Natural Distributed Algorithms

- Lecture -

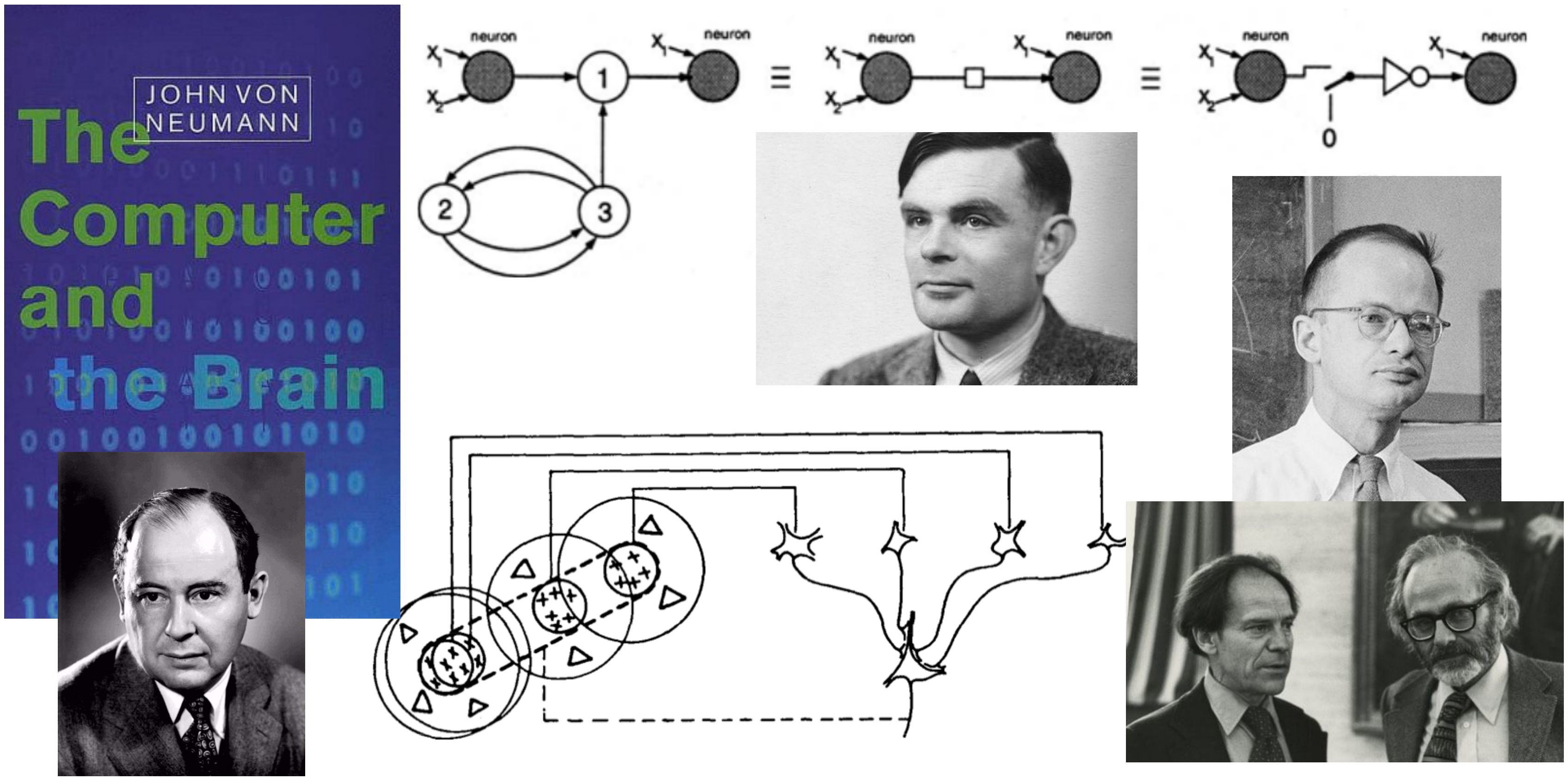
Simple Distributed Graph Sparsification as an Inquiry towards Neural Pruning

Emanuele Natale
CNRS - UCA

CdL in Informatica
Università degli Studi di Roma
"Tor Vergata"

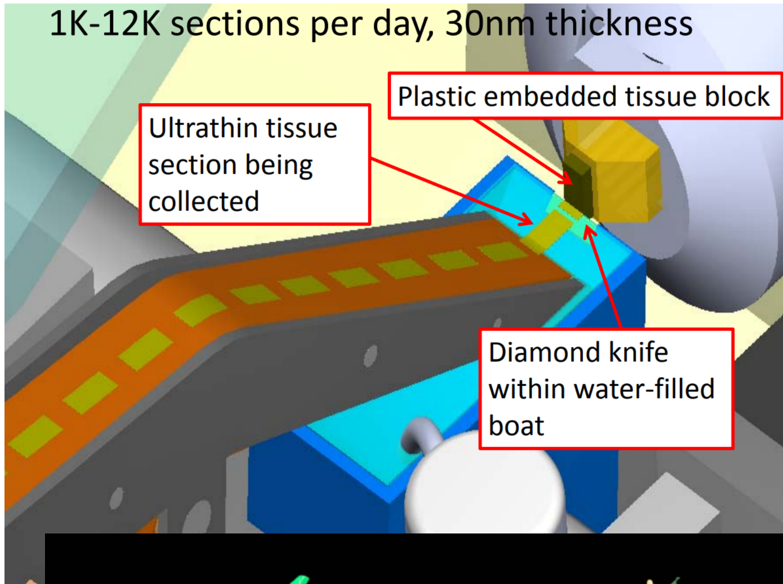
The Brain and Computation

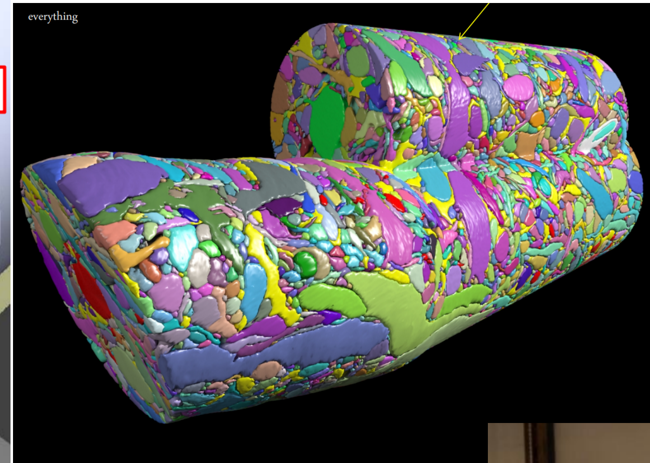
Von Neumann, Turing, McCulloch, Pitts, Barlow... were interested in the other field to better understand theirs.



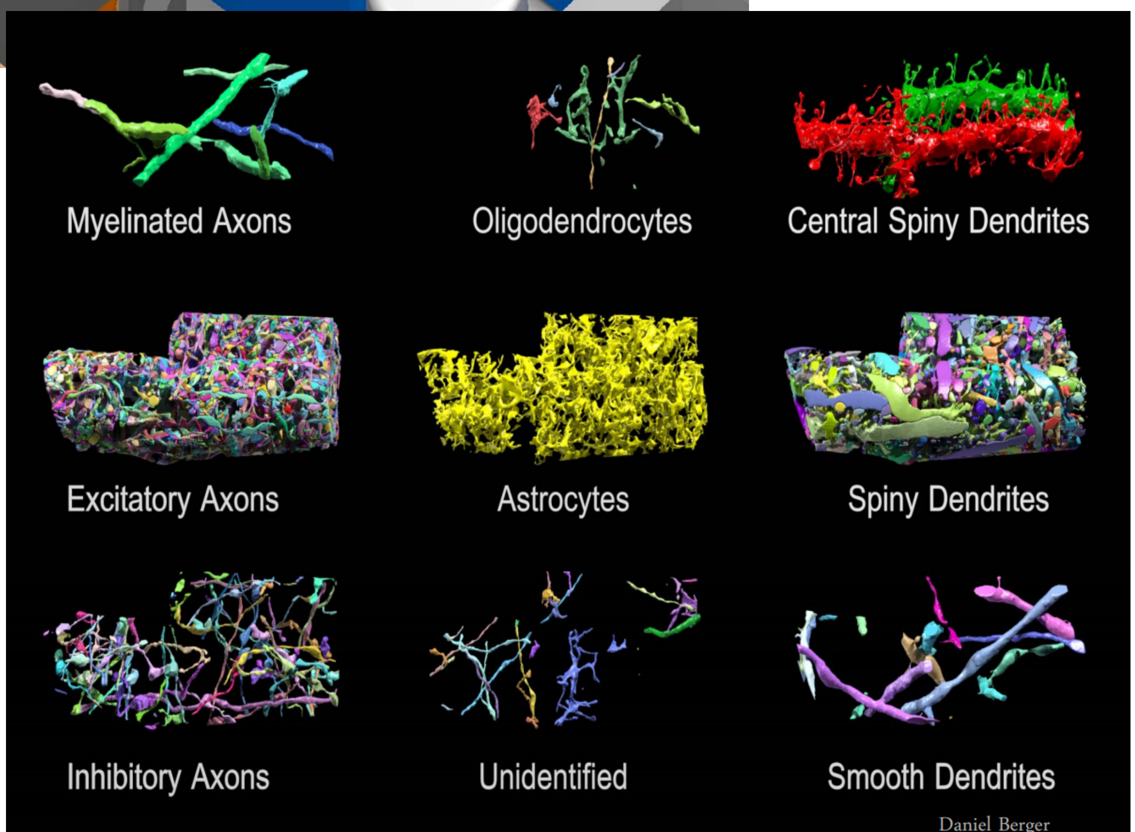
Both fields have exploded in knowledge but have also grown further apart.

