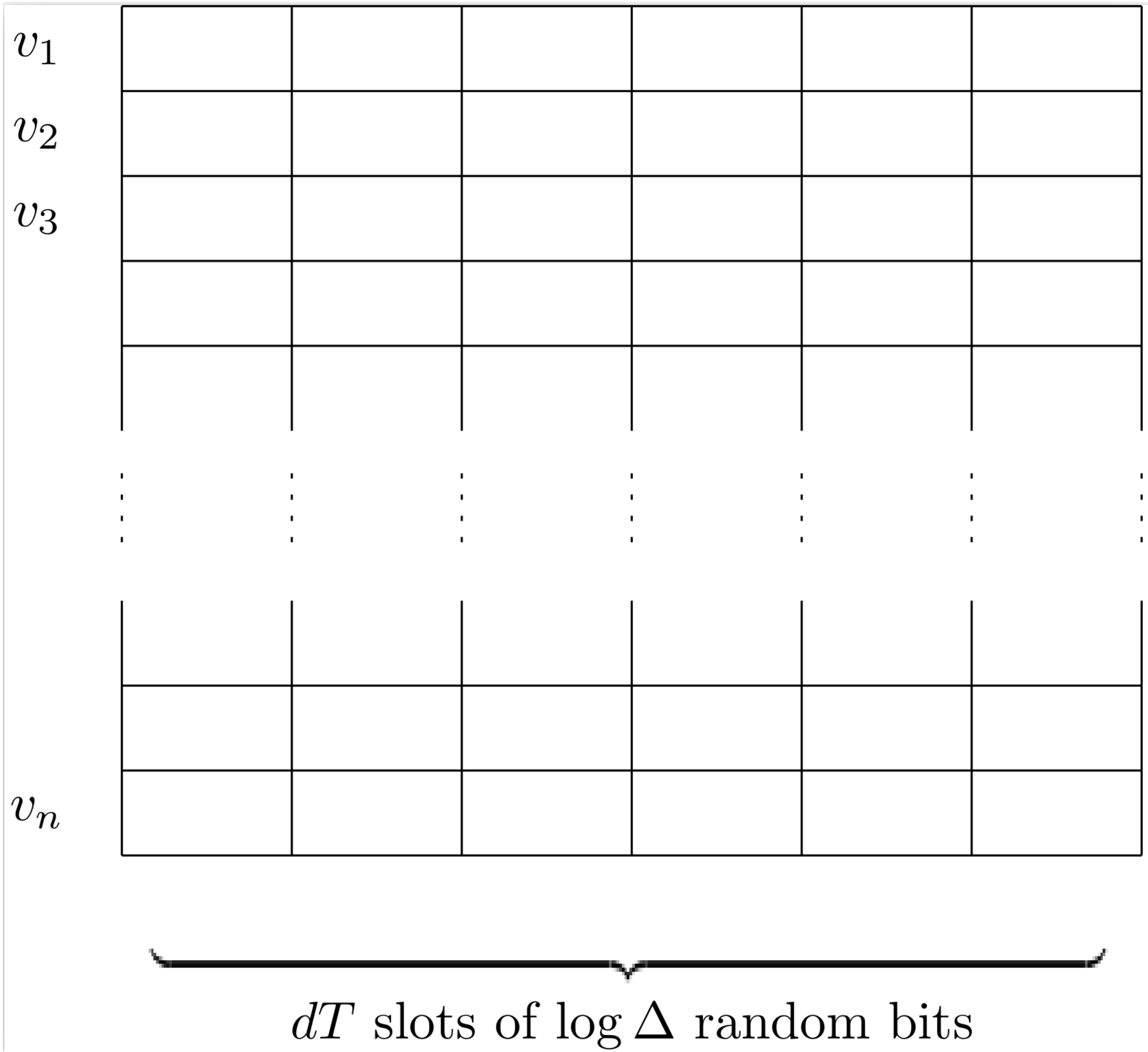
$\log n$ bits, $n-(\log n+s)$ bits

By the Encoding Lemma
$\Pr(|C_B(x)| \le \log|X| - s) = \Pr(|C_B(x)| \le n - s) \le 2^{-s}$

Encoding Arg. for Running Time (Warm Up)

Implementation: For each node v_i, array of dT entries of $\log \Delta$ bits

If RAES doesn't terminate in $O(\log n)$ rounds there exist node v with a rejected request at each round



Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$

Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$
- v's request ℓ_v: $2 \log \ell_v$

Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$
- v's request ℓ_v: $2 \log \ell_v$
- v's accepted requests: $2 \log d'$

Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$
- v's request ℓ_v: $2 \log \ell_v$
- v's accepted requests: $2 \log d'$
- position of v's accepted requests in ℓ_v: $\log \binom{\ell_v}{d'}$

Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$
- v's request ℓ_v: $2 \log \ell_v$
- v's accepted requests: $2 \log d'$
- position of v's accepted requests in ℓ_v: $\log \binom{\ell_v}{d'}$
- destinations of accepted requests: $d' \log \Delta$

Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$
- v's request ℓ_v: $2 \log \ell_v$
- v's accepted requests: $2 \log d'$
- position of v's accepted requests in ℓ_v: $\log \binom{\ell_v}{d'}$
- destinations of accepted requests: $d' \log \Delta$
- destinations of rejected requests: $(\ell_v - d') \log \frac{n}{c}$

Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$
- v's request ℓ_v: $2 \log \ell_v$
- v's accepted requests: $2 \log d'$
- position of v's accepted requests in ℓ_v: $\log \binom{\ell_v}{d'}$
- destinations of accepted requests: $d' \log \Delta$
- destinations of rejected requests: $(\ell_v - d') \log \frac{n}{c}$

Observation: at each round there are at most $\frac{n}{c}$ rejecting nodes

Encoding for Always-Rejected v

We encode with the following bits

- v's identity: $\log n$
- v's request ℓ_v: $2 \log \ell_v$
- v's accepted requests: $2 \log d'$
- position of v's accepted requests in ℓ_v: $\log \binom{\ell_v}{d'}$
- destinations of accepted requests: $d' \log \Delta$
- destinations of rejected requests: $(\ell_v - d') \log \frac{n}{c}$

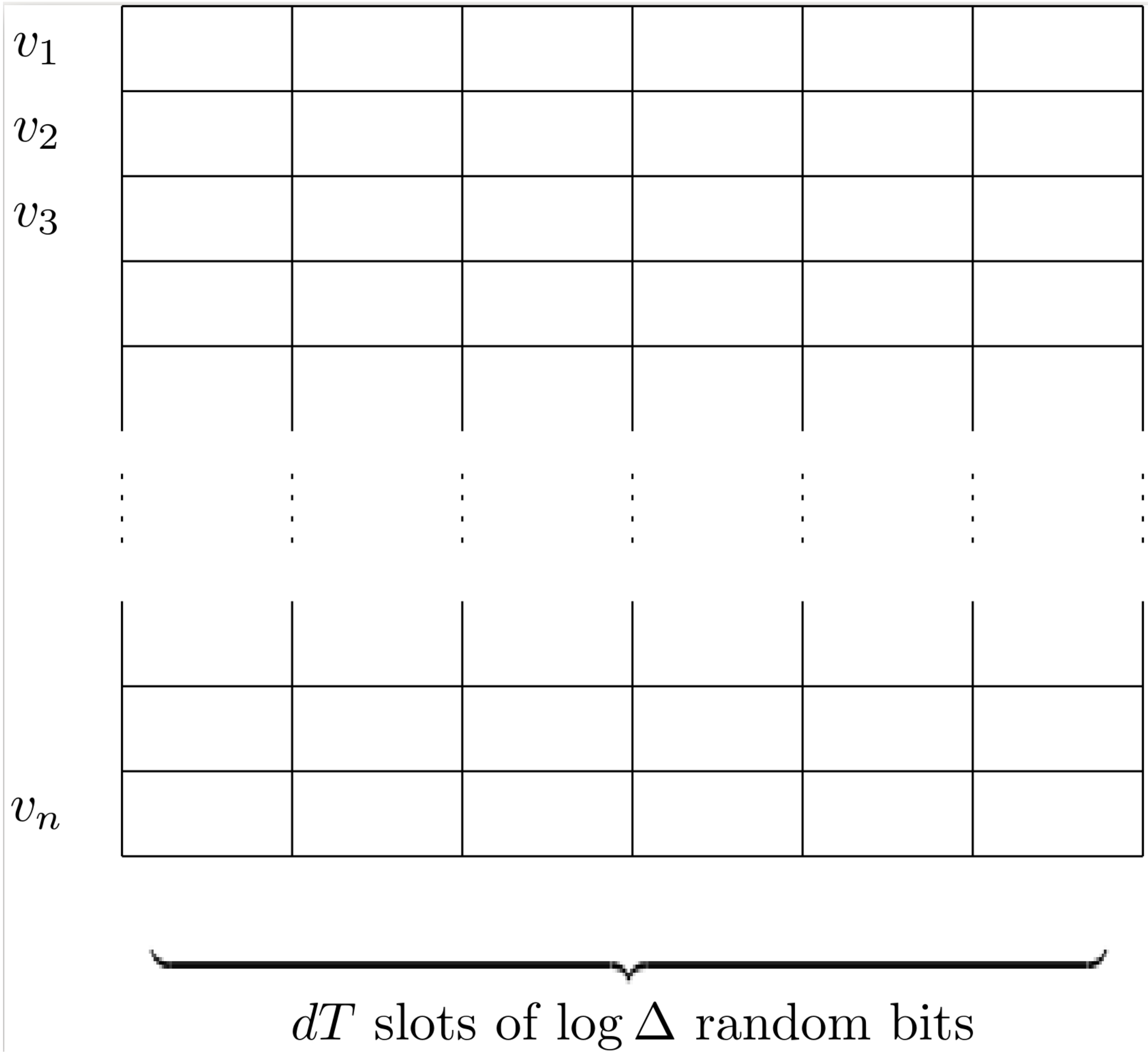
Observation: at each round there are at most $\frac{n}{c}$ rejecting nodes

