Pseudorandomness: Some Distributed Applications

Alexandre Nolin

CISPA Helmholtz Center for Information Security

ADGA 2023, 09.10.2023

What this talk will be about

Goals of this talk

- Introduce several key ideas about pseudorandomness using a toy example.

  Existence via Probabilistic Method, connections to error correcting codes, hash functions.

- See (up to 3) examples how these ideas have been used in distributed computing.

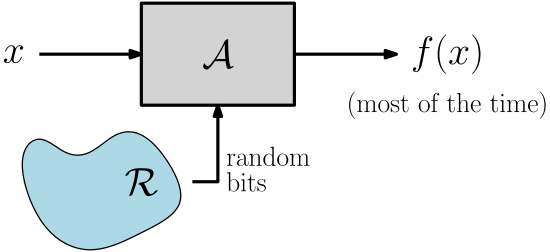
  Low-congestion sampling, derandomization.

- See some extra pseudorandom objects on the way.

  Expander graphs, pairwise independent hash functions.

What is pseudorandomness?

Given a randomized algorithm, can we swap its source of random bits for a simpler one?

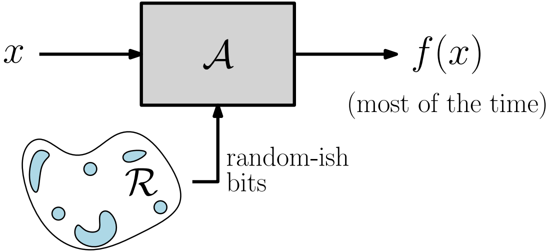


No:

Yes:

What is pseudorandomness?

Given a randomized algorithm, can we swap its source of random bits for a simpler one?

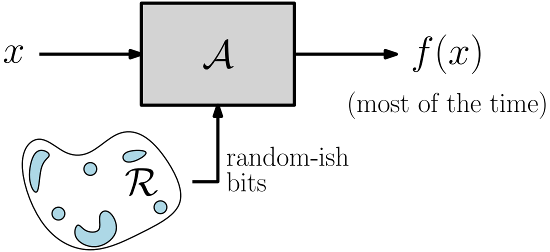


No:

Yes:

What is pseudorandomness?

Given a randomized algorithm, can we swap its source of random bits for a simpler one?

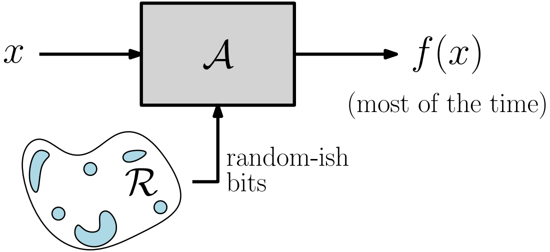


No: if each string fails on some input, some minimum needed.

Yes:

What is pseudorandomness?

Given a randomized algorithm, can we swap its source of random bits for a simpler one?

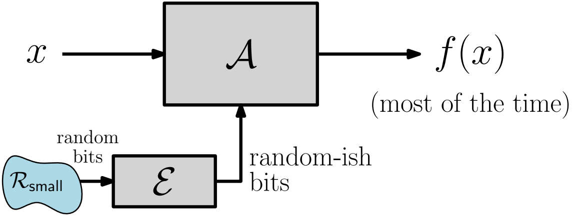


No: if each string fails on some input, some minimum needed.

Yes: an algorithm shouldn't be table to tell apart all distributions.

What is pseudorandomness?

Given a randomized algorithm, can we swap its source of random bits for a simpler one?



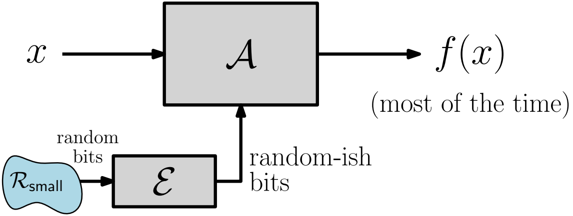
No: if each string fails on some input, some minimum needed.

Yes: an algorithm shouldn't be table to tell apart all distributions.

Furthermore: is the sparsification **reasonably computable**?

What is pseudorandomness?

Given a randomized algorithm, can we swap its source of random bits for a simpler one?



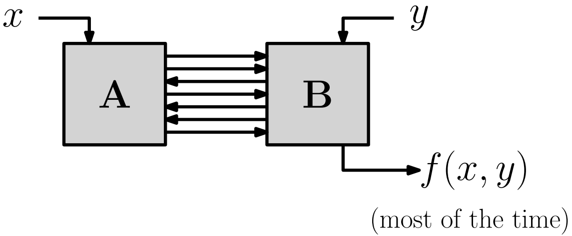
No: if each string fails on some input, some minimum needed.

Yes: an algorithm shouldn't be table to tell apart all distributions.

Furthermore: is the sparsification **reasonably computable**?

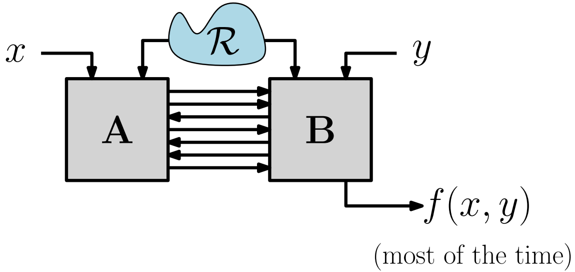
Pseudorandomness = study of when this is possible.

Example: Equality in 2-party Communication Complexity



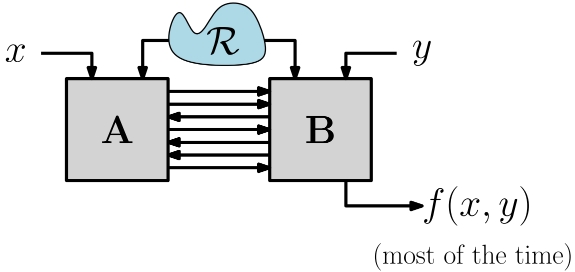
Take $f = \mathsf{EQ}_n$: $\mathsf{EQ}_n(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$ with $x, y \in \{0, 1\}^n$.

Example: Equality in 2-party Communication Complexity



Take $f = \mathsf{EQ}_n$: $\mathsf{EQ}_n(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$ with $x, y \in \{0, 1\}^n$.

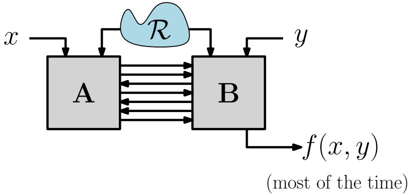
Example: Equality in 2-party Communication Complexity



Take $f = \mathsf{EQ}_n$: $\mathsf{EQ}_n(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$ with $x, y \in \{0, 1\}^n$.

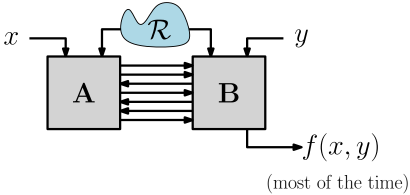
For error $\epsilon < 1/2$, simple algorithm using $\lceil \log(1/\epsilon) \rceil$ bits?