Computational Neuroscience: Data





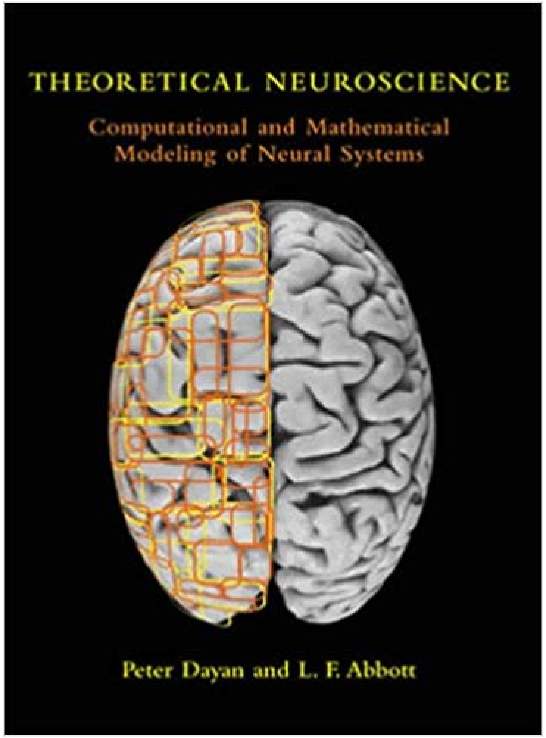
$1\ \text{mm}^3$ of mouse brain $\Longrightarrow$ 300 TB of image data



Computational Neuroscience: Theory

Issues:

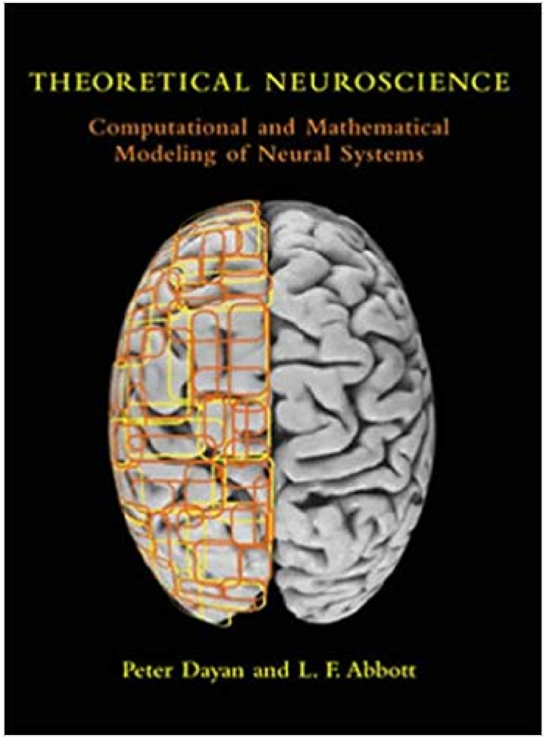
- Far from experimentalists



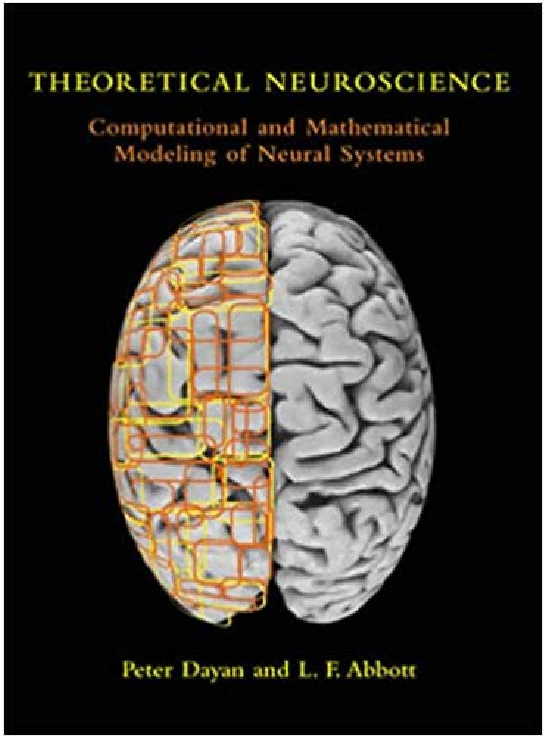
Computational Neuroscience: Theory

Issues:

- Far from experimentalists
- Internally divided



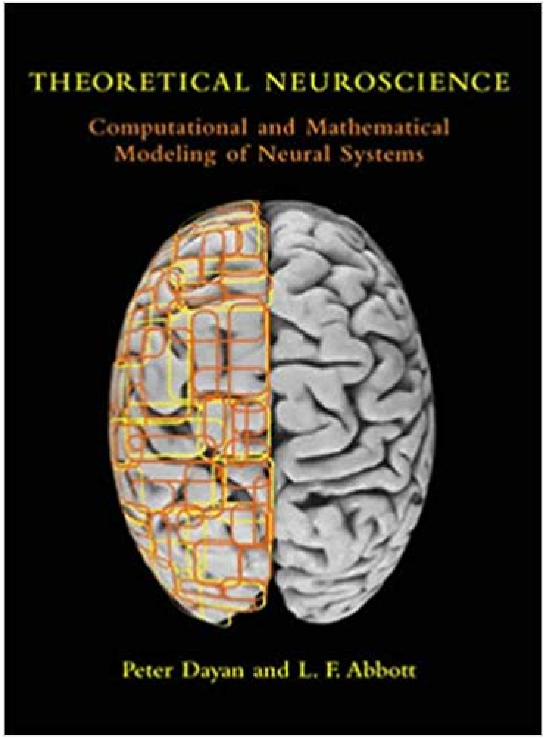
Computational Neuroscience: Theory



Issues:

- Far from experimentalists
- Internally divided
- Led mostly by physicists

Computational Neuroscience: Theory



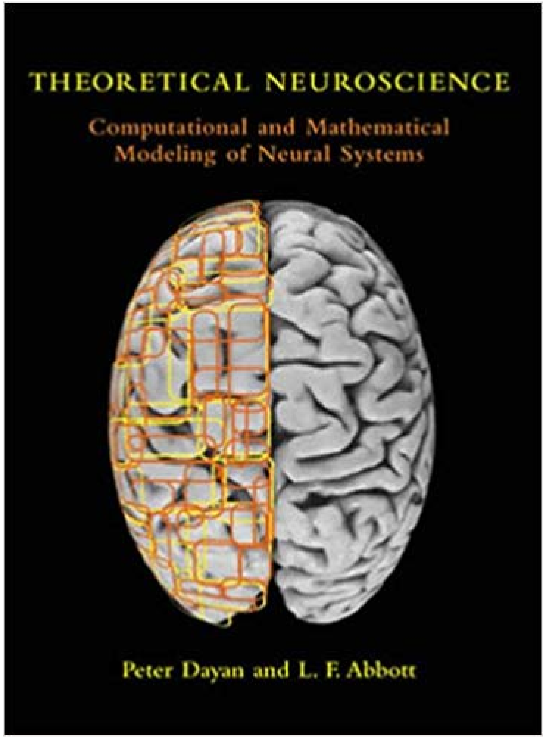
Issues:

- Far from experimentalists
- Internally divided
- Led mostly by physicists

Theories:

- Neural networks for learning: Pitts & McCulloch ('47), Rosenblatt ('58), Hubel & Wiesel ('62), ...

Computational Neuroscience: Theory



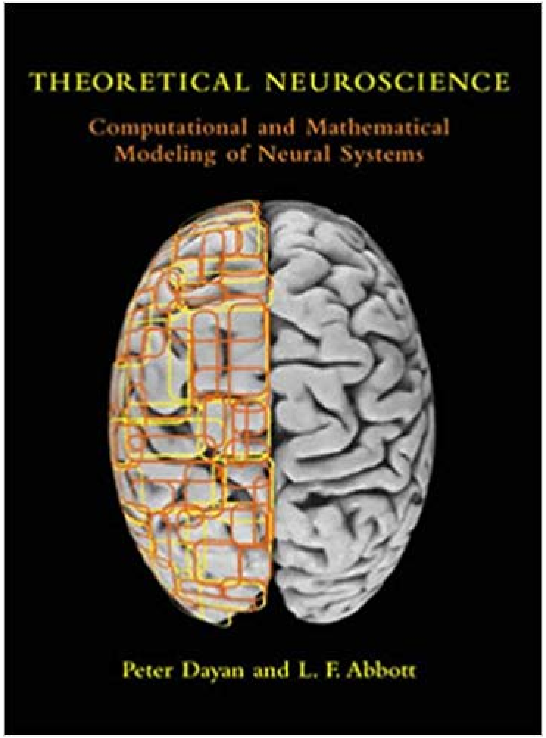
Issues:

- Far from experimentalists
- Internally divided
- Led mostly by physicists

Theories:

- Neural networks for learning: Pitts & McCulloch ('47), Rosenblatt ('58), Hubel & Wiesel ('62), ...
- Neural-dynamics model for specific neural phenomena (associative memory, grid cells, place cells, oscillations, ...)