After calculations we see that we save $\frac{1}{2}\ell_v \log(\alpha c) - \log n = \Omega(\log n)$

Encoding Argument for Expansion

Implementation: For each node v_i, array of dT entries of $\log \Delta$ bits

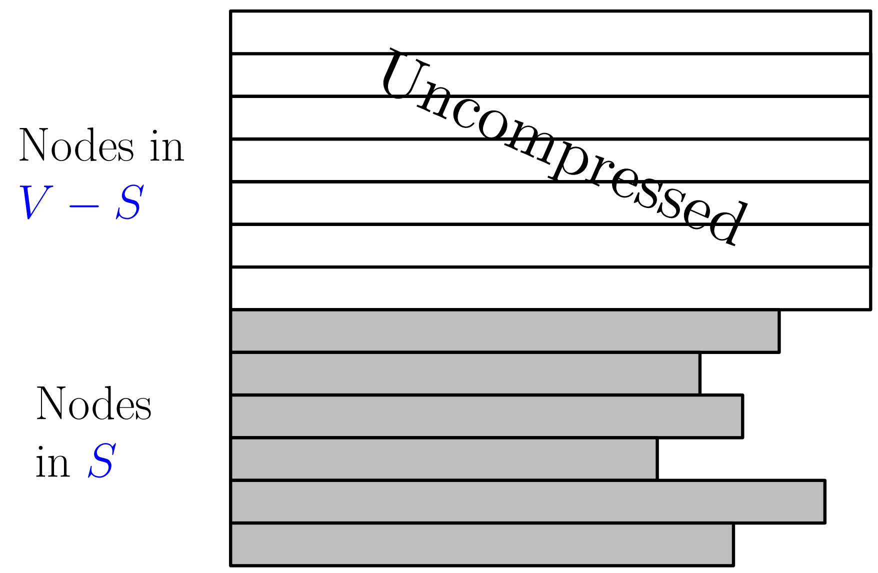
We show that if the execution results in a non-expander, then it can be represented with $ndt \log \Delta - \Omega(\log n)$ bits



Compressing the Non-Expanding Set

Encoding:

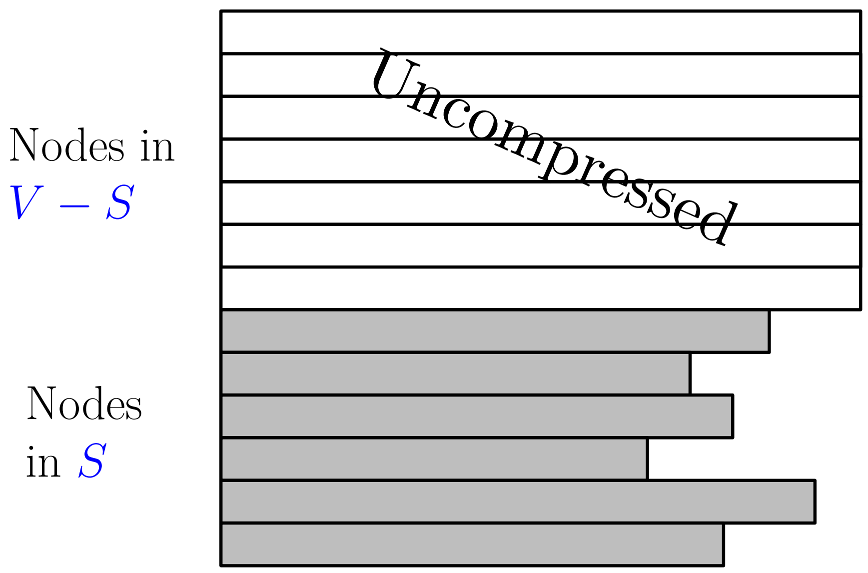
- Randomness of $V - S$
- Set S: $\log |S| + \log \binom{n}{s}$



Compressing the Non-Expanding Set

Encoding:

- Randomness of $V - S$
- Set S: $\log |S| + \log \binom{n}{s}$
- Accepted connections: $\sum_{v \in S} 2 \log \ell_v + \log \binom{\ell_v}{d}$

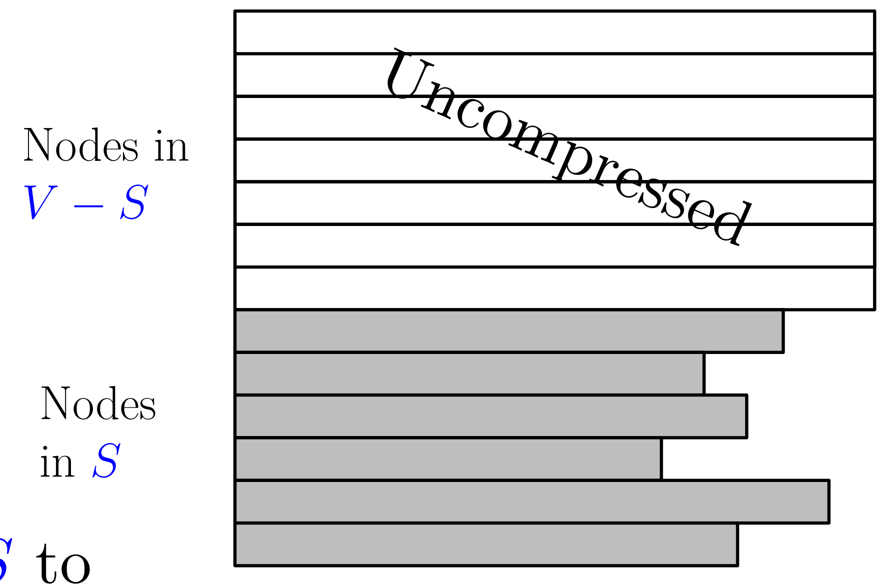


Compressing the Non-Expanding Set

Encoding:

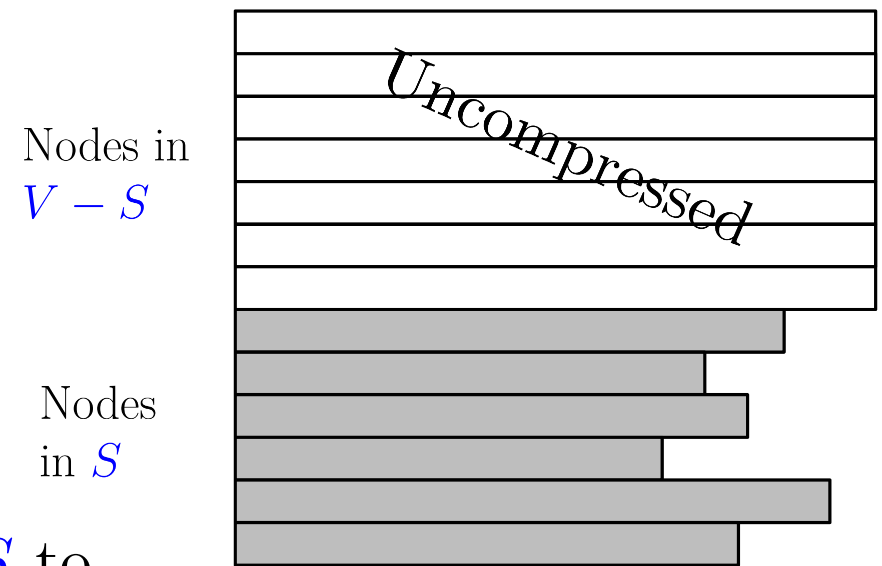
- Randomness of $V - S$
- Set S: $\log |S| + \log \binom{n}{s}$
- Accepted connections: $\sum_{v \in S} 2 \log \ell_v + \log \binom{\ell_v}{d}$
- Accepted connections from S to $V - S$: $\sum_{v \in S} 2 \log(\epsilon_v d) + \log \binom{d}{\epsilon_v d}$

ϵ_v: fraction of v's accepted connections towards $V - S$



Compressing the Non-Expanding Set

Encoding:



- Randomness of $V - S$
- Set S: $\log|S| + \log\binom{n}{s}$
- Accepted connections: $\sum_{v \in S} 2\log \ell_v + \log\binom{\ell_v}{d}$
- Accepted connections from S to $V - S$: $\sum_{v\in S} 2\log(\epsilon_v d) + \log\binom{d}{\epsilon_v d}$

  ϵ_v: fraction of v's accepted connections towards $V - S$

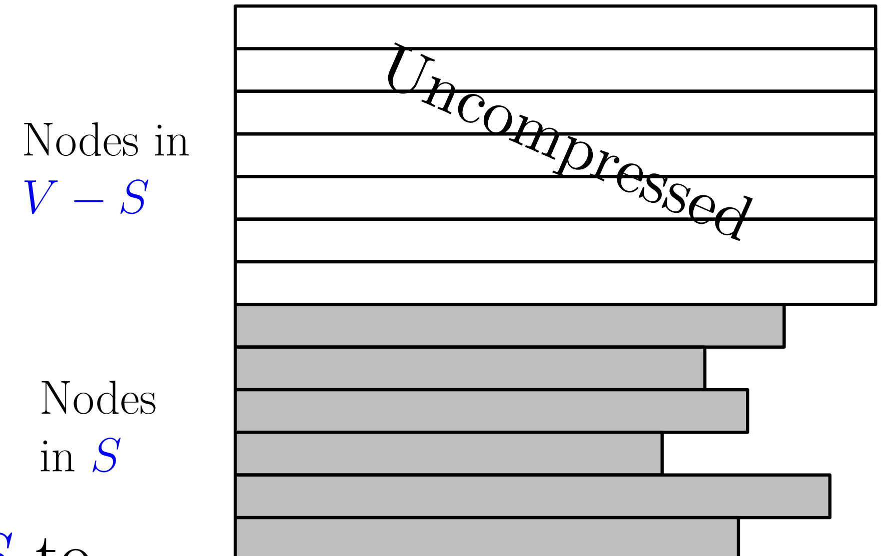
- Destinations of connections from S:
  $\sum_{v \in S}(1-\epsilon_v) d \log((1-\delta_v)\Delta) + \sum_{v\in S} \epsilon_v d \log \Delta$

  connections to S — connections to $V - S$ (uncompressed)