Algorithm for Equality with shared randomness



1. Repeat for t in $[\lceil \log(1/\epsilon) \rceil]$
   1.1 Alice reads $r^{(t)} \in \{0,1\}^n$ from shared randomness.
   1.2 Alice computes $\langle x, r^{(t)} \rangle = \sum_{i=1}^{n} x_i \cdot r_i^{(t)} \mod 2$, sends it to Bob.
2. Bob outputs "`Equal`" iff $\langle x, r^{(t)} \rangle = \langle y, r^{(t)} \rangle, \forall t \in [\lceil \log(1/\epsilon) \rceil]$

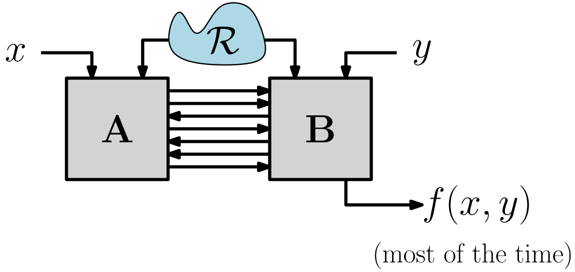
Algorithm for Equality with shared randomness



1. Repeat for t in $[\lceil \log(1/\epsilon) \rceil]$
   1.1 Alice reads $r^{(t)} \in \{0,1\}^n$ from shared randomness.
   1.2 Alice computes $\langle x, r^{(t)} \rangle = \sum_{i=1}^{n} x_i \cdot r_i^{(t)} \mod 2$, sends it to Bob.
2. Bob outputs "`Equal`" iff $\langle x, r^{(t)} \rangle = \langle y, r^{(t)} \rangle, \forall t \in [\lceil \log(1/\epsilon) \rceil]$

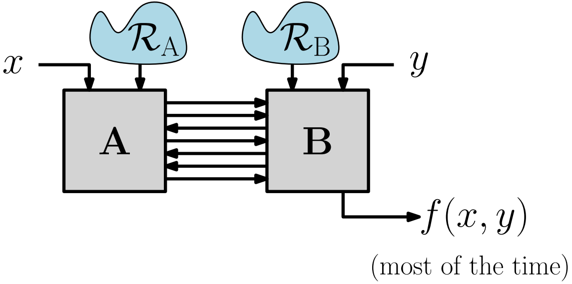
Proof: when $x \neq y$, each $r^{(t)}$ has a probability $1/2$ to be s.t. $\langle x, r^{(t)} \rangle \neq \langle y, r^{(t)} \rangle$ (to see it, focus on any index i s.t. $x_i \neq y_i$).

Do we need that many random strings?



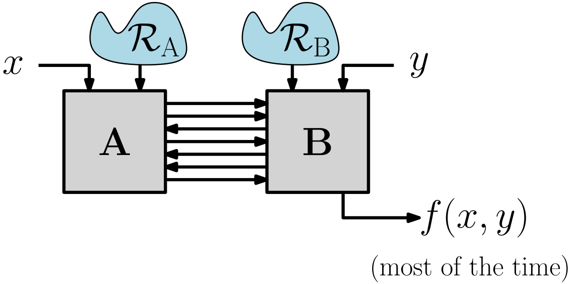
Algorithm from previous slide: $n \cdot \lceil \log(1/\epsilon) \rceil$ random bits.

Do we need that many random strings?



Algorithm from previous slide: $n \cdot \lceil \log(1/\epsilon) \rceil$ random bits.

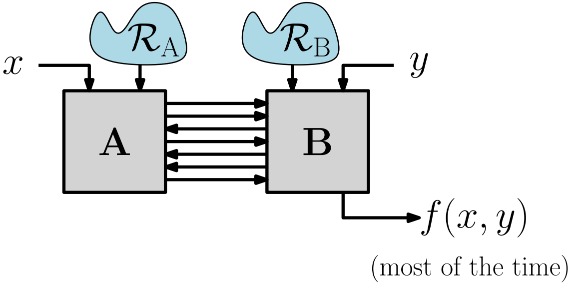
Do we need that many random strings?



Algorithm from previous slide: $n \cdot \lceil \log(1/\epsilon) \rceil$ random bits.

If we no longer have shared randomness, sampling the randomness on one side and sharing it is **way too expensive**.

Do we need that many random strings?



Algorithm from previous slide: $n \cdot \lceil \log(1/\epsilon) \rceil$ random bits.

If we no longer have shared randomness, sampling the randomness on one side and sharing it is **way too expensive**.

Can we identify a subset of the randomness that works?

First Important Idea: probabilistic method

Goal: find subset $R \subseteq \{0,1\}^n$ such that:

- (small) $|R| \ll 2^n$,
- (useful) $\forall x, y \in \{0,1\}^n, x \neq y, \Pr_{r \in R}[\langle x, r \rangle \neq \langle y, r \rangle] \geq 1/4$.

Result: there exists such a set R of size $O(n)$

First Important Idea: probabilistic method

Goal: find subset $R \subseteq \{0,1\}^n$ such that:

- (small) $|R| \ll 2^n$,
- (useful) $\forall x, y \in \{0,1\}^n, x \neq y, \Pr_{r \in R}[\langle x, r\rangle \neq \langle y, r\rangle] \geq 1/4$.

Result: there exists such a set R of size $O(n)$

Proof: Pick specific pair x, y with $x \neq y$. In k random strings $r_1, \ldots, r_k$, $k/2$ in expectation are s.t. $\langle x, r\rangle \neq \langle y, r\rangle$.

Probability that less than $k/4$ strings are s.t. $\langle x, r\rangle \neq \langle y, r\rangle$ is bounded by: $\exp(-k/12)$ (Chernoff bound)

Probality that less than $k/4$ strings are s.t. $\langle x, r\rangle \neq \langle y, r\rangle$ for some pair x, y: $2^{2n} \cdot \exp(-k/12) < 1$ for $k > (24\ln(2)) \cdot n$.

First Important Idea: probabilistic method

Goal: find subset $R \subseteq \{0,1\}^n$ such that:

- (small) $|R| \ll 2^n$,
- (useful) $\forall x, y \in \{0,1\}^n, x \neq y, \Pr_{r \in R}[\langle x, r\rangle \neq \langle y, r\rangle] \geq 1/4$.

Result: there exists such a set R of size $O(n)$

Consequence: Given only private randomness, EQ_n can still be solved with ϵ error in $O(\log(1/\epsilon) \cdot \log(n))$ communication.

First Important Idea: probabilistic method

Goal: find subset $R \subseteq \{0,1\}^n$ such that:

- (small) $|R| \ll 2^n$,
- (useful) $\forall x, y \in \{0,1\}^n, x \neq y, \Pr_{r \in R}[\langle x, r\rangle \neq \langle y, r\rangle] \geq 1/4$.

Result: there exists such a set R of size $O(n)$

Consequence: Given only private randomness, EQ_n can still be solved with ϵ error in $O(\log(1/\epsilon) \cdot \log(n))$ communication.

Limitation: Is the algorithm explicit? How expensive is it to compute R?

First Important Idea: probabilistic method

Goal: find subset $R \subseteq \{0,1\}^n$ such that:

- (small) $|R| \ll 2^n$,
- (useful) $\forall x, y \in \{0,1\}^n, x \neq y, \Pr_{r \in R}[\langle x, r \rangle \neq \langle y, r \rangle] \geq 1/4$.

Result: there exists such a set R of size $O(n)$

Consequence: Given only private randomness, EQ_n can still be solved with ϵ error in $O(\log(1/\epsilon) \cdot \log(n))$ communication.

Limitation: Is the algorithm explicit? How expensive is it to compute R? **exponential** (in n)

Probabilistic method at its fullest: pseudorandom generators

More generally: Consider any randomized algorithm with N inputs, failure probability ϵ. We can identify a subset of its randomness of size $O(\epsilon^{-1} \log N)$ such that the error is at most 2ϵ when using random strings from this sparse randomness.

Probabilistic method at its fullest: pseudorandom generators

More generally: Consider any randomized algorithm with N inputs, failure probability ϵ. We can identify a subset of its randomness of size $O(\epsilon^{-1} \log N)$ such that the error is at most 2ϵ when using random strings from this sparse randomness.