Computational Neuroscience: Theory



Issues:

- Far from experimentalists
- Internally divided
- Led mostly by physicists

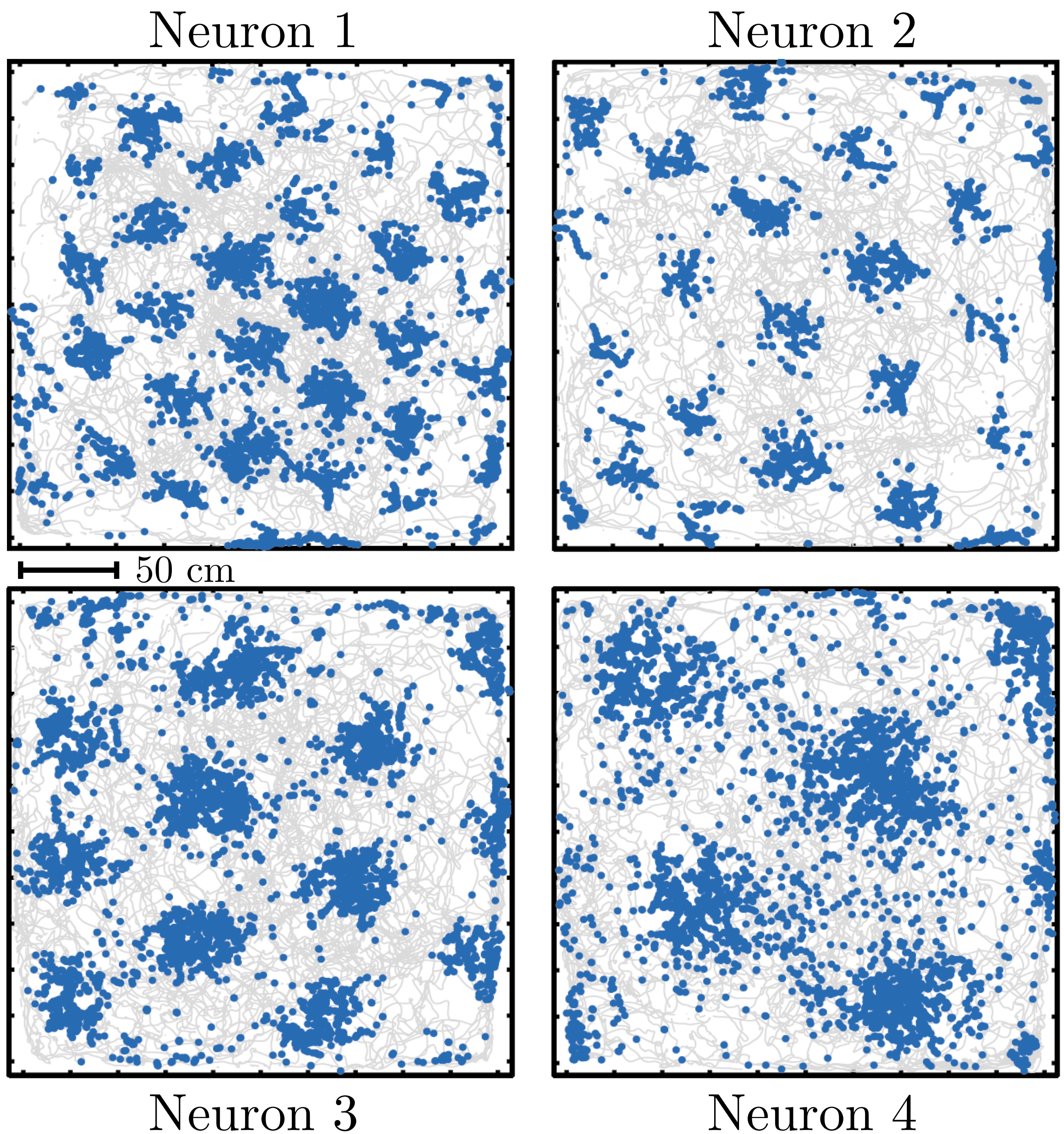
Theories:

- Neural networks for learning: Pitts & McCulloch ('47), Rosenblatt ('58), Hubel & Wiesel ('62), ...
- Neural-dynamics model for specific neural phenomena (associative memory, grid cells, place cells, oscillations, ...)
- Works from *Theoretical Computer Science*: Neuroidal Model by Valiant ('94), models of associative memory by Papadimitriou et al, ('15), Lynch et al. ('16) and Navlakha et al. ('17), ...

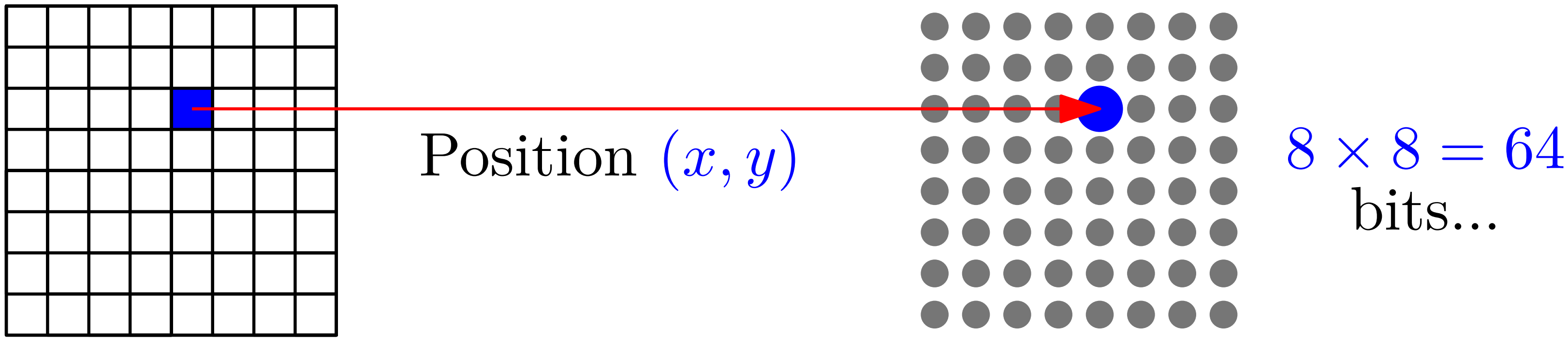
Does the Brain use Algorithms?

How are you aware of your location in space?

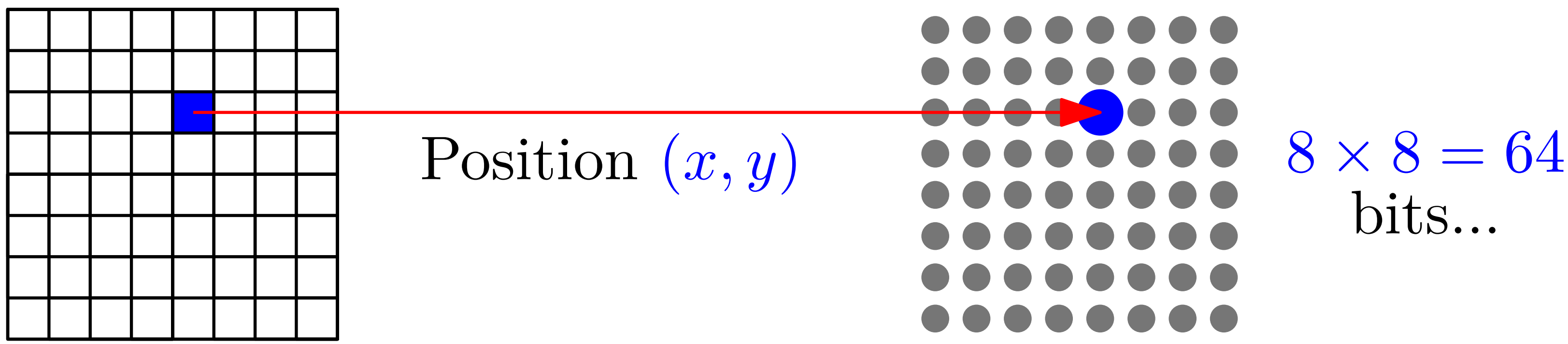
2014 Nobel Prize in Physiology to J. O'Keefe & M. B. and E. Moser for discovery of cells that constitute a positioning system in the brain