  δ_v: fraction of v's edges towards $V - S$ in G

Compressing the Non-Expanding Set

Encoding:



- Randomness of $V - S$
- Set S: $\log|S| + \log\binom{n}{s}$
- Accepted connections: $\sum_{v \in S} 2\log \ell_v + \log\binom{\ell_v}{d}$
- Accepted connections from S to $V - S$: $\sum_{v \in S} 2\log(\epsilon_v d) + \log\binom{d}{\epsilon_v d}$

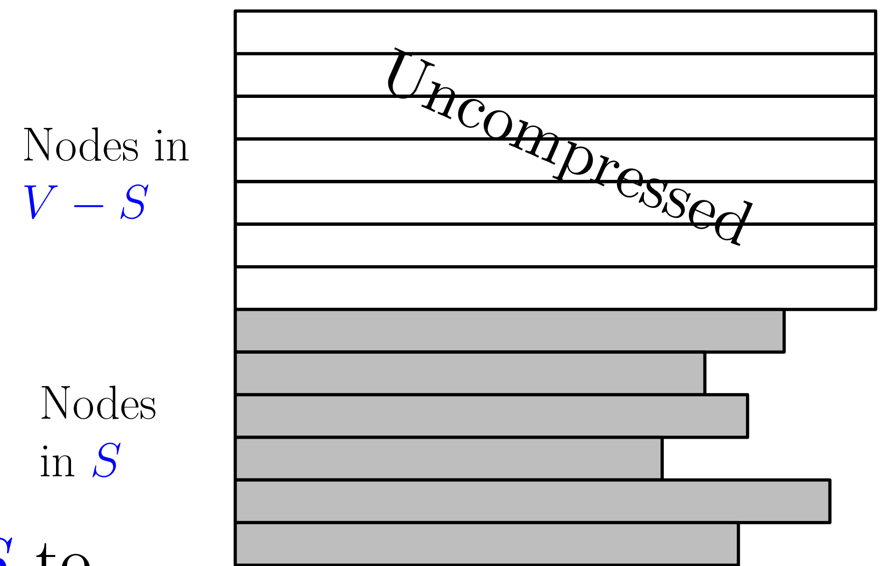
  ϵ_v: fraction of v's accepted connections towards $V - S$
- Destinations of connections from S:
  $\sum_{v \in S}(1 - \epsilon_v)d\log((1-\delta_v)\Delta) + \sum_{v \in S}\epsilon_v d \log\Delta$

  connections to S — connections to $V - S$ (uncompressed)
- **Rejected requests**

δ_v: fraction of v's edges towards $V - S$ in G

Compressing the Non-Expanding Set

Encoding:



- Randomness of $V - S$
- Set S: $\log|S| + \log\binom{n}{s}$
- Accepted connections:
  $\sum_{v \in S} 2\log \ell_v + \log\binom{\ell_v}{d}$
- Accepted connections from S to $V - S$: $\sum_{v \in S} 2\log(\epsilon_v d) + \log\binom{d}{\epsilon_v d}$

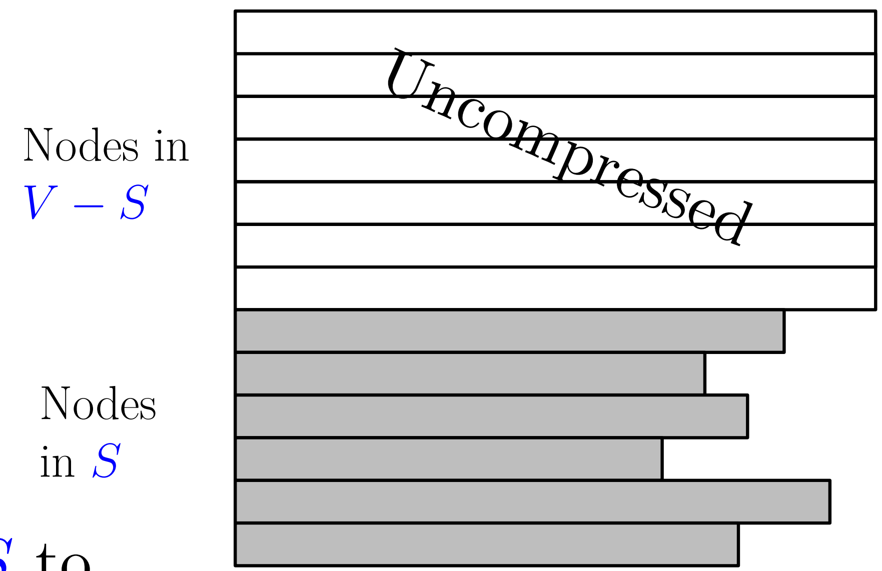
  ϵ_v: fraction of v's accepted connections towards $V - S$
- Destinations of connections from S:
  $\sum_{v \in S}(1-\epsilon_v)d\log((1-\delta_v)\Delta) + \sum_{v \in S}\epsilon_v d\log\Delta$

  connections to S — connections to $V - S$ (uncompressed)

  δ_v: fraction of v's edges towards $V - S$ in G
- **Rejected requests**
- Unused randomness
  (after node's termination)