Even more generally: There is a set of $O(n)$ random strings such that for all circuits with a description of size n each correctly computing some function with error ϵ, they still all correctly compute their function if using this randomness.

Probabilistic method at its fullest: pseudorandom generators

More generally: Consider any randomized algorithm with N inputs, failure probability ϵ. We can identify a subset of its randomness of size $O(\epsilon^{-1} \log N)$ such that the error is at most 2ϵ when using random strings from this sparse randomness.

Even more generally: There is a set of $O(n)$ random strings such that for all circuits with a description of size n each correctly computing some function with error ϵ, they still all correctly compute their function if using this randomness.

Pseudorandom generators: generators of such strings. An explicit construction would give $\mathsf{P} = \mathsf{BPP}$, by enumerating through all the strings.

Second Important Idea: connection to other objects

Pick a set of random strings $R \subseteq \{0,1\}^n$ like we just constructed, i.e., for any $x, y \in \{0,1\}^n$ such that $x \neq y$:

$$\{r \in R : \langle x, r \rangle \neq \langle y, r \rangle\} \geq |R|/4 \ .$$

For each $x \in \{0,1\}^n$, consider the $|R|$-bit word $w_R(x)$:

$$w_R(x) = \langle x, r_1 \rangle . \langle x, r_2 \rangle . \ldots . \langle x, r_{|R|} \rangle \ .$$

Property:

Second Important Idea: connection to other objects

Pick a set of random strings $R \subseteq \{0,1\}^n$ like we just constructed, i.e., for any $x, y \in \{0,1\}^n$ such that $x \neq y$:

$$\{r \in R : \langle x, r\rangle \neq \langle y, r\rangle\} \geq |R|/4 \; .$$

For each $x \in \{0,1\}^n$, consider the $|R|$-bit word $w_R(x)$:

$$w_R(x) = \langle x, r_1\rangle.\langle x, r_2\rangle.\ldots.\langle x, r_{|R|}\rangle \; .$$

Property: for any pair $x \neq y$, $w_R(x)$ and $w_R(y)$ differ on at least $|R|/4$ bits.

Second Important Idea: connection to other objects

Pick a set of random strings $R \subseteq \{0,1\}^n$ like we just constructed, i.e., for any $x, y \in \{0,1\}^n$ such that $x \neq y$:

$$\{r \in R : \langle x, r \rangle \neq \langle y, r \rangle\} \geq |R|/4 \ .$$

For each $x \in \{0,1\}^n$, consider the $|R|$-bit word $w_R(x)$:

$$w_R(x) = \langle x, r_1 \rangle . \langle x, r_2 \rangle . \dots . \langle x, r_{|R|} \rangle \ .$$

Property: for any pair $x \neq y$, $w_R(x)$ and $w_R(y)$ differ on at least $|R|/4$ bits.

It's an Error Correcting Code!

With parameters $[24 \ln(2) n, n, 6 \ln(2) n]$, over $\mathbb{F}_2$.

Second Important Idea: connection to other objects

Pick a set of random strings $R \subseteq \{0,1\}^n$ like we just constructed, i.e., for any $x, y \in \{0,1\}^n$ such that $x \neq y$:

$$\{r \in R : \langle x, r\rangle \neq \langle y, r\rangle\} \geq |R|/4 \ .$$

For each $x \in \{0,1\}^n$, consider the $|R|$-bit word $w_R(x)$:

$$w_R(x) = \langle x, r_1\rangle . \langle x, r_2\rangle . \ldots . \langle x, r_{|R|}\rangle \ .$$

Property: for any pair $x \neq y$, $w_R(x)$ and $w_R(y)$ differ on at least $|R|/4$ bits.

It's an Error Correcting Code!

With parameters $[24\ln(2)n, n, 6\ln(2)n]$, over $\mathbb{F}_2$.

Conversely: Can get more randomness-efficient algorithms from results on Error Correcting Codes.

Second Important Idea continued: another connected object

We can also see each $r \in R$ as a map h_r: $\{0,1\}^n \to 0, 1$, with $h_r(x) = \langle x, r \rangle$

Definition ((ϵ-almost) Pairwise-Independent Hash Functions)

A set $\mathcal{H}$ of functions $h : [N] \to [M]$ s.t.:
$\forall x \neq y \in [N]^2, z, z' \in [M]^2$,
$|\Pr_{h \in \mathcal{H}}[h(x) = z \wedge h(y) = z'] - 1/M^2| \leq \epsilon/M^2$.

There exists a family of such hash functions of size $\text{poly}(M, 1/\epsilon)(\log N / \log\log N)$.

Lessons learned so far

Given ANY randomized algorithm, we can reduce its use of randomness to $O(\log n)$ bits without affecting its success/failure probability too much.

Lessons learned so far

Given ANY randomized algorithm, we can reduce its use of randomness to $O(\log n)$ bits without affecting its success/failure probability too much.

However this is existential. While existence of some pseudorandom object can often be done by probabilistic method, finding an explicit construction can be harder.

Lessons learned so far

Given ANY randomized algorithm, we can reduce its use of randomness to $O(\log n)$ bits without affecting its success/failure probability too much.

However this is existential. While existence of some pseudorandom object can often be done by probabilistic method, finding an explicit construction can be harder.

Going the last mile requires finding some explicit structure that works in the given application (error correcting codes, families of hash functions, fields, polynomials, classes of graphs).

Distributed applications

Distributed application 1: random sampling

Setting: CONGEST model:

- Graph $G = (V, E)$ is the input and the communication network.
- $n =$ number of nodes, $\Delta =$ maximum degree. Known by all nodes.
- In a round, each node can send $O(\log n)$ bits on each incident edge.
- Complexity is the number of rounds to achieve the wanted result.

LOCAL model: same with infinite bandwidth.

Distributed application 1: random sampling

Setting: CONGEST model:

- Graph $G = (V, E)$ is the input and the communication network.
- $n =$ number of nodes, $\Delta =$ maximum degree. Known by all nodes.
- In a round, each node can send $O(\log n)$ bits on each incident edge.
- Complexity is the number of rounds to achieve the wanted result.

LOCAL model: same with infinite bandwidth.

- k-coloring: each node must be properly colored with a value from $\{1, \ldots, k\}$.
- k-list-coloring: each node starts with a list of k colors, from which it must choose its color.
- palette Ψ_v: colors still available to a node v as it avoids colors already chosen by neighbors.
- edge-coloring: we have to color the edges instead of the nodse

Distributed application 1: random sampling

Setting: CONGEST model:

- Graph $G = (V, E)$ is the input and the communication network.
- $n =$ number of nodes, $\Delta =$ maximum degree. Known by all nodes.
- In a round, each node can send $O(\log n)$ bits on each incident edge.
- Complexity is the number of rounds to achieve the wanted result.

LOCAL model: same with infinite bandwidth.

Highlights from [HN23, HNT22], lowering the complexity in CONGEST to that in LOCAL (or almost):

- In $O(\log^* n)$: $(1+\epsilon)\Delta$ coloring, $(1+\epsilon)\Delta$ edge-coloring, when $\Delta \in \Omega(\log^{1+1/\log^*} n)$. [HN23]
- $\deg +1$-list-coloring in $O(\log^5 \log n)$ [HNT22], $(2\Delta - 1)$-edge coloring in $O(\log^4 \log n)$ rounds. [HN23]

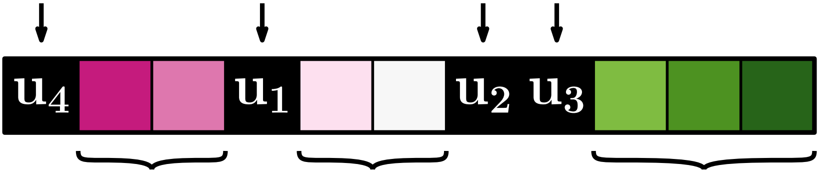
  (saving an additive $\log \Delta$ in both cases)

Fast coloring with an excess of colors [SW10]

- Consider the nodes v s.t. $|\Psi_v| \geq 2x \cdot d_v$.
  (i.e., with $\times 2x$ more available colors than active neighbors)
- Let each such node try x colors.
- They each get colored w.p. $1 - 2^{-x}$.