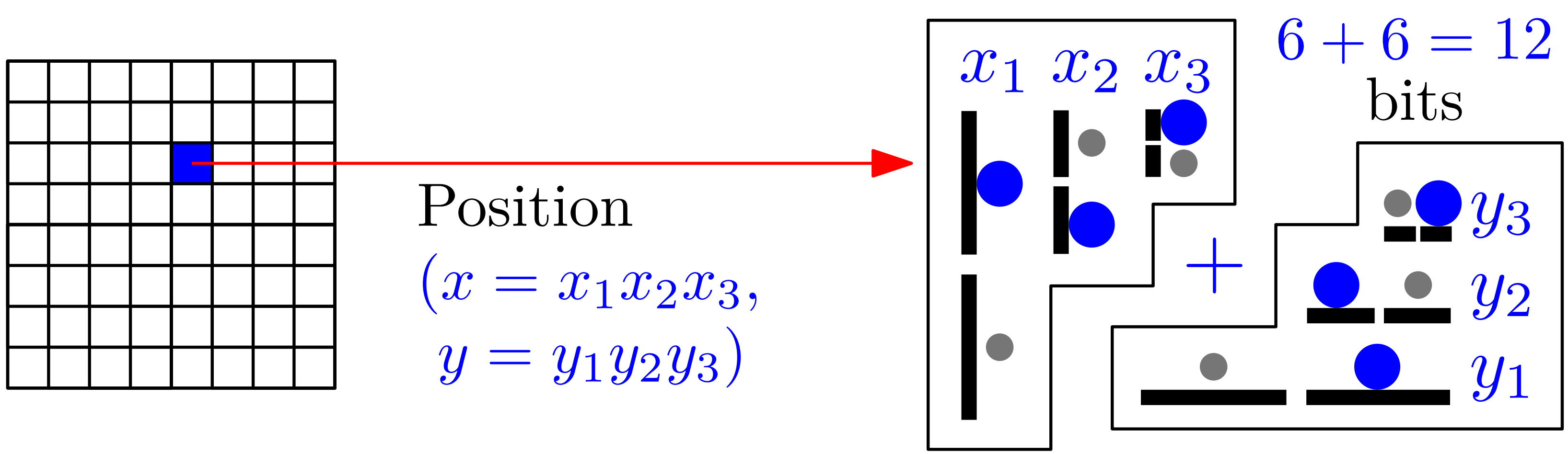


The Principle of Efficiency

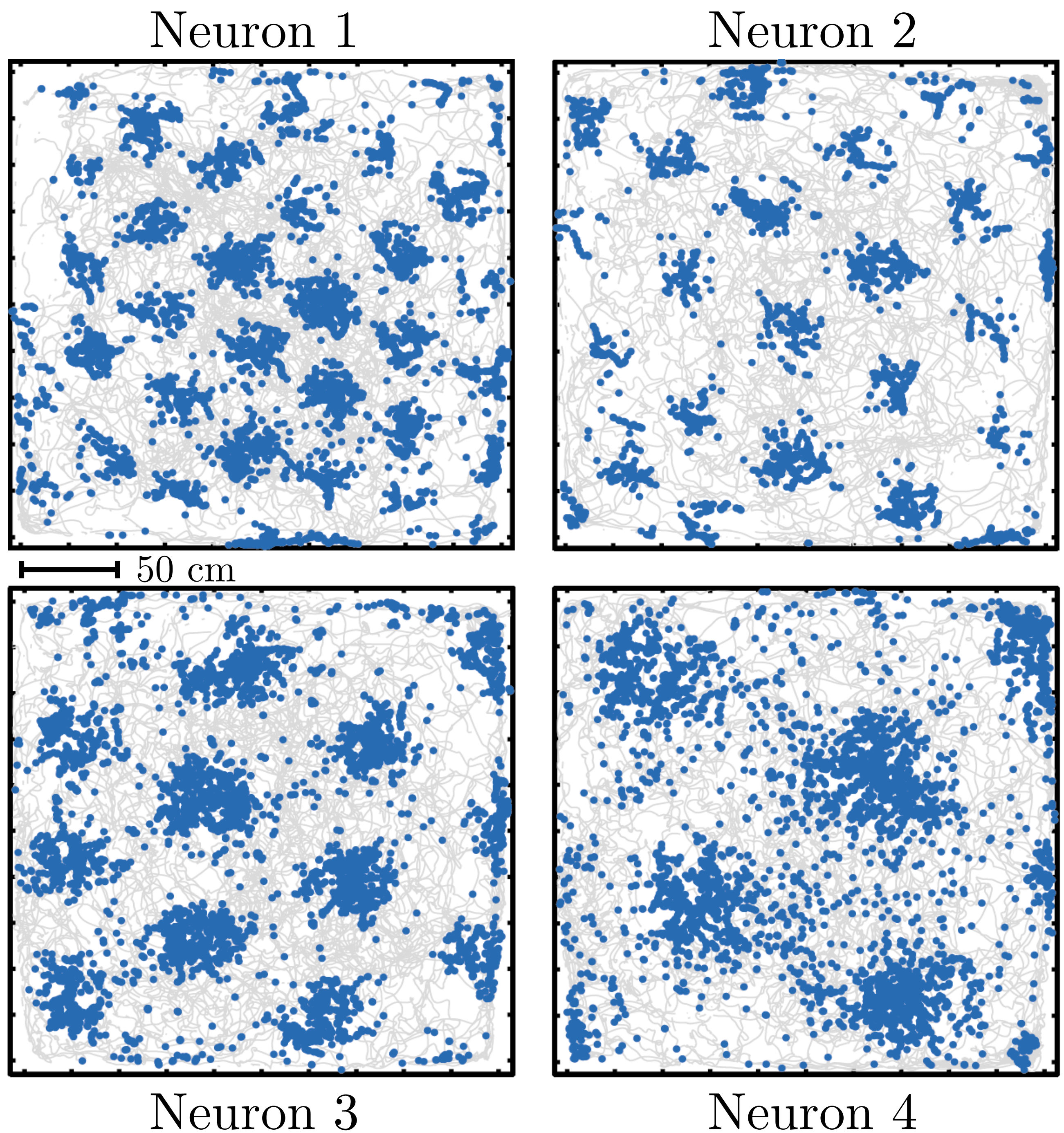


The Principle of Efficiency





Grid Cells Encodes Position Efficiently



Neural Pruning

Neural pruning is a fundamental phenomenon in nervous systems. What are the algorithmic principles that guide it?

Development

Research article 3631

Cellular mechanisms of dendrite pruning in *Drosophila*: insights from in vivo time-lapse of remodeling dendritic arborizing sensory neurons

Darren W. Williams*,† and James W. Truman

WIREs Developmental Biology

A fly's view of neuronal remodeling

Shiri P. Yaniv and Oren Schuldiner*

hhmi | Howard Hughes Medical Institute

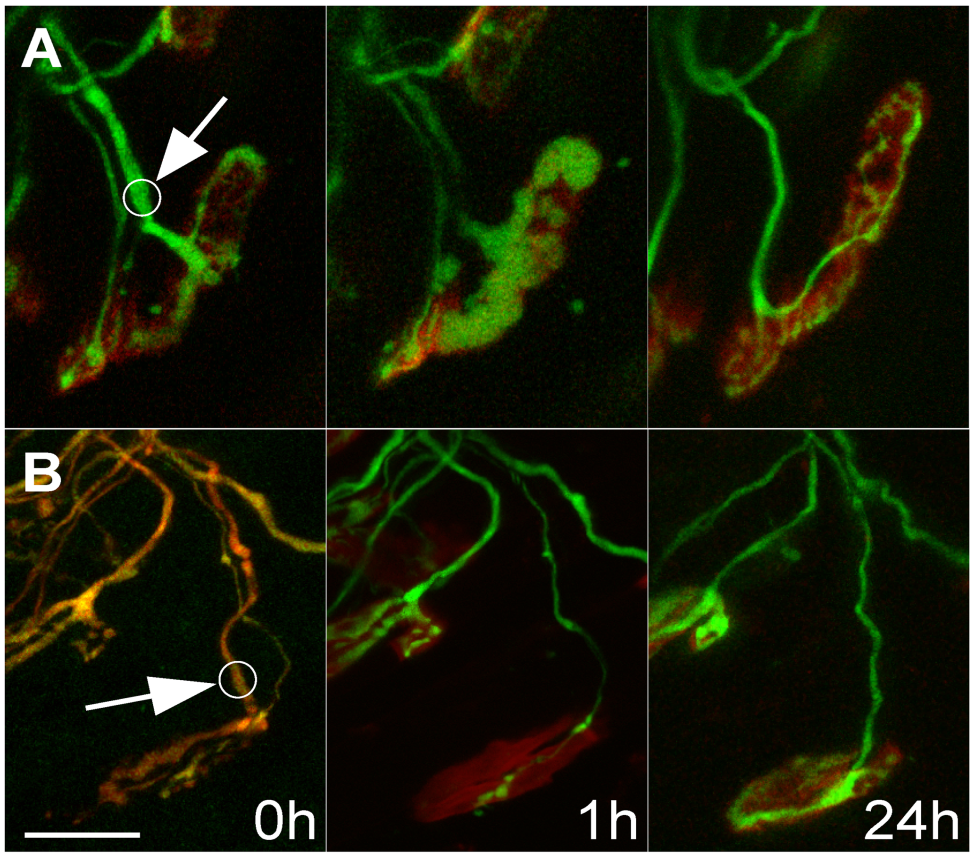
Published as: *Annu Rev Cell Dev Biol*. 2015 November 13; 31: 779–805.

Sculpting Neural Circuits by Axon and Dendrite Pruning

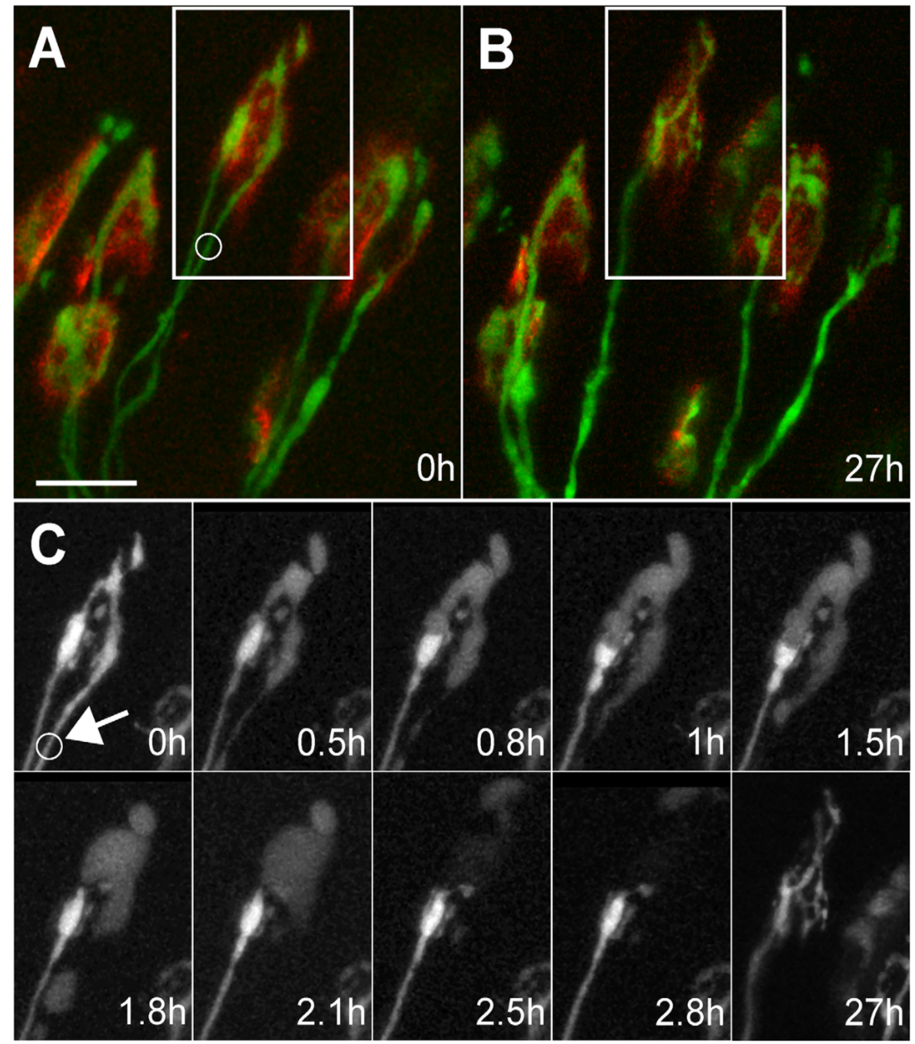
Martin M. Riccomagno[1] and **Alex L. Kolodkin**[2]