Compressing the Non-Expanding Set

Encoding:



- Randomness of $V - S$
- Set S: $\log |S| + \log \binom{n}{s}$
- Accepted connections: $\sum_{v \in S} 2 \log \ell_v + \log \binom{\ell_v}{d}$
- Accepted connections from S to $V - S$: $\sum_{v \in S} 2 \log(\epsilon_v d) + \log \binom{d}{\epsilon_v d}$

  ϵ_v: fraction of v's accepted connections towards $V - S$
- Destinations of connections from S:
  $\sum_{v \in S} (1 - \epsilon_v) d \log((1 - \delta_v)\Delta) + \sum_{v \in S} \epsilon_v d \log \Delta$

  connections to S — connections to $V - S$ (uncompressed)
- **Rejected requests**
- Unused randomness (after node's termination)

δ_v: fraction of v's edges towards $V - S$ in G

Compressing Accepted Connections I

To represent accepted requests from S we need

$$\sum_{v\in S}(1-\epsilon_v)d\log((1-\delta_v)\Delta)+\sum_{v\in S}\epsilon_v d\log\Delta$$

$$\leq sd\log\Delta-\frac{1-\epsilon}{2}sd\log\frac{n}{s}+2\epsilon ds$$

where $\epsilon=\frac{1}{s}\sum_{v\in S}\epsilon_v$

Compressing Accepted Connections I

To represent accepted requests from S we need

$$\sum_{v \in S} (1 - \epsilon_v) d \log((1 - \delta_v)\Delta) + \sum_{v \in S} \epsilon_v d \log \Delta$$

$$\leq sd \log \Delta - \frac{1 - \epsilon}{2} sd \log \frac{n}{s} + 2\epsilon ds$$

where $\epsilon = \frac{1}{s} \sum_{v \in S} \epsilon_v$

With simple calculations

$sd \log \Delta - (\sum_{v \in S} (1 - \epsilon_v) d \log((1 - \delta_v)\Delta) + \sum_{v \in S} \epsilon_v d \log \Delta)$
$\geq d \sum_{v \in S} (1 - \epsilon_v) \log \frac{1}{1 - \delta_v}$

Compressing Accepted Connections I

To represent accepted requests from S we need

$$\sum_{v\in S}(1-\epsilon_v)d\log((1-\delta_v)\Delta) + \sum_{v\in S}\epsilon_v d\log\Delta$$

$$\leq sd\log\Delta - \frac{1-\epsilon}{2}sd\log\frac{n}{s} + 2\epsilon ds$$

where $\epsilon = \frac{1}{s}\sum_{v\in S}\epsilon_v$

With simple calculations

$$sd\log\Delta - \left(\sum_{v\in S}(1-\epsilon_v)d\log((1-\delta_v)\Delta) + \sum_{v\in S}\epsilon_v d\log\Delta\right)$$
$$\geq d\sum_{v\in S}(1-\epsilon_v)\log\frac{1}{1-\delta_v}$$

Two cases: $s < \alpha\Delta$ and $\alpha\Delta \leq s \leq \frac{n}{2}$...

Compressing Accepted Connections II

Goal: bound $d \sum_{v \in S} (1 - \epsilon_v) \log \frac{1}{1-\delta_v}$

Case $s < \alpha\Delta$

Use $\Delta(1 - \delta_v) \leq s$ and $(\frac{\Delta}{s})^2 > \frac{\Delta}{s}\frac{1}{\alpha} = \frac{\Delta}{s}\frac{n}{\Delta} = \frac{n}{s}$

hence $d \sum_{v \in S} (1 - \epsilon_v) \log \frac{1}{1-\delta_v} > \frac{1-\epsilon}{2} sd \log \frac{n}{s}$

Compressing Accepted Connections II

Goal: bound $d\sum_{v\in S}(1-\epsilon_v)\log\frac{1}{1-\delta_v}$

Case $s < \alpha\Delta$

Use $\Delta(1-\delta_v) \le s$ and $(\frac{\Delta}{s})^2 > \frac{\Delta}{s}\frac{1}{\alpha} = \frac{\Delta}{s}\frac{n}{\Delta} = \frac{n}{s}$

hence $d\sum_{v\in S}(1-\epsilon_v)\log\frac{1}{1-\delta_v} > \frac{1-\epsilon}{2}sd\log\frac{n}{s}$

Case $\alpha\Delta \le s \le \frac{n}{2}$

Rewrite $-(1-\epsilon)sd\sum_{v\in S}\frac{1-\epsilon_v}{(1-\epsilon)s}\log\frac{1}{1-\delta_v}$

use Jensen's inequality to get $(1-\epsilon)sd\log\frac{1-\epsilon}{1-\delta}$

Compressing Accepted Connections II

Goal: bound $d\sum_{v\in S}(1-\epsilon_v)\log\frac{1}{1-\delta_v}$

Case $s < \alpha\Delta$

Use $\Delta(1-\delta_v) \le s$ and $(\frac{\Delta}{s})^2 > \frac{\Delta}{s}\frac{1}{\alpha} = \frac{\Delta}{s}\frac{n}{\Delta} = \frac{n}{s}$

hence $d\sum_{v\in S}(1-\epsilon_v)\log\frac{1}{1-\delta_v} > \frac{1-\epsilon}{2}sd\log\frac{n}{s}$