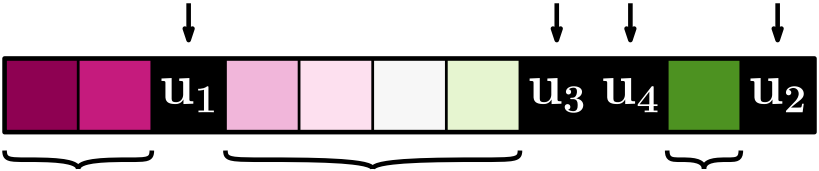
Fast coloring with an excess of colors [SW10]

- Consider the nodes v s.t. $|\Psi_v| \geq 2x \cdot d_v$.
  (i.e., with $\times 2x$ more available colors than active neighbors)
- Let each such node try x colors.
- They each get colored w.p. $1 - 2^{-x}$.



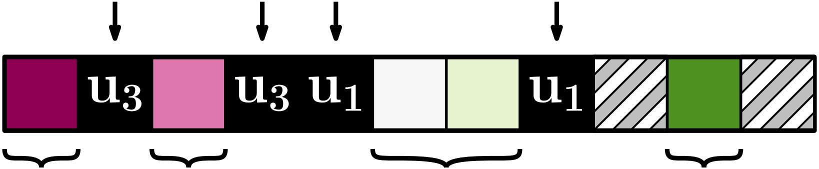
Fast coloring with an excess of colors [SW10]

- Consider the nodes v s.t. $|\Psi_v| \geq 2x \cdot d_v$.
  (i.e., with $\times 2x$ more available colors than active neighbors)
- Let each such node try x colors.
- They each get colored w.p. $1 - 2^{-x}$.



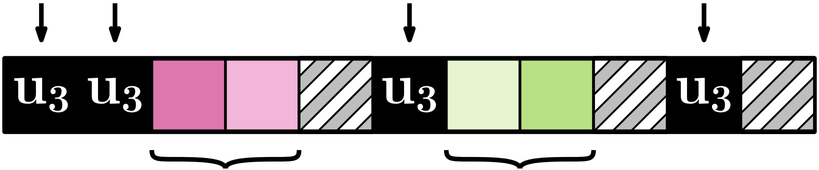
Fast coloring with an excess of colors [SW10]

- Consider the nodes v s.t. $|\Psi_v| \geq 2x \cdot d_v$.
  (i.e., with $\times 2x$ more available colors than active neighbors)
- Let each such node try x colors.
- They each get colored w.p. $1 - 2^{-x}$.



Fast coloring with an excess of colors [SW10]

- Consider the nodes v s.t. $|\Psi_v| \geq 2x \cdot d_v$.
  (i.e., with $\times 2x$ more available colors than active neighbors)
- Let each such node try x colors.
- They each get colored w.p. $1 - 2^{-x}$.



Fast coloring with an excess of colors [SW10]

Algorithm for 2Δ-coloring in LOCAL, for a node v, with neighbors of higher ID $N^+(v)$.

1. Set $x \leftarrow 1$.
2. For $i = 1$ to $O(\log^* n)$
   2.1 Repeat $O(1)$ times:
      2.1.1 v picks a set S_v of x random colors in its palette Ψ_v.
      2.1.2 v sends S_v to $N(v)$, computes $T_v = S_v \setminus \bigcup_{u \in N^+(v)} S_u$.
      2.1.3 If $T_v \neq \emptyset$, v permanently adopts a color in T_v.
      It sends its final color to its neighbors and stops the algorithm.
   2.2 Set $x \leftarrow \min(2^x, \log n)$.
3. If $\Delta < O(\log^{1+1/\log^* n} n)$, finish the coloring with a deterministic algorithm. [GK21]

Fast coloring with an excess of colors [SW10]

Algorithm for 2Δ-coloring in $\textsc{Local}$, for a node v, with neighbors of higher ID $N^+(v)$.

1. Set $x \leftarrow 1$.
2. For $i = 1$ to $O(\log^* n)$
   2.1 Repeat $O(1)$ times:
      2.1.1 v picks a set S_v of x random colors in its palette Ψ_v.
      2.1.2 v sends S_v to $N(v)$, computes $T_v = S_v \setminus \bigcup_{u \in N^+(v)} S_u$.
      2.1.3 If $T_v \neq \emptyset$, v permanently adopts a color in T_v.
      It sends its final color to its neighbors and stops the algorithm.

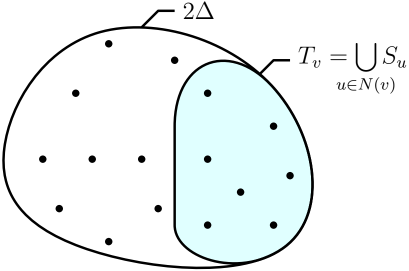
   2.2 Set $x \leftarrow \min(2^x, \log n)$.
3. If $\Delta < O(\log^{1+1/\log^* n} n)$, finish the coloring with a deterministic algorithm. [GK21]

Proof idea: the degree of each node is bounded by $\max(\frac{2\Delta}{x}, O(\log n))$, w.h.p., throughout the algorithm.

Doing the same in CONGEST: the cost of sending a set

Basic argument of the algorithm is:

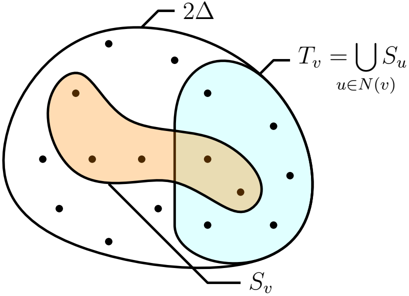
- given a space of colors $\mathcal{C}$, and
- a subset $T \subseteq \mathcal{C}$ of colors to avoid of size $|T| \leq |\mathcal{C}|/2$,
- when taking a random set $S \subseteq \mathcal{C}$ of size x, the probability that $S \subseteq T$ is $\leq 2^{-x}$.



Doing the same in CONGEST: the cost of sending a set

Basic argument of the algorithm is:

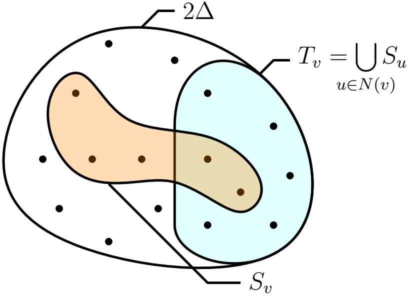
- given a space of colors $\mathcal{C}$, and
- a subset $T \subseteq \mathcal{C}$ of colors to avoid of size $|T| \leq |\mathcal{C}|/2$,
- when taking a random set $S \subseteq \mathcal{C}$ of size x, the probability that $S \subseteq T$ is $\leq 2^{-x}$.



Doing the same in CONGEST: the cost of sending a set

Basic argument of the algorithm is:

- given a space of colors $\mathcal{C}$, and
- a subset $T \subseteq \mathcal{C}$ of colors to avoid of size $|T| \leq |\mathcal{C}|/2$,
- when taking a random set $S \subseteq \mathcal{C}$ of size x, the probability that $S \subseteq T$ is $\leq 2^{-x}$.



Issue for Congest: describing an arbitrary set $S \subseteq [2\Delta]$, $|S_v| \in [1, \Theta(\log n)]$ requires $\Theta(\log n \cdot \log \Delta)$ bits.