Neural Pruning Example: Innervation in Muscular Junctions

[Gan & Lichtman, Science '03; Turney & Lichtman, PLOS Bio. '12; Tapia et al., Neuron '12]:

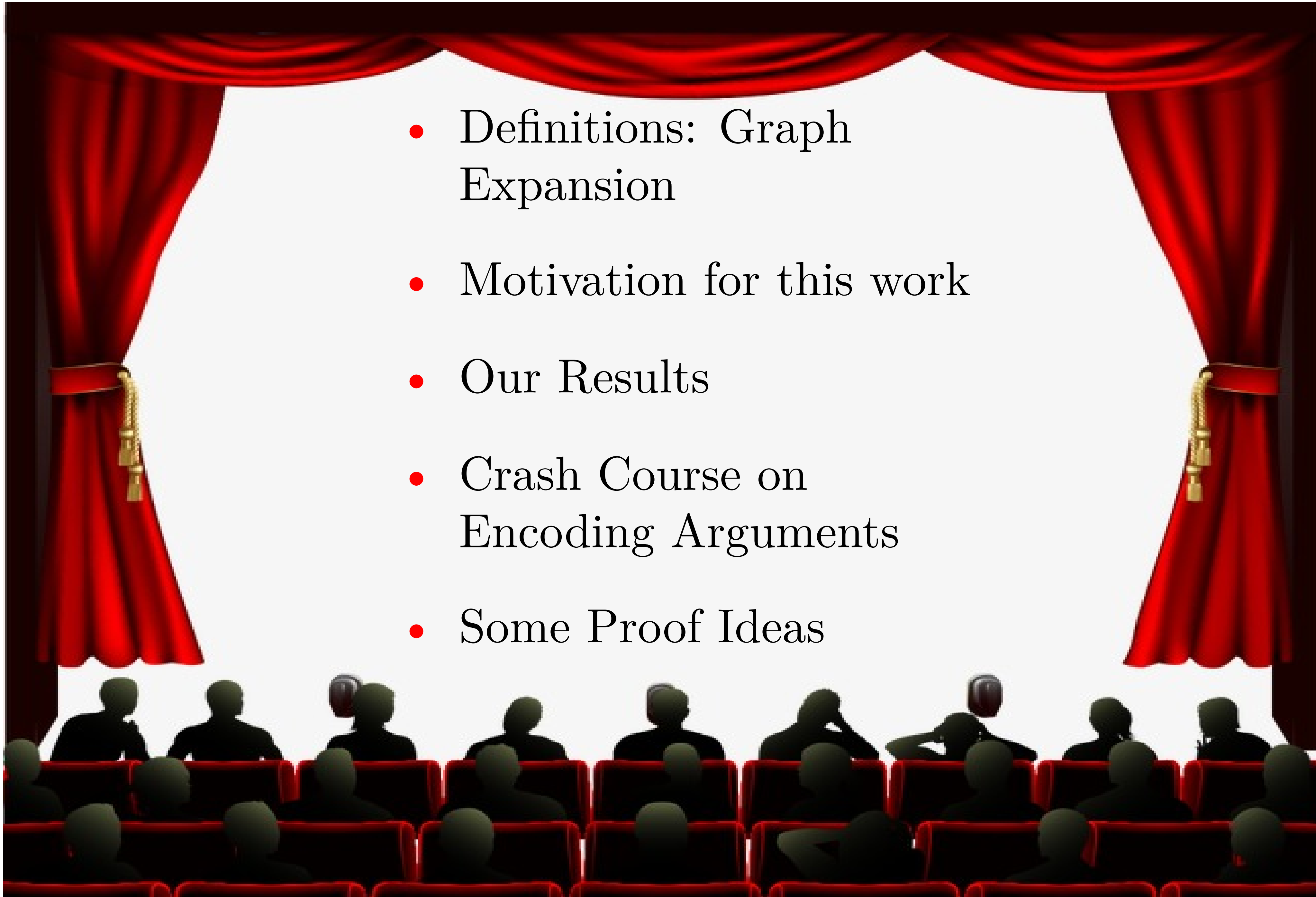


A sparsification process occurs which aims at having at least one axon per innervation site.



Picture from Turney & Lichtman, PLOS Bio. '12

Outline of the rest of the talk



- 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?

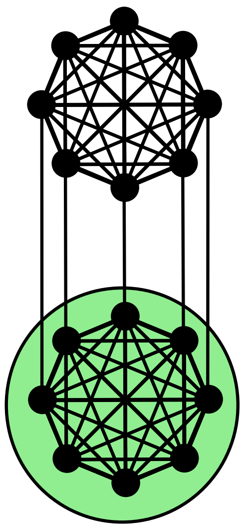
- Attempt 1. Number of edges going out of S: $e(S, V - S) = |\{(u, v) | u \in S, v \in V - S\}|$

Graph Expansion I

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

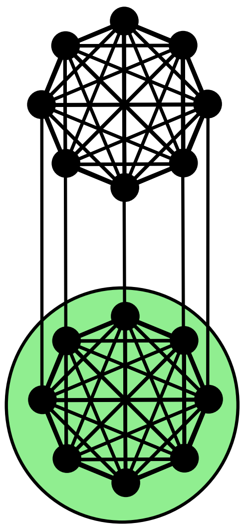
- Attempt 1. Number of edges going out of S: $e(S, V-S) = |\{(u,v) | u \in S, v \in V-S\}|$

  Problem: big sets are better than small ones



Graph Expansion I

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



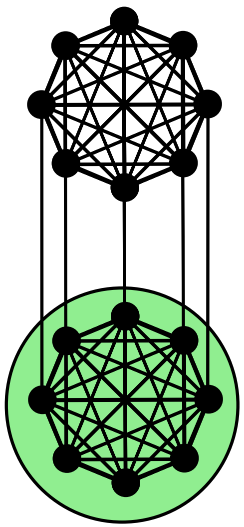
- Attempt 1. Number of edges going out of S: $e(S, V - S) = |\{(u, v)|u \in S, v \in V - S\}|$

  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

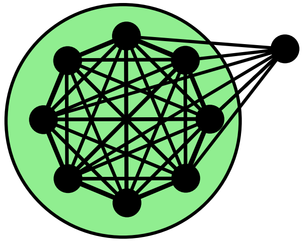
What is a good measure of *connectedness* for a set of nodes S?



- 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

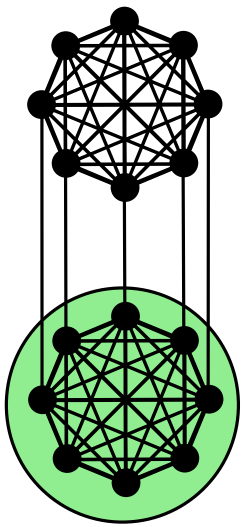
- Attempt 2. We also divide by the sum of its degrees $vol(S) = \sum_{u \in S} d_u$: $\frac{e(S,V-S)}{vol(S)}$



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

Graph Expansion I

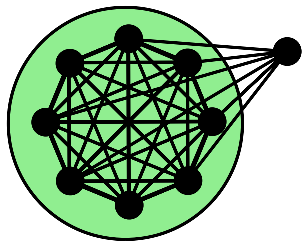
What is a good measure of *connectedness* for a set of nodes S?



- 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 $vol(S)$

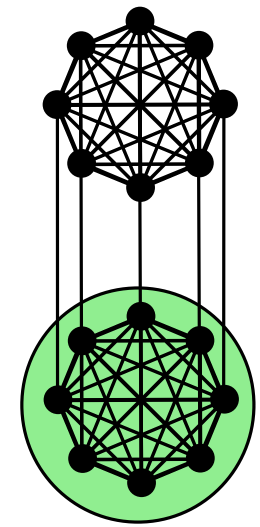
- Attempt 3. We consider the "worst" between S and $V-S$: $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$

Graph Expansion I

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

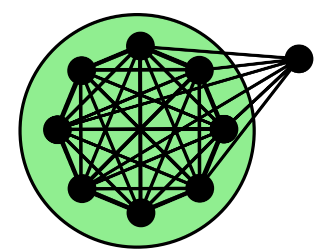
- Attempt 1. Number of edges going out of S: $e(S, V-S) = |\{(u,v) | u \in S, v \in V-S\}|$

  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 $V-S$: $\boxed{\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}}$ → conductance

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

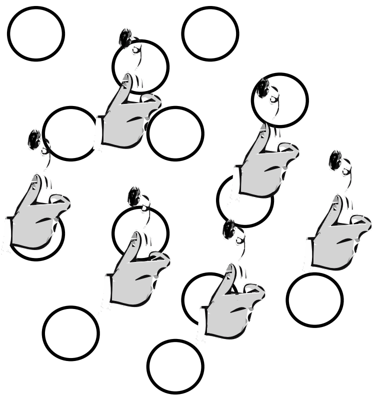
In regular graphs $\frac{e(S,V-S)}{\min\{vol(S),vol(V-S)\}}$ is equivalent to $\phi(S) = \frac{e(S,V-S)}{vol(S)}$ assuming $S \leq \frac{n}{2}$

