Sublinear expected time coloring algorithms

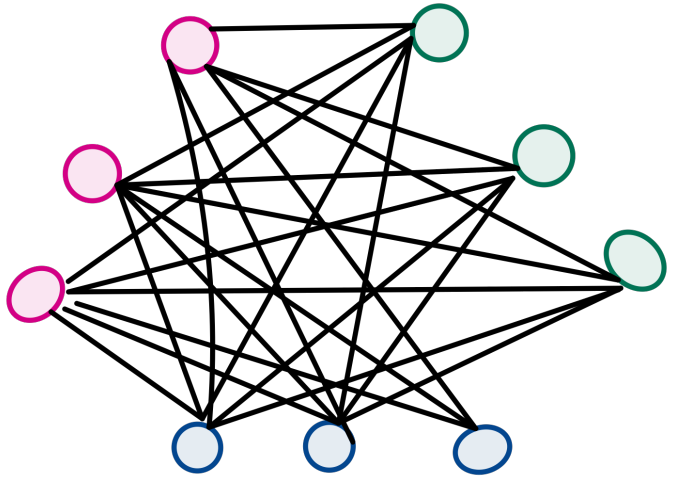
Ronitt Rubinfeld

MIT

with Cassandra Marcussen (Harvard), Ted Pyne (MIT), Asaf Shapira (TAU), Shlomo Tauber (TAU)

k-coloring

k-coloring: color vertices of graph with k colors such that no adjacent vertices share color.



NP-hard for $k \geq 3$

Hard to approximate to factor of $n^{1-\epsilon}$

Average-case k-coloring

$U =$ uniform distribution over k −colorable graphs

Expected time to color graphs from U? [Dyer-Frieze 89]

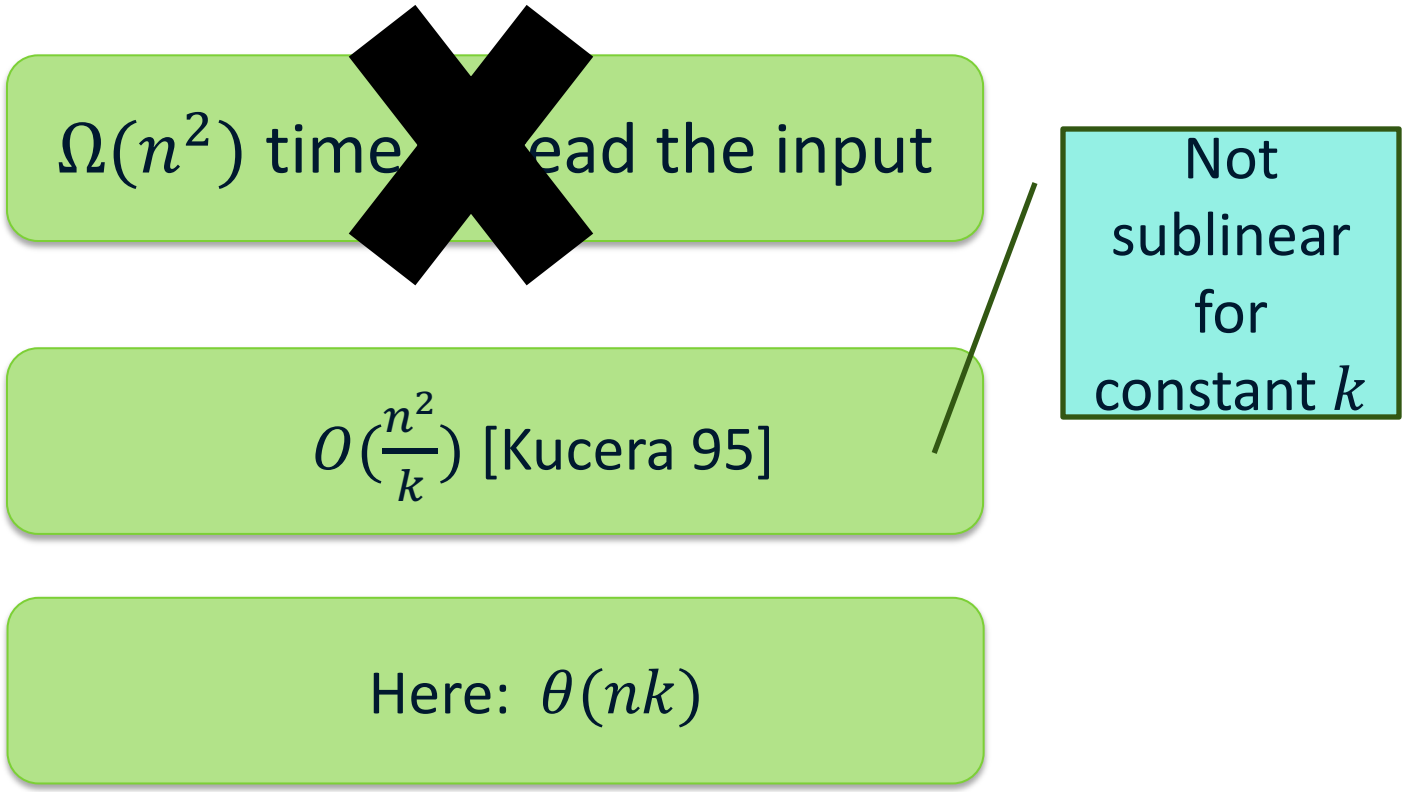
Desired Algorithm:

1. k-colors any k-colorable graph correctly
2. Efficient on average

Most k-colorable graphs can be quickly colored

- Polynomial time for all but small fraction of graphs (not enough for average case bounds)
  - Sparse k-colorable graphs, structure of efficiently k-colorable graphs, deciding k-colorability,… (e.g. [Alon Kahale 94] [Kucera 77,93] [Krivelevich 02] …)
- Polynomial expected time e.g.,
  - $O(n^2)$ expected time, constant k [Dyer Frieze 89]
  - $O\left(\frac{n^2}{k}\right)$ expected time, $k = o(\sqrt{\frac{n}{\log^2(n)}}$ [Kucera 95]

Possible lower bound?



A challenge

Typical approach:

- Find fast algorithm for large class of inputs
- Run slow algorithm on others

Total runtime: $(1-\alpha)\cdot T_{fast} + \alpha \cdot T_{slow}$

Need $\alpha \ll 1/T_{slow}$

e.g., If T_{slow} is 2^n, then $\alpha \ll 2^{-n}$

Models of random graphs

Distribution on k-colorable graphs

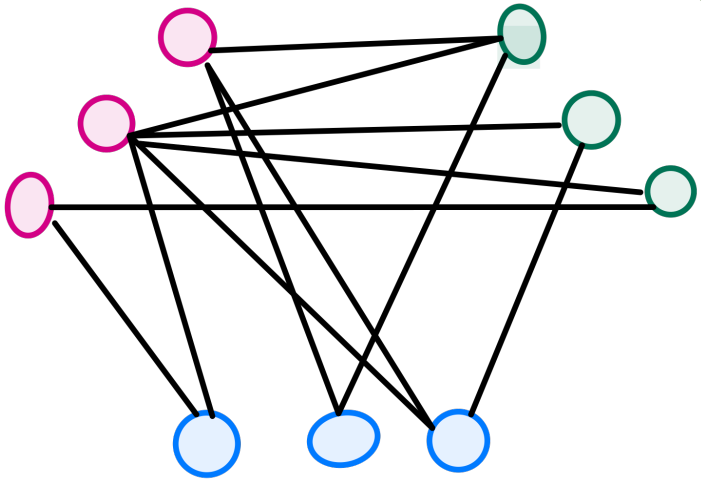
MODEL 1 (uniform k-colorable graph):

Pick uniformly from k-colorable graphs on n vtcs

Distribution on k-colorable graphs

MODEL 2 (planted k-coloring):

- Equipartition nodes into k sets $X_1, X_2 \ldots, X_k$
- For each (u, v) such that $u \in X_i, v \in X_j$ and $i \neq j$
  - Flip coin with bias p to determine if $(u, v) \in E$



p is constant

Distribution on k-colorable graphs

MODEL 1 (uniform k-colorable graph):

Pick uniformly from k-colorable graphs on n vtcs

MODEL 2 (planted k-coloring):