Solution 1 (existential), by probabilistic method

Same proof scheme as before:

- Fix $T \subseteq [2\Delta]$, $|T| \leq \Delta$.
- For a random set $S \subseteq [2\Delta]$ of size x, $\Pr[S \subseteq T] \leq 2^{-x}$.
- Taking k random sets $S_1 \dots S_k \subseteq [2\Delta]$, $\leq k \cdot 2^{-x}$ are expected to be $\subseteq T$.
- The probability that $\geq 2 \cdot k2^{-x}$ of the sets are $\subseteq T$ is $\leq e^{-k \cdot 2^{-x}/3}$.
- There are $\leq 2^{2\Delta}$ possible sets T.
- With $x \in O(\log n)$, $k \in \text{poly}(\log \Delta, n)$, k random sets are s.t. $T \subseteq [2\Delta]$, $|T| \leq \Delta$, $|\{i : S_i \subseteq T\}| \leq 2k2^{-x}$ with probability > 0.

Solution 1 (existential), by probabilistic method

Same proof scheme as before:

- Fix $T \subseteq [2\Delta]$, $|T| \leq \Delta$.
- For a random set $S \subseteq [2\Delta]$ of size x, $\Pr[S \subseteq T] \leq 2^{-x}$.
- Taking k random sets $S_1 \dots S_k \subseteq [2\Delta]$, $\leq k \cdot 2^{-x}$ are expected to be $\subseteq T$.
- The probability that $\geq 2 \cdot k2^{-x}$ of the sets are $\subseteq T$ is $\leq e^{-k \cdot 2^{-x}/3}$.
- There are $\leq 2^{2\Delta}$ possible sets T.
- With $x \in O(\log n)$, $k \in \text{poly}(\log \Delta, n)$, k random sets are s.t. $T \subseteq [2\Delta]$, $|T| \leq \Delta$, $|\{i : S_i \subseteq T\}| \leq 2k2^{-x}$ with probability > 0.

Can strengthen it a bit: For any sufficiently large set T, most sets $S_1 \dots S_{\text{poly}(\log \Delta, n)}$ intersect T in $\approx S_i \cdot |T|/(2\Delta)$ elements.

Representative sets

Lemma (Representative sets)

Let U be a universe of size k. A family $\mathcal{F} = \{S_1, \ldots, S_t\}$ of s-sized sets is said to be an (α, δ, ν)-representative family iff:

$$\forall T \subseteq U, |T| \geq \delta k : \quad \Pr_{i \in_u [t]}\left[\left|\frac{|S_i \cap T|}{|S_i|} - \frac{|T|}{k}\right| \leq \alpha \frac{|T|}{k}\right] \geq (1-\nu), \quad (1)$$

$$\forall T \subseteq U, |T| < \delta k : \quad \Pr_{i \in_u [t]}\left[\frac{|S_i \cap T|}{|S_i|} - \delta \leq \alpha\delta\right] \geq (1-\nu), \quad (2)$$

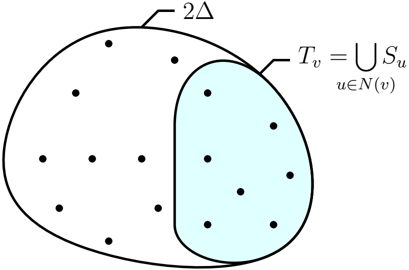
$$\forall u \in U : \quad \Pr_{i \in_u [t]}[u \in S_i] \in [1-\alpha, 1+\alpha]\frac{s \cdot t}{k}. \quad (3)$$

Such families exist for $t \in \Theta(k/\nu + k \log k)$ and $s \in \Theta(\alpha^{-2}\delta^{-1}\log(1/\nu))$.

Proof: probabilistic method.

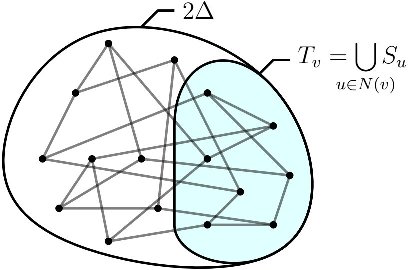
Solution 2 (explicit), using expander graphs

We add a graph structure on the space of colors.



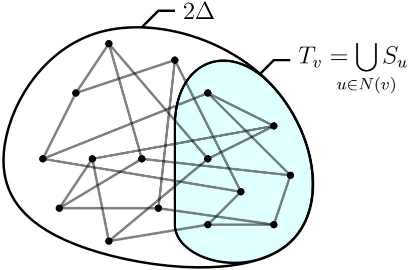
Solution 2 (explicit), using expander graphs

We add a graph structure on the space of colors.



Solution 2 (explicit), using expander graphs

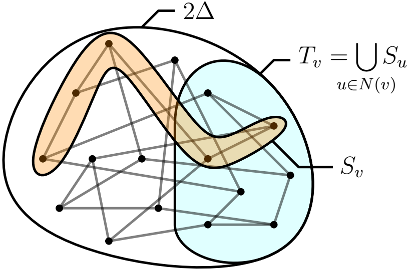
We add a graph structure on the space of colors.



On a graph of degree $O(1)$, describing a walk of length k takes $O(\log \Delta)$ (to describe the starting vertex) $+O(k)$ bits (to describe the k steps taken from this starting node).

Solution 2 (explicit), using expander graphs

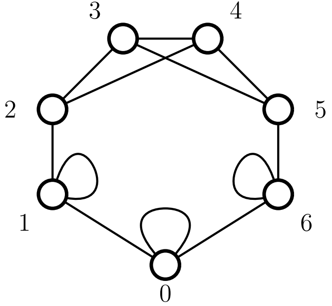
We add a graph structure on the space of colors.



On a graph of degree $O(1)$, describing a walk of length k takes $O(\log \Delta)$ (to describe the starting vertex) $+O(k)$ bits (to describe the k steps taken from this starting node).

But why should such sets have the right properties?

Spectral expanders and random walks



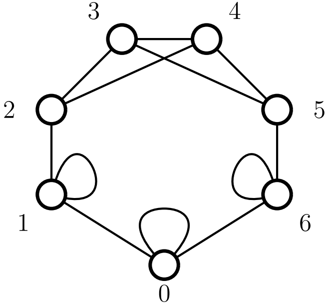
1	1	0	0	0	0	1
1	1	1	0	0	0	0
0	1	0	1	1	0	0
0	0	1	0	1	1	0
0	0	1	1	0	1	0
0	0	0	1	1	1	1
1	0	0	0	0	1	0

Consider a connected, non-bipartite, regular digraph G, transition matrix M_G.

Consider eigenvalues of M_G $\lambda_1, \ldots, \lambda_n$, s.t. $|\lambda_i| \geq |\lambda_{i+1}|$

The largest eigenvalue is always $\lambda_1 = 1$ and corresponds to the uniform distribution, others are < 1.

Spectral expanders and random walks



$$\begin{matrix} 1/3 & 1/3 & 0 & 0 & 0 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 0 & 0 & 0 & 1/3 & 0 \end{matrix}$$

Consider a connected, non-bipartite, regular digraph G, transition matrix M_G.

Consider eigenvalues of M_G $\lambda_1, \dots, \lambda_n$, s.t. $|\lambda_i| \geq |\lambda_{i+1}|$

The largest eigenvalue is always $\lambda_1 = 1$ and corresponds to the uniform distribution, others are < 1.

Expanders mix well

The spectral gap $\gamma = 1 - |\lambda_2|$ captures how fast and close a random walk gets to the uniform distribution.