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 $S \leq \frac{n}{2}$

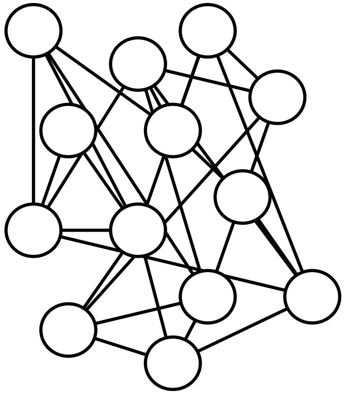
Interpretation. In regular graphs, $\phi(S) = \Pr(\text{random walk on random node of } S \text{ exits it})$

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 :

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

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 :

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

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

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 :

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

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

$$1_S^T A 1_S \qquad 1_S^T\left(\frac{\Delta}{n}J\right)1_S$$

$$2e(S,S) - \frac{\Delta}{n}|S|^2$$

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 :

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

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

$$1_S^T A 1_S \qquad 1_S^T\left(\frac{\Delta}{n}J\right)1_S$$

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

2nd-largest eigenvalue

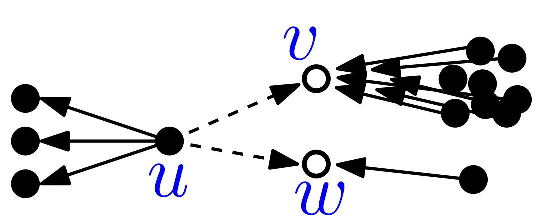
Algorithm **R**equest - **A**ccept if **E**nough **S**pace

Algorithm RAES(G, d, c) for each node v:

- Set $d_{out} = 0$ and assume connections are directed
- At the start of each round,
  if ($d_{out} < d$) then
  send $d - d_{out}$ requests to random neighbors
- At the end of each round
  if (current requests $+$ new ones $\leq cd$) then
  accept all request
  else
  reject all current requests

  if ($d_{out} = d$) then
  forget edge orientation

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.

Mathematical Interest of the Process

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.

Mathematical Interest of the Process

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:
How to find the *low-degree sub-expander*?

Distributed-Computing Interest of the Process

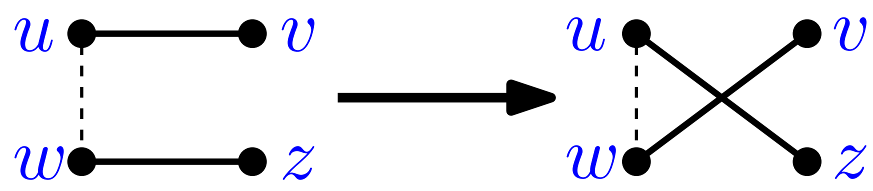
Several works propose complicated distributed construction of expanders:

- Law and Siu [INFOCOM'03]: incremental construction using Hamiltonian cycles

Distributed-Computing Interest of the Process

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



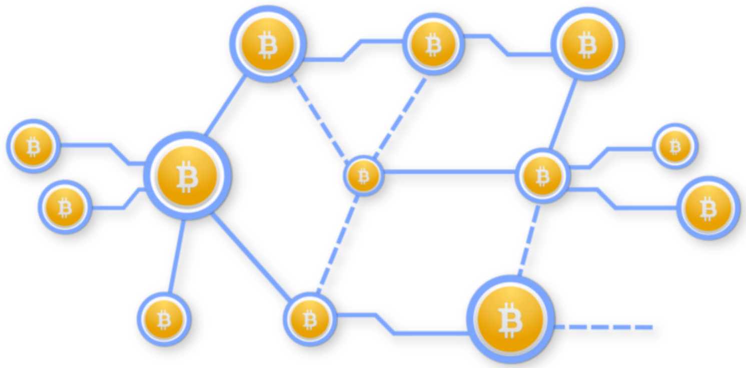
Bonus Motivations from CS

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

Bonus Motivations from CS

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

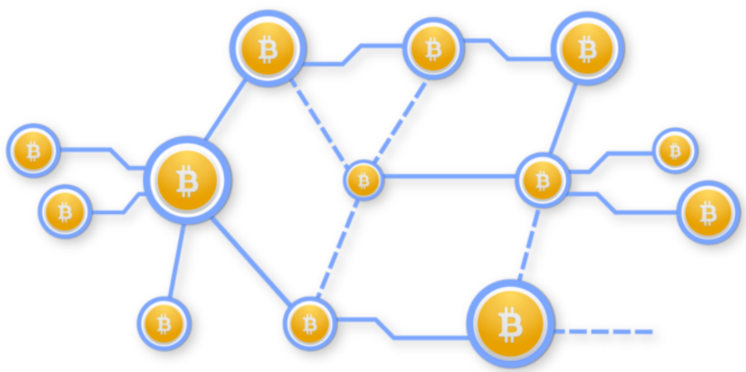
- Model creation of overlay networks in protocols such as BitTorrent (P2P) or Bitcoin (distributed ledgers)



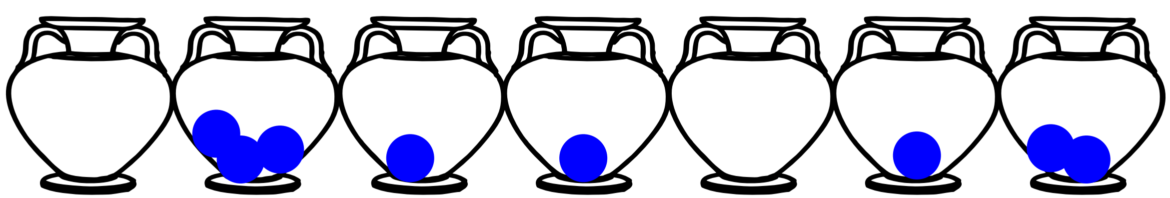
Bonus Motivations from CS

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

- Model creation of overlay networks in protocols such as BitTorrent (P2P) or Bitcoin (distributed ledgers)



- Distributed construction of constant-degree graph implies *constant-load balancing* algorithm.
  Previous works: almost-tight load balancing in poly time (Berenbrink et al., SPAA'14)



The Theorem

Theorem [Becchetti, Clementi, N., Pasquale, Trevisan. 2019. "Finding a Bounded-Degree Expander Inside a Dense One."]
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 $d(c+1)$.

The Theorem

Theorem [Becchetti, Clementi, N., Pasquale, Trevisan. 2019. "Finding a Bounded-Degree Expander Inside a Dense One."]
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 Arguments

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

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

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

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

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:

$$\Big(\underbrace{\text{index } i \text{ of first bit of } B}_{\log n \text{ bits}},\ \underbrace{\text{all other bits of the string except those at entry } i, i+1, \ldots, i+\log n+s}_{n-(\log n+s) \text{ bits}}\Big)$$

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:

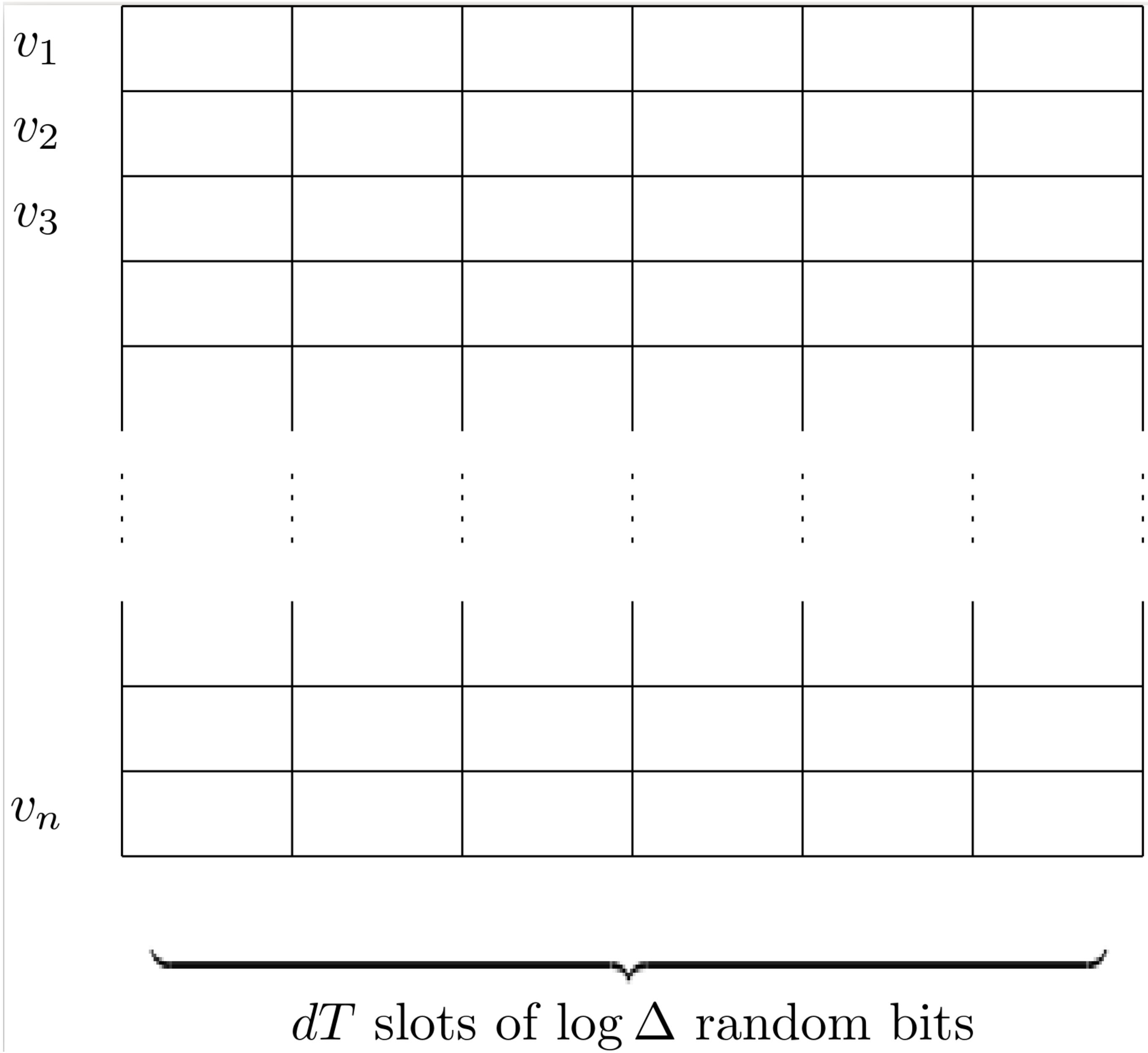
$$\Big(\underbrace{\text{index } i \text{ of first bit of } B}_{\log n \text{ bits}} \;,\; \underbrace{\text{all other bits of the string except those at entry } i, i+1, \ldots, i+\log n+s}_{n-(\log n+s) \text{ bits}} \Big)$$