Case $\alpha\Delta \le s \le \frac{n}{2}$

Rewrite $-(1-\epsilon)sd\sum_{v\in S}\frac{1-\epsilon_v}{(1-\epsilon)s}\log\frac{1}{1-\delta_v}$

use Jensen's inequality to get $(1-\epsilon)sd\log\frac{1-\epsilon}{1-\delta}$

To bound $1-\delta$ we use the **Expander Mixing Lemma**:

$$(1-\delta) \le \frac{s}{n} + \lambda$$

Compressing Accepted Connections II

Goal: bound $d\sum_{v\in S}(1-\epsilon_v)\log\frac{1}{1-\delta_v}$

Case $s < \alpha\Delta$

Use $\Delta(1-\delta_v) \le s$ and $(\frac{\Delta}{s})^2 > \frac{\Delta}{s}\frac{1}{\alpha} = \frac{\Delta}{s}\frac{n}{\Delta} = \frac{n}{s}$

hence $d\sum_{v\in S}(1-\epsilon_v)\log\frac{1}{1-\delta_v} > \frac{1-\epsilon}{2}sd\log\frac{n}{s}$

Case $\alpha\Delta \le s \le \frac{n}{2}$

Rewrite $-(1-\epsilon)sd\sum_{v\in S}\frac{1-\epsilon_v}{(1-\epsilon)s}\log\frac{1}{1-\delta_v}$

use Jensen's inequality to get $(1-\epsilon)sd\log\frac{1-\epsilon}{1-\delta}$

To bound $1-\delta$ we use the **Expander Mixing Lemma**:

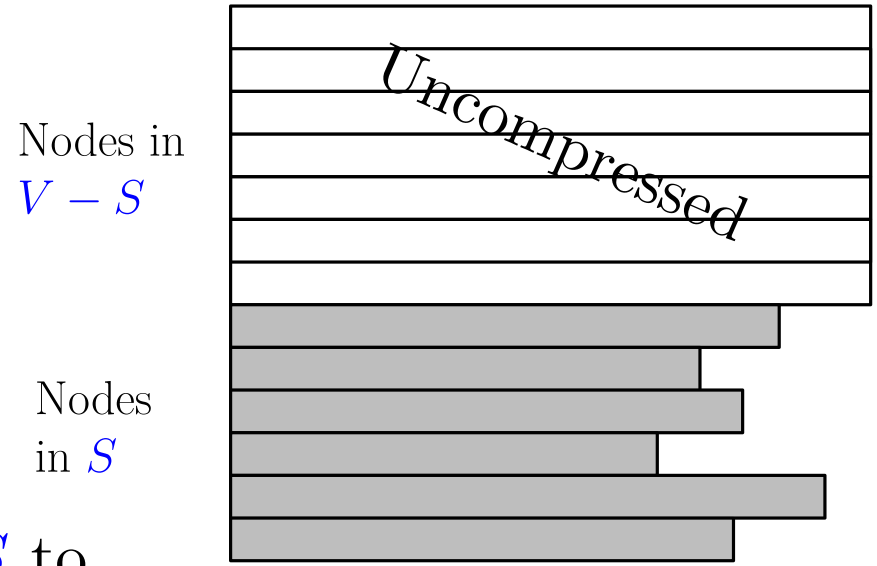
$$(1-\delta) \le \frac{s}{n} + \lambda$$

together with hypothesis on s and λ, it implies

$$(1-\epsilon)sd\log\frac{1-\epsilon}{1-\delta} > (1-\epsilon)sd\log\frac{n}{s} - 2\epsilon ds$$

Compressing the Non-Expanding Set

Encoding:



- Randomness of $V - S$
- Set S: $\log |S| + \log \binom{n}{s}$
- Accepted connections: $\sum_{v \in S} 2 \log \ell_v + \log \binom{\ell_v}{d}$
- Accepted connections from S to $V - S$: $\sum_{v \in S} 2 \log(\epsilon_v d) + \log \binom{d}{\epsilon_v d}$

  ϵ_v: fraction of v's accepted connections towards $V - S$
- Destinations of connections from S: $\underbrace{\sum_{v \in S} (1 - \epsilon_v) d \log((1 - \delta_v)\Delta)}_{\text{connections to } S} + \underbrace{\sum_{v \in S} \epsilon_v d \log \Delta}_{\text{connections to } V - S \text{ (uncompressed)}}$
- **Rejected requests**
- Unused randomness (after node's termination)

Compressing Rejected Requests (Idea)

With $\ell_v - d'$ bits we encode which requests are rejected.
The hard part is compressing their *destinations*, for which we use the following notions:

Semi-saturated nodes ss_t: accepted connections until time $t - 1 +$ requests from $V - S$ are $> \frac{dc}{2}$
Critical nodes c_t: not semi-saturated at time t but accepted $+$ rejected connections are $> cd$

Compressing Rejected Requests (Idea)

With $\ell_v - d'$ bits we encode which requests are rejected. The hard part is compressing their *destinations*, for which we use the following notions:

Semi-saturated nodes ss_t: accepted connections until time $t - 1 +$ requests from $V - S$ are $> \frac{dc}{2}$
Critical nodes c_t: not semi-saturated at time t but accepted $+$ rejected connections are $> cd$

Claim. semi-saturated nodes $\leq \frac{n}{2n}$ and critical nodes $\leq \frac{n}{c}$.