Theorem ([Hea08, WX05])

Consider an expander graph G with spectral gap $\gamma = 1 - |\lambda_2|$, and a subset $S \subseteq V$ of its nodes. Consider a random walk of length k, and let X measure how many steps of the walk are in S. For any $\epsilon > 0$:

$$\Pr\left[\left|\frac{1}{k}X - \frac{|S|}{n}\right| \geq \epsilon\right] \leq 2e^{-\epsilon^2\gamma k/4}$$

Expanders mix well

The spectral gap $\gamma = 1 - |\lambda_2|$ captures how fast and close a random walk gets to the uniform distribution.

Theorem ([Hea08, WX05])

Consider an expander graph G with spectral gap $\gamma = 1 - |\lambda_2|$, and a subset $S \subseteq V$ of its nodes. Consider a random walk of length k, and let X measure how many steps of the walk are in S. For any $\epsilon > 0$:

$$\Pr\left[\left|\frac{1}{k}X - \frac{|S|}{n}\right| \geq \epsilon\right] \leq 2e^{-\epsilon^2 \gamma k/4}$$

Just need to find an expander over 2Δ nodes, with constant degree, spectral gap $\Omega(1)$.

Existence of expanders

Do expanders like we want exist?

Existence of expanders

Do expanders like we want exist?

Well yes!

Existence of expanders

Do expanders like we want exist?

Well yes! By probabilistic method!

...

Existence of expanders

Do expanders like we want exist?

Well yes! By probabilistic method!

...

But explicit constructions are also known.

Graphs defined by the structure of a prime fields, recursive constructions...

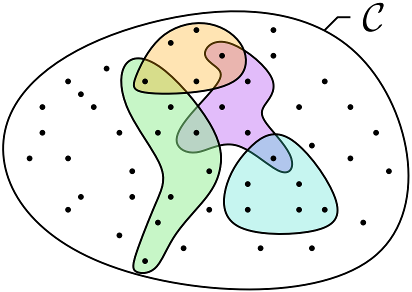
Consequences [HN23]

Δ	Coloring tasks	Complexity in CONGEST
$\Omega(\log^{1+1/\log^* n} n)$	$(1+\epsilon)\Delta$-vertex $(1+\epsilon)\Delta$-edge	$O(\log^* n)$
$O(\log^{1+1/\log^* n} n)$	$(1+\epsilon)\Delta$-vertex	$O(\log^3 \log n)$
	$(2\Delta - 1)$-edge	$O(\log^4 \log n)$
$\Omega(\sqrt{\log^{1+1/\log^* n} n})$	$(1+\epsilon)\Delta^2$-vertex distance-2	$O(\log^* n)$
$O(\sqrt{\log^{1+1/\log^* n} n})$		$O(\log^4 \log n)$
$\Omega(\log^{1+1/c'} n)$	$\Delta \log^{(c)}$-vertex $\Delta \log^{(c)}$-edge	$O(1)$
$\Omega(\sqrt{\log^{1+1/c'} n})$	$\Delta^2 \log^{(c)} n$-vertex distance-2	

Techniques: Rödl nibble [DGP98] (for $(1+\epsilon)\Delta$-edge coloring), shattering [BEPS16] and deterministic algorithm [GK21] (for smaller Δ).

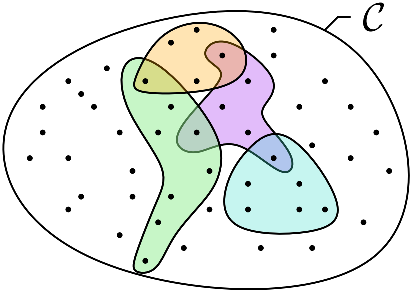
Harder case: non-shared color space [HNT22]

Suppose now that each node has a **list** of 2Δ colors in a much bigger universe of colors $\mathcal{C}$.



Harder case: non-shared color space [HNT22]

Suppose now that each node has a **list** of 2Δ colors in a much bigger universe of colors $\mathcal{C}$.



The expander construction no longer works here. How do we extend the argument?

Existential solution: pure hashing

Suppose nodes could sample and communicate a perfect hash function $\mathcal{C} \to [4\Delta]$. To try colors, nodes could:

1. Consider the part of their palette that hash to a value $\leq \Theta(\log n)$. ($\Theta(\log n)$ do, in expectation)
2. Send a $\Theta(\log n)$ bitmap to say "I'm trying colors with these hashes"

Sketch of the why and how

Existential solution: pure hashing

Suppose nodes could sample and communicate a perfect hash function $\mathcal{C} \to [4\Delta]$. To try colors, nodes could:

1. Consider the part of their palette that hash to a value $\leq \Theta(\log n)$. ($\Theta(\log n)$ do, in expectation)
2. Send a $\Theta(\log n)$ bitmap to say "I'm trying colors with these hashes"

Sketch of the why and how

Using hashing adds another source of failure (collision between colors), but overall, when trying x colors, one of the tried color should work with probability $1 - \exp(-\Omega(x))$.

Existential solution: pure hashing

Suppose nodes could sample and communicate a perfect hash function $\mathcal{C} \to [4\Delta]$. To try colors, nodes could:

1. Consider the part of their palette that hash to a value $\leq \Theta(\log n)$. ($\Theta(\log n)$ do, in expectation)
2. Send a $\Theta(\log n)$ bitmap to say "I'm trying colors with these hashes"

Sketch of the why and how

Using hashing adds another source of failure (collision between colors), but overall, when trying x colors, one of the tried color should work with probability $1 - \exp(-\Omega(x))$.

As usual, **probabilistic method** to find a small set of hash functions with the right properties.

Existential solution: pure hashing

Suppose nodes could sample and communicate a perfect hash function $\mathcal{C} \rightarrow [4\Delta]$. To try colors, nodes could:

1. Consider the part of their palette that hash to a value $\leq \Theta(\log n)$. ($\Theta(\log n)$ do, in expectation)
2. Send a $\Theta(\log n)$ bitmap to say "I'm trying colors with these hashes"

Sketch of the why and how

Using hashing adds another source of failure (collision between colors), but overall, when trying x colors, one of the tried color should work with probability $1 - \exp(-\Omega(x))$.

As usual, **probabilistic method** to find a small set of hash functions with the right properties.

Extends all previous results to the list-setting, but with very large local computation.

Doing without the existential arguments

1. Each node v finds an $\epsilon/2$-almost-pairwise independent hash function $h_v : \mathcal{C} \to [4\Delta/\epsilon]$ with $\leq \epsilon\Delta$ collisions.
2. Each node samples colors/hashes using the expander graph structure applied to the hash space $[2\Delta/\epsilon]$.
3. Neighbors of a node answer if they are trying a color with the same hash through h_v than colors tried by v.

Doing without the existential arguments

1. Each node v finds an $\epsilon/2$-almost-pairwise independent hash function $h_v : \mathcal{C} \to [4\Delta/\epsilon]$ with $\leq \epsilon\Delta$ collisions.
2. Each node samples colors/hashes using the expander graph structure applied to the hash space $[2\Delta/\epsilon]$.
3. Neighbors of a node answer if they are trying a color with the same hash through h_v than colors tried by v.

General idea: each node has at least $(2 - \epsilon)\Delta$ colors which are not in collision with its hash function.

Each color tried by a neighbor removes at most one possible hash for v. As before, nodes sample more and more colors but always guaranteeing that each node has a constant fraction of its palette untried by neighbors in any given round.

Each node, when sampling hashes, should thus succeed in finding a color with the usual probability.

The point of it

Allowed us to get a $\text{poly} \log \log n$ algorithm for $\deg + 1$-list-coloring, and the list-coloring versions of all previously mentionned problems (with extra colors).

Also, the algorithm for $\deg + 1$-list-coloring finishes in $O(\log^* n)$ for $\Delta > \log^7 n$.

Also useful in subsequent work: see distance-2 paper at this DISC [FHN23].