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

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

factor 2 because
prefix-free encoding

- v's identity: $\log n$
- number of v's requests ℓ_v: $2 \log \ell_v$

Encoding for Always-Rejected v

We encode with the following bits

factor 2 because
prefix-free encoding

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

Encoding for Always-Rejected v

We encode with the following bits

factor 2 because
prefix-free encoding

- v's identity: $\log n$
- number of v's requests 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

factor 2 because
prefix-free encoding

- v's identity: $\log n$
- number of v's requests 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

factor 2 because
prefix-free encoding

- v's identity: $\log n$
- number of v's requests 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

factor 2 because prefix-free encoding

- v's identity: $\log n$
- number of v's requests 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

factor 2 because prefix-free encoding

- v's identity: $\log n$
- number of v's requests 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

factor 2 because prefix-free encoding

- v's identity: $\log n$
- number of v's requests 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: $(\ell_v - d) \log \frac{n}{c}$

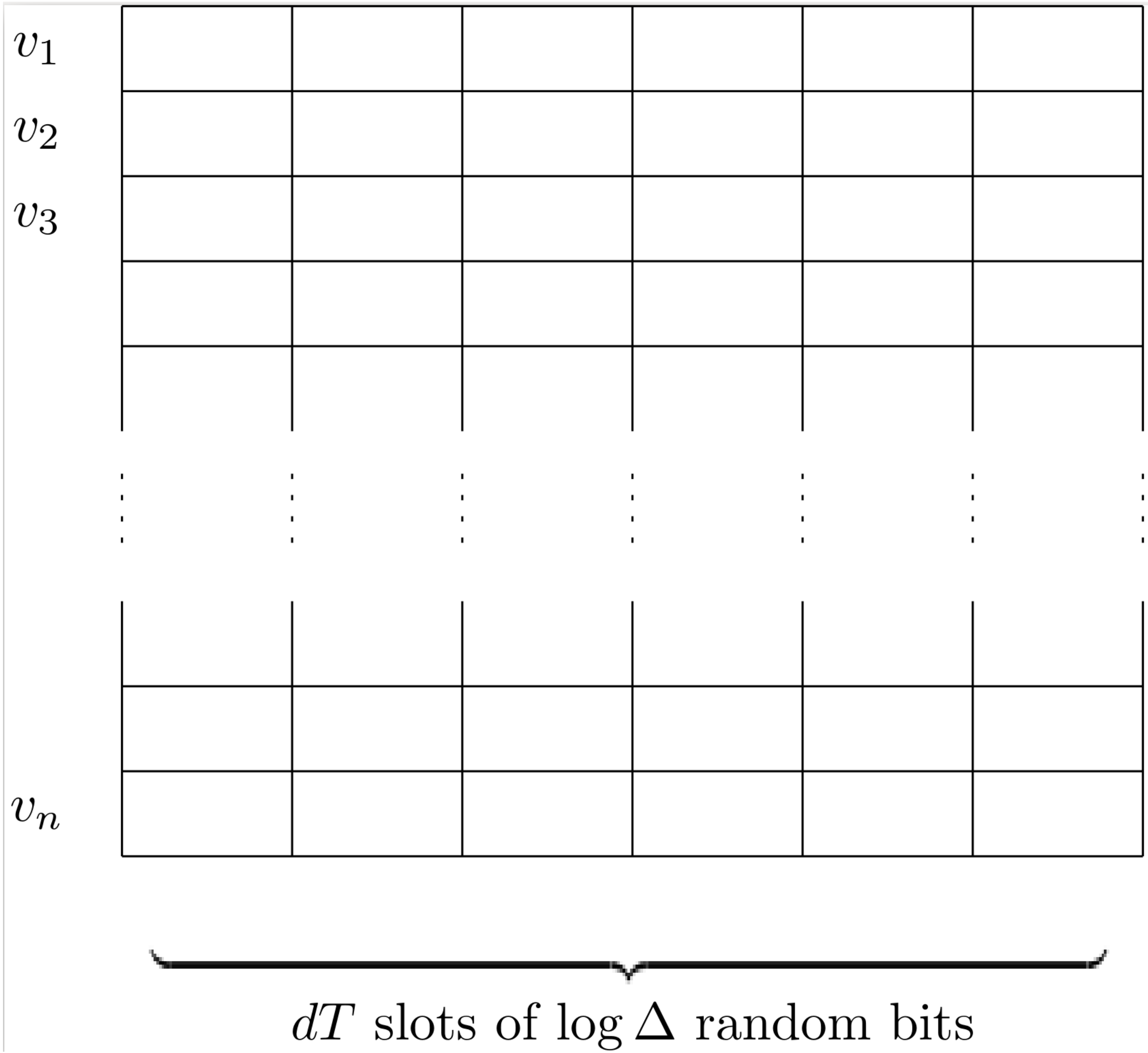
Observation: at each round there are at most $\frac{n}{c}$ rejecting nodes

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

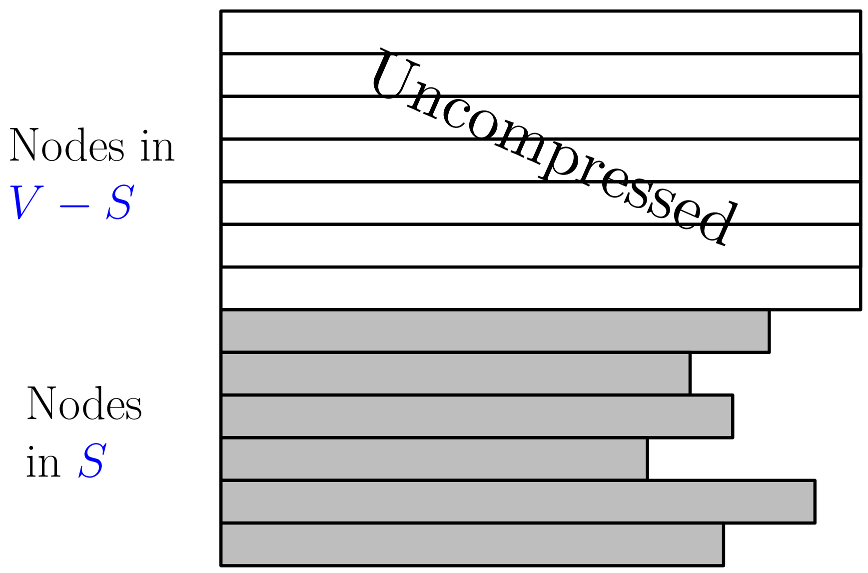
We show that if the execution results in a non-expander, then it can be represented with $ndt \log \Delta - \Omega(\log n)$ bits



Compressing the Non-Expanding Set

Encoding:

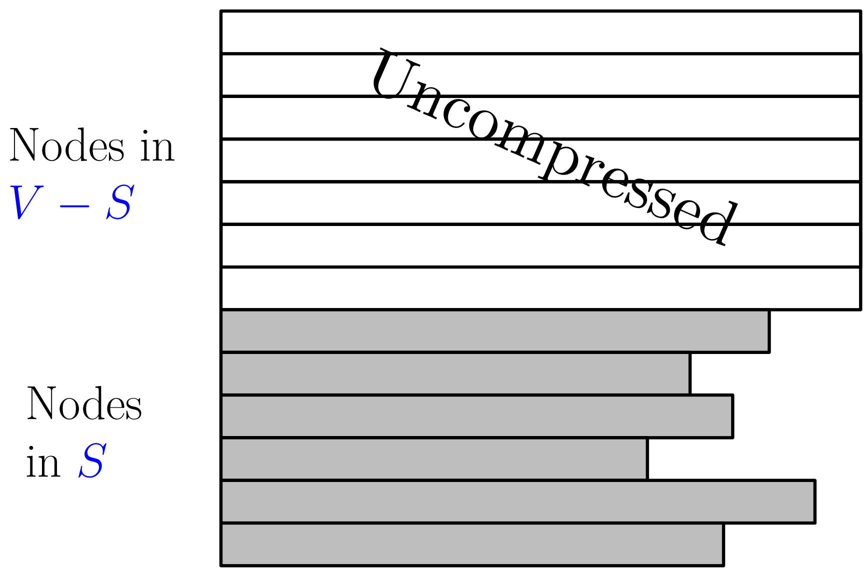
- Randomness of $V - S$
- Set S: $\log|S| + \log\binom{n}{s}$



Compressing the Non-Expanding Set

Encoding:

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

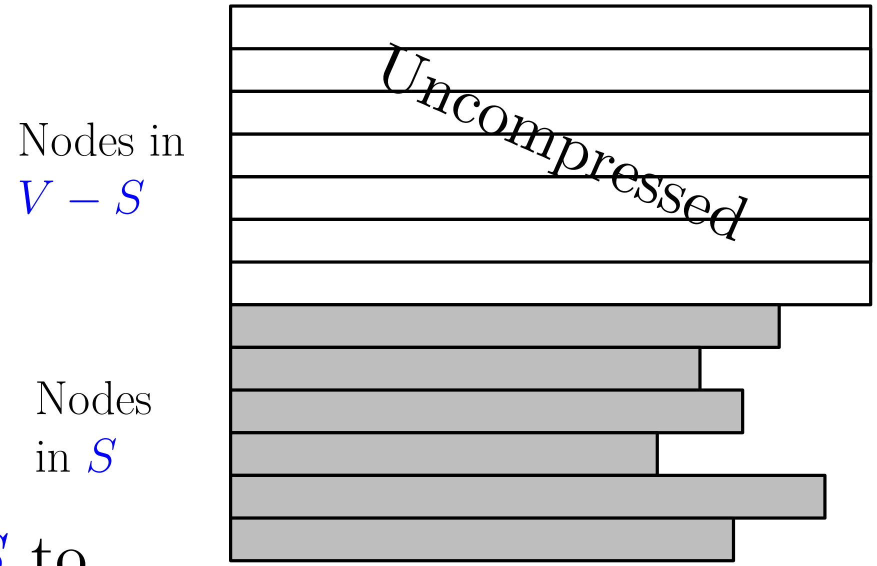


Compressing the Non-Expanding Set

Encoding:

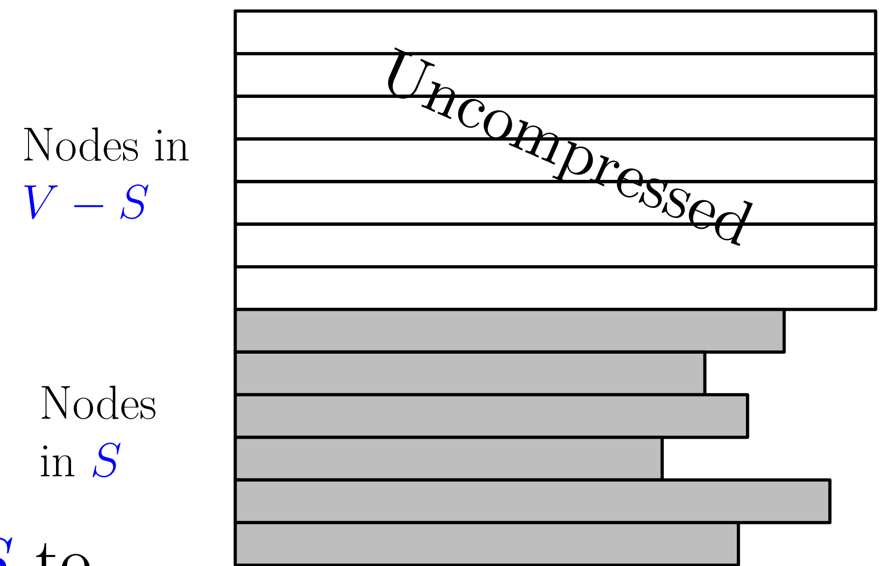
- Randomness of $V - S$
- Set S: $\log|S| + \log\binom{n}{s}$
- Accepted connections: $\sum_{v\in S} 2\log \ell_v + \log\binom{\ell_v}{d}$
- Accepted connections from S to $V - S$: $\sum_{v\in S} 2\log(\epsilon_v d) + \log\binom{d}{\epsilon_v d}$

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 $V-S$

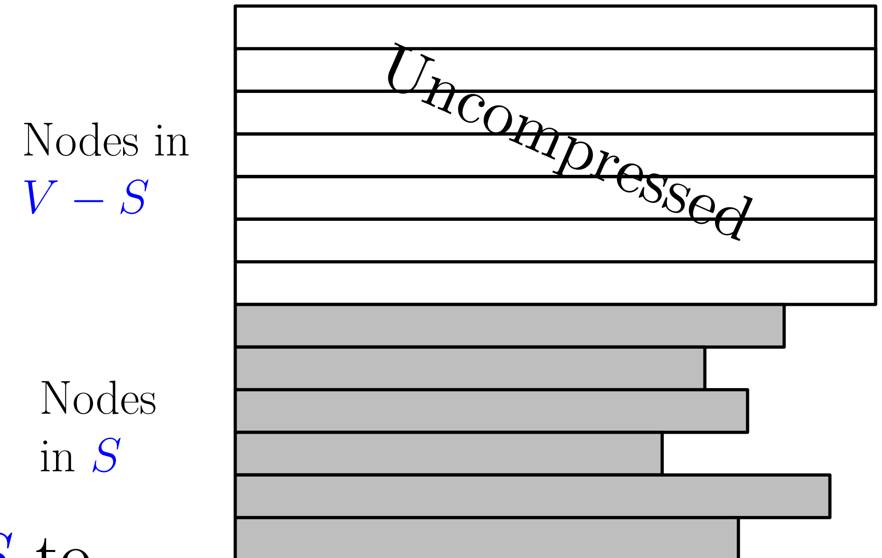
- Destinations of connections from S:

  $\underbrace{\sum_{v\in S}(1-\epsilon_v)d\log((1-\delta_v)\Delta)}_{\text{connections to } S} + \underbrace{\sum_{v\in S}\epsilon_v d\log\Delta}_{\text{connections to } V-S \text{ (uncompressed)}}$

  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:
  $$\underbrace{\sum_{v \in S} (1 - \epsilon_v) d \log((1 - \delta_v)\Delta)}_{\text{connections to } S} + \underbrace{\sum_{v \in S} \epsilon_v d \log \Delta}_{\text{connections to } V-S \text{ (uncompressed)}}$$
- **Rejected requests**

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

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:

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

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

Compressing Accepted Connections I

To represent accepted requests from S we need

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

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

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

Compressing Accepted Connections I

To represent accepted requests from S we need

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

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

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

With simple calculations

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

Compressing Accepted Connections I

To represent accepted requests from S we need

$$\sum_{v \in S} (1-\epsilon_v) d \log((1-\delta_v)\Delta) + \sum_{v \in S} \epsilon_v d \log \Delta$$
$$\leq sd \log \Delta - \boxed{\frac{1-\epsilon}{2} sd \log \frac{n}{s} + 2\epsilon ds}$$

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

With simple calculations

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

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

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

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

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

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

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**:

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

together with hypothesis on s and λ, it implies

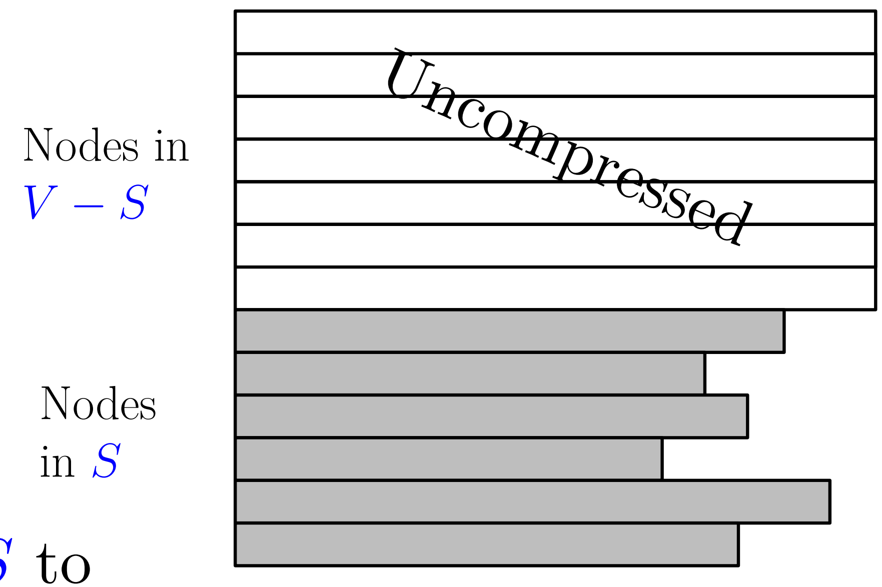
$$(1-\epsilon)sd\log\frac{1-\epsilon}{1-\delta}>(1-\epsilon)sd\log\frac{n}{s}-2\epsilon ds$$

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

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

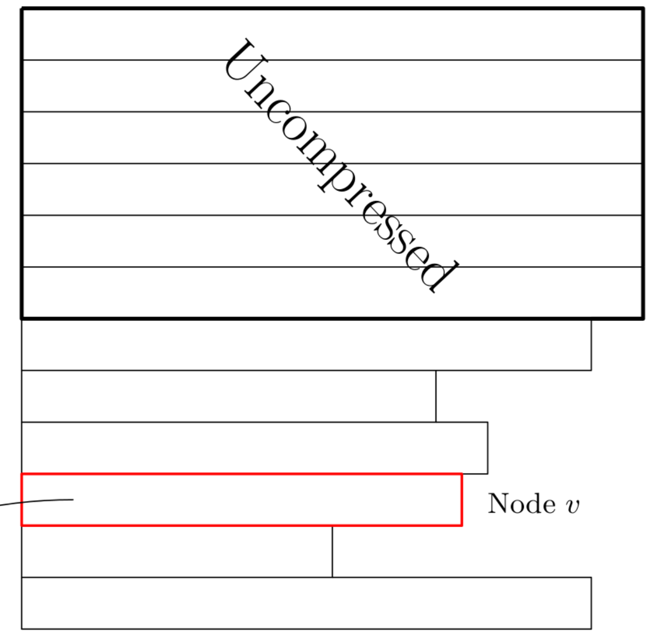
Critical Nodes

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.
$\ell_v - d$	$\log(n/c)$	$\log c_{t_1}$	$\log c_{t_2}$	$\log(n/c)$	$\log(n/c)$	$\log c_{t_k}$

Project Idea

Simulate the RAES protocol on *random Δ-regular graphs*, varying the parameters Δ, d and c.

- Becchetti, Clementi, N., Pasquale, Trevisan. 2019. "Finding a Bounded-Degree Expander Inside a Dense One.

Simulations should be performed using open-source software with some effort to make them efficient (e.g. coded in Python using Numpy), and the source code should be made publicly available (e.g. on Gitlab) and GPL licensed.

ATTENTION: Creating a random Δ-regular graph is not immediate!