Compressing Rejected Requests (Idea)

With $\ell_v - d'$ bits we encode which requests are rejected. The hard part is compressing their *destinations*, for which we use the following notions:

Semi-saturated nodes ss_t: accepted connections until time $t-1+$ requests from $V-S$ are $> \frac{dc}{2}$

Critical nodes c_t: not semi-saturated at time t but accepted $+$ rejected connections are $> cd$

Claim. semi-saturated nodes $\leq \frac{n}{2n}$ and critical nodes $\leq \frac{n}{c}$.

We can then write

$$ss(v) \log \frac{2n}{c} + \sum_1^T rc_t(v) \log c_t$$

Where $rss(v)$ is the number of rejected connections from v to semisaturated nodes and $rc_t(v)$ is the number of rejected connections from v to critical nodes at time t

Compression Summary

Set S

Size	Index of the set

$2\log|S| + \log\binom{n}{|S|}$

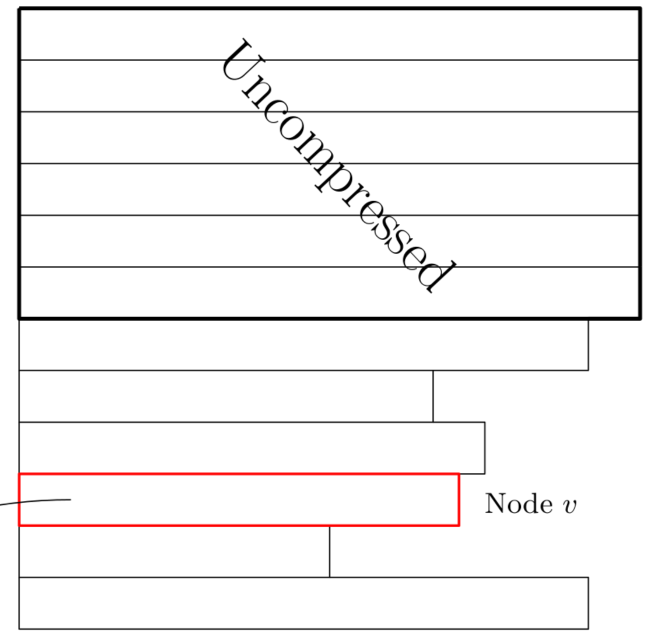
Critical Nodes

Sizes	Indices of sets

$\sum_{t=1}^{T}\left[\log c_t + \log\binom{n}{c_t}\right]$

Nodes in $V \setminus S$

Nodes in S



Subset of accepted requests	Subset of accepted requests in S	Destinations of **accepted requests** ouside S (uncompressed) + + inside S (compressed)	Destinations of **rejected requests**
$2\log\ell_v + \log\binom{\ell_v}{d}$	$2\log(\varepsilon_v d) + \log\binom{d}{\varepsilon_v d}$	$\varepsilon_v d\log\Delta +$ $+(1-\varepsilon_v)d\log((1-\delta)\Delta)$	

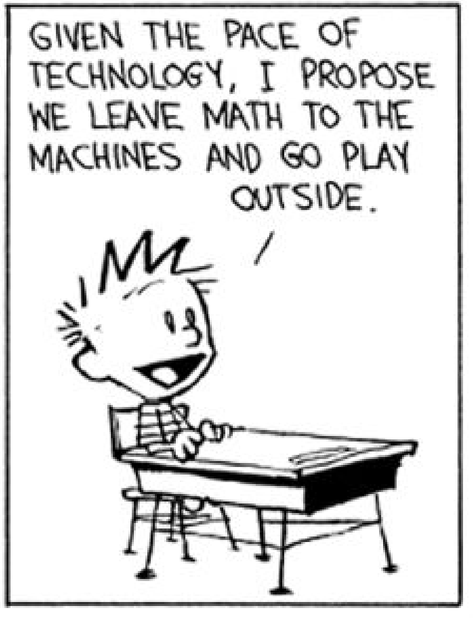
Semi-satured / Critical	S.-sat. dest.	Crit. dest.	Crit. dest.	S.-sat. dest.	S.-sat. dest.	Crit. dest.
$\ell_v - d$	$\log(n/c)$	$\log c_{t_1}$	$\log c_{t_2}$	$\log(n/c)$	$\log(n/c)$	$\log c_{t_k}$

Open Problems

- Generalizing to non-dense expanders.
  E.g., not clear if all nodes can achieve d connections if $\Delta = o(n)$ (if $\Delta = O(\log n)$, this happens w.h.p.).

- Extending analysis to non-regular graphs.

- Investigate robustness of RAES when nodes join or leave the network.



Thank You!