Distributed application 2: derandomization

Pseudorandomness is also essential to derandomizing randomized algorithms: less random bits to deterministically choose, the better!

- MPC [CDP21a, CDP21b, CC22, FGG22, CCDM23a]
- Congested Clique [CPS20, CDP21c, CCDM23b]

Concluding

To go further if this has piqued your interest, “Pseudorandomness” by Salil Vadhan is a great survey. [Vad12]

Concluding

To go further if this has piqued your interest, "Pseudorandomness" by Salil Vadhan is a great survey. [Vad12]

Some open questions:

- explicit representative hash functions?

  The explicit solution presented before works for distance-1 node-coloring, not some other settings (edge-coloring, distance-2 coloring).

- For the MPC results relying on pseudorandom generators, do without them.

  The PRGs fit in low-space MPC, but requires exponential computation.

Concluding

To go further if this has piqued your interest, “Pseudorandomness” by Salil Vadhan is a great survey. [Vad12]

Some open questions:

- explicit representative hash functions?

  The explicit solution presented before works for distance-1 node-coloring, not some other settings (edge-coloring, distance-2 coloring).

- For the MPC results relying on pseudorandom generators, do without them.

  The PRGs fit in low-space MPC, but requires exponential computation.

Thanks!

Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider.
The locality of distributed symmetry breaking.
Journal of the ACM, 63(3):20:1–20:45, 2016.

Sam Coy and Artur Czumaj.
Deterministic massively parallel connectivity.
In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 162–175. ACM, 2022.

Sam Coy, Artur Czumaj, Peter Davies, and Gopinath Mishra.
Fast parallel degree+1 list coloring.
CoRR, abs/2302.04378, 2023.

Sam Coy, Artur Czumaj, Peter Davies, and Gopinath Mishra.
Optimal (degree+1)-coloring in congested clique.

In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 46:1–46:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

Artur Czumaj, Peter Davies, and Merav Parter.
Graph sparsification for derandomizing massively parallel computation with low space.
ACM Trans. Algorithms, 17(2):16:1–16:27, 2021.

Artur Czumaj, Peter Davies, and Merav Parter.
Improved deterministic $(\Delta+1)$ coloring in low-space MPC.
In *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 469–479. ACM, 2021.

Artur Czumaj, Peter Davies, and Merav Parter.
Simple, deterministic, constant-round coloring in congested clique and MPC.
SIAM J. Comput., 50(5):1603–1626, 2021.

Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman.
Derandomizing local distributed algorithms under bandwidth restrictions.
Distributed Comput., 33(3-4):349–366, 2020.

Devdatt Dubhashi, David A Grable, and Alessandro Panconesi.

Near-optimal, distributed edge colouring via the nibble method.
Theoretical Computer Science, 203(2):225–252, 1998.

Michael Elkin, Seth Pettie, and Hsin-Hao Su.
$(2\Delta - 1)$-edge-coloring is much easier than maximal matching in the distributed setting.
In *Proc. of 26th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 355–370, 2015.

Manuela Fischer, Jeff Giliberti, and Christoph Grunau.

Improved deterministic connectivity in massively parallel computation.
In *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA*, volume 246 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin.
Fast coloring despite congested relays.
In *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*, volume 281 of *LIPIcs*, pages 19:1–19:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

Mohsen Ghaffari and Fabian Kuhn.
Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition.
In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1009–1020. IEEE, 2021.

Alexander Healy.
Randomness-efficient sampling within NC1.
Computational Complexity, 17:3–37, 2008.

Magnús M. Halldórsson and Alexandre Nolin.
Superfast coloring in congest via efficient color sampling.
Theoretical Computer Science, 948:113711, 2023.

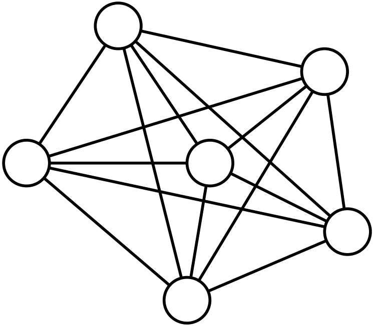
Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan.
Overcoming congestion in distributed coloring.
In Alessia Milani and Philipp Woelfel, editors, *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 26–36. ACM, 2022.

Johannes Schneider and Roger Wattenhofer.
A new technique for distributed symmetry breaking.
In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 257–266. ACM, 2010.

Salil P. Vadhan.
Pseudorandomness.
Foundations and Trends in Theoretical Computer Science, 7(1-3):1–336, 2012.

Avi Wigderson and David Xiao.
A randomness-efficient sampler for matrix-valued functions and applications.
In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 397–406. IEEE Computer Society, 2005.

Creating slack: sparsity



Intuition: if your neighbors are mostly disconnected, it's likely some of them will pick the same color.

Formally: $\zeta_v = \frac{1}{\Delta}\left(\binom{\Delta}{2} - |E[N(v)]|\right)$.

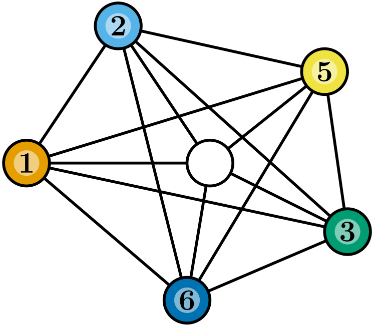
Theorem ([EPS15]): Let each node try a random color, then afterwards:

$$\Pr[s_v \leq c \cdot \zeta_v] \leq \exp(-c' \cdot \zeta_v).$$

for some universal constants c and c'.

Nodes of the line-graph L_G ($e \in E_G$ iff $v_e \in V_{L_G}$; $v_e v_{e'} \in E_{L_G}$ iff $e = uv, e' = uv'$) have sparsity $\Omega(\Delta)$. **In an edge-coloring setting, [EPS15]'s result gives $\Omega(\Delta)$ slack.**

Creating slack: sparsity



Intuition: if your neighbors are mostly disconnected, it's likely some of them will pick the same color.

Formally: $\zeta_v = \frac{1}{\Delta}\left(\binom{\Delta}{2} - |E[N(v)]|\right)$.

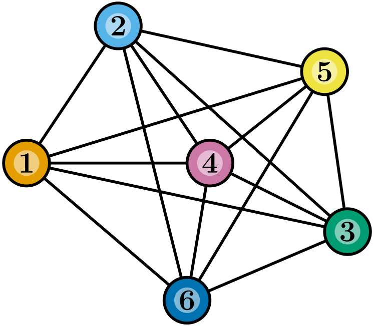
Theorem ([EPS15]): Let each node try a random color, then afterwards:

$$\Pr[s_v \leq c \cdot \zeta_v] \leq \exp(-c' \cdot \zeta_v).$$

for some universal constants c and c'.

Nodes of the line-graph L_G ($e \in E_G$ iff $v_e \in V_{L_G}$; $v_e v_{e'} \in E_{L_G}$ iff $e = uv, e' = uv'$) have sparsity $\Omega(\Delta)$. **In an edge-coloring setting, [EPS15]'s result gives $\Omega(\Delta)$ slack.**

Creating slack: sparsity



Intuition: if your neighbors are mostly disconnected, it's likely some of them will pick the same color.

Formally: $\zeta_v = \frac{1}{\Delta}\left(\binom{\Delta}{2} - |E[N(v)]|\right)$.

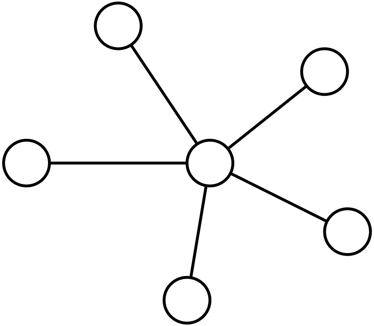
Theorem ([EPS15]): Let each node try a random color, then afterwards:

$$\Pr[s_v \le c \cdot \zeta_v] \le \exp(-c' \cdot \zeta_v).$$

for some universal constants c and c'.

Nodes of the line-graph L_G ($e \in E_G$ iff $v_e \in V_{L_G}$; $v_e v_{e'} \in E_{L_G}$ iff $e = uv, e' = uv'$) have sparsity $\Omega(\Delta)$. **In an edge-coloring setting, [EPS15]'s result gives $\Omega(\Delta)$ slack.**

Creating slack: sparsity



Intuition: if your neighbors are mostly disconnected, it's likely some of them will pick the same color.

Formally: $\zeta_v = \frac{1}{\Delta}\left(\binom{\Delta}{2} - |E[N(v)]|\right)$.

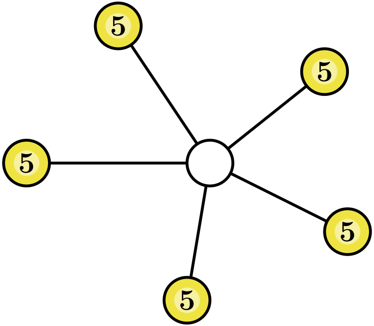
Theorem ([EPS15]): Let each node try a random color, then afterwards:

$$\Pr[s_v \leq c \cdot \zeta_v] \leq \exp(-c' \cdot \zeta_v).$$

for some universal constants c and c'.

Nodes of the line-graph L_G ($e \in E_G$ iff $v_e \in V_{L_G}$; $v_e v_{e'} \in E_{L_G}$ iff $e = uv, e' = uv'$) have sparsity $\Omega(\Delta)$. **In an edge-coloring setting, [EPS15]'s result gives $\Omega(\Delta)$ slack.**

Creating slack: sparsity



Intuition: if your neighbors are mostly disconnected, it's likely some of them will pick the same color.

Formally: $\zeta_v = \frac{1}{\Delta}\left(\binom{\Delta}{2} - |E[N(v)]|\right)$.

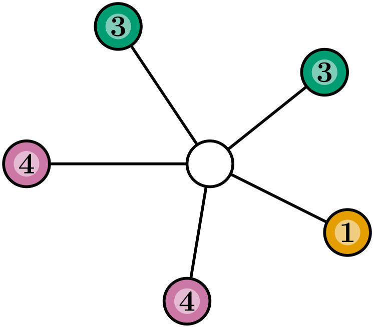
Theorem ([EPS15]): Let each node try a random color, then afterwards:

$$\Pr[s_v \leq c \cdot \zeta_v] \leq \exp(-c' \cdot \zeta_v).$$

for some universal constants c and c'.

Nodes of the line-graph L_G ($e \in E_G$ iff $v_e \in V_{L_G}$; $v_e v_{e'} \in E_{L_G}$ iff $e = uv, e' = uv'$) have sparsity $\Omega(\Delta)$. **In an edge-coloring setting, [EPS15]'s result gives $\Omega(\Delta)$ slack.**

Creating slack: sparsity



Intuition: if your neighbors are mostly disconnected, it's likely some of them will pick the same color.

Formally: $\zeta_v = \frac{1}{\Delta}\left(\binom{\Delta}{2} - |E[N(v)]|\right)$.

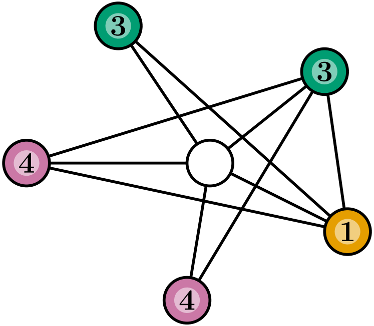
Theorem ([EPS15]): Let each node try a random color, then afterwards:

$$\Pr[s_v \leq c \cdot \zeta_v] \leq \exp(-c' \cdot \zeta_v).$$

for some universal constants c and c'.

Nodes of the line-graph L_G ($e \in E_G$ iff $v_e \in V_{L_G}$; $v_e v_{e'} \in E_{L_G}$ iff $e = uv, e' = uv'$) have sparsity $\Omega(\Delta)$. **In an edge-coloring setting, [EPS15]'s result gives $\Omega(\Delta)$ slack.**

Creating slack: sparsity



Intuition: if your neighbors are mostly disconnected, it's likely some of them will pick the same color.

Formally: $\zeta_v = \frac{1}{\Delta}\left(\binom{\Delta}{2} - |E[N(v)]|\right)$.

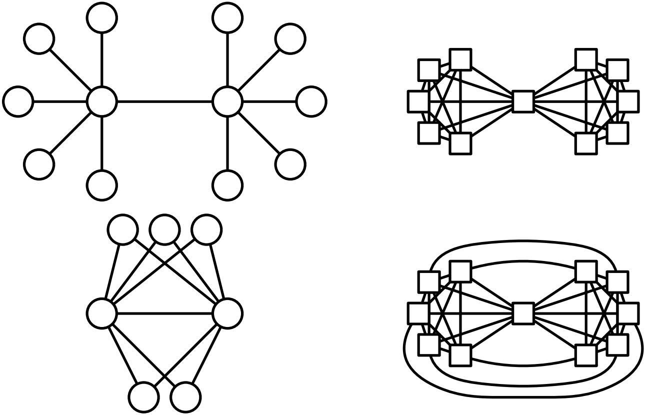
Theorem ([EPS15]): Let each node try a random color, then afterwards:

$$\Pr[s_v \le c \cdot \zeta_v] \le \exp(-c' \cdot \zeta_v).$$

for some universal constants c and c'.

Nodes of the line-graph L_G ($e \in E_G$ iff $v_e \in V_{L_G}$; $v_e v_{e'} \in E_{L_G}$ iff $e = uv, e' = uv'$) have sparsity $\Omega(\Delta)$. **In an edge-coloring setting, [EPS15]'s result gives $\Omega(\Delta)$ slack.**

Inherent sparsity of line-graphs



Pairwise-Independent Hash Functions

Definition ((ϵ-almost) Pairwise-Independent Hash Functions)

A set $\mathcal{H}$ of functions $h : [N] \to [M]$ s.t.:
$\forall x \neq y \in [N]^2, z, z' \in [M]^2$,
$|\Pr_{h \in \mathcal{H}}[h(x) = z \wedge h(y) = z'] - 1/M^2| \leq \epsilon/M^2$.

Let p be a prime such that $p \geq M^3 \cdot \log_p N/\epsilon$.

Decompose x into basis p, i.e., compute $x_1, \ldots, x_{\lceil \log_p N \rceil}$ such that $x = \sum_i x_i \cdot p^i$.

Consider the polynomial $P_x(X) = \sum_i x_i \cdot X^i$, of degree $\leq \log_p N$.

Pick a, b, z uniformly at random in $\{0, \ldots, p-1\}$.

Compute: $h_{a,b,z} = ((a \cdot P_x(z) + b) \mod p) \mod M$.

Finite fields of size p^k, p prime

Example with $p = 2$.

Pick a degree k polynomial irreducible over $\mathbb{F}_2$. For $p = 2$, $P(X) = X^k + X + 1$ always works: $P(0) = P(1) = 1 \mod 2$.

Consider the $\mathbb{F}_2[X]/P(X)$, i.e., polynomials over $\mathbb{F}_2$ mod P.

It is equivalent to defining that there's an element a s.t. $a^k = 1 + a$, and considering the 2^k possible words that can be written as combinations of $1, a, \ldots, a^{k-1}$.

Example of a multiplication over this field: let $k = 3$, $(1 + a^2) \times a = a + a^3 = a + 1 + a = 1$.