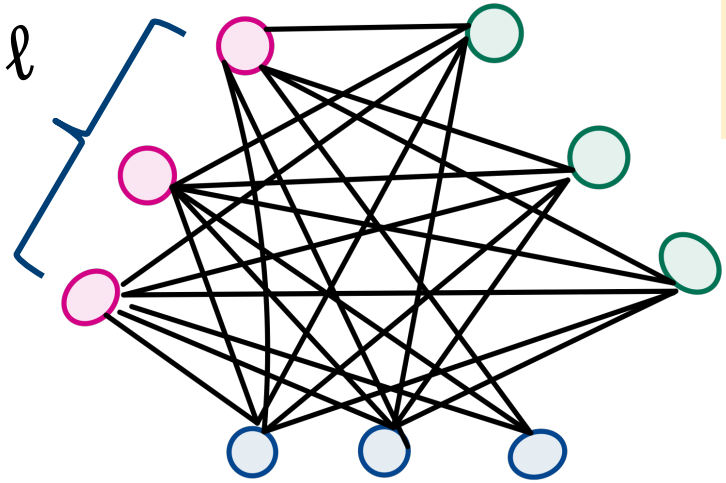
- Equipartition nodes into k sets $X_1, X_2 \ldots, X_k$
- For each (u, v) such that $u \in X_i, v \in X_j$ and $i \neq j$
  - Flip coin with bias p to determine if $(u, v) \in E$

Claim: For designing sublinear time average case algorithms, definitions are equivalent
(similar to Dyer-Frieze)

Warmup: constant k

Special "core" substructure: Complete k-partite graph

- Core $\mathcal{C}_{k,\ell}(\mathrm{V},\mathrm{E})$:
  - $V = V_1 \cup V_2 \ldots \cup V_k \; s.t. \; |V_i| \approx \ell$, where $\ell = O(\log k)$
  - $E = \{(u,v) | \, u \in V_i, v \in V_k \; i \neq k\}$

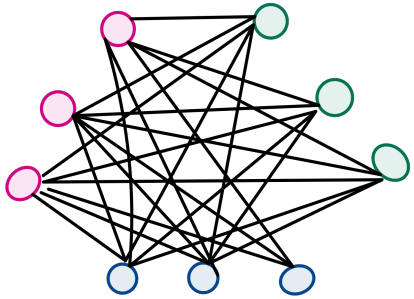


Claim: For any given $k \cdot \ell$ vertices, the probability they are a $\mathcal{C}_{k,\ell}$ is $p^{O(k^2\ell^2)}$

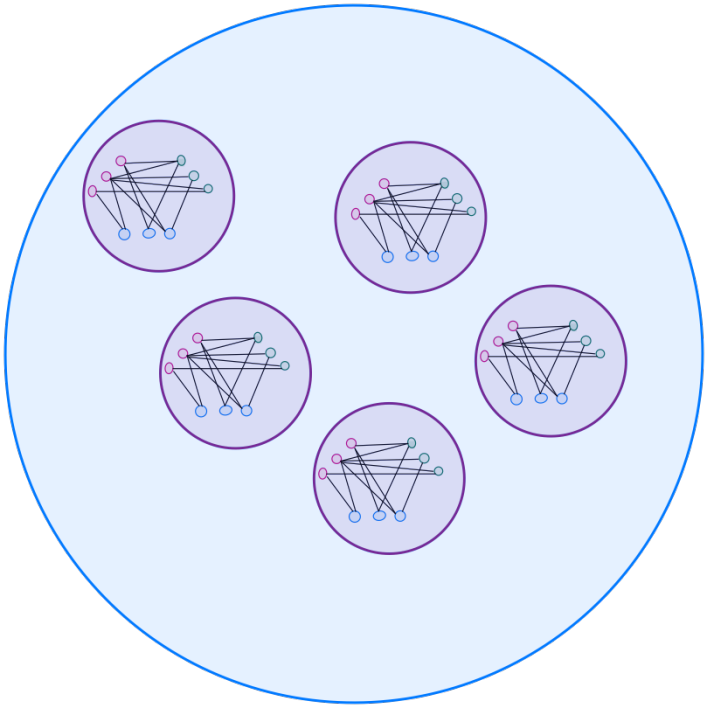
this is constant!

$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

Nice properties of core $\mathcal{C}_{k,\ell}$

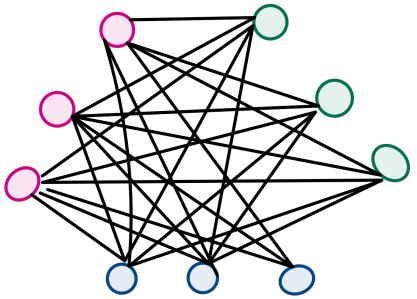


1. There are lots of them
   - Can find in time $O(p^{-O(k^2\ell^2)})$



$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

Nice properties of core $\mathcal{C}_{k,\ell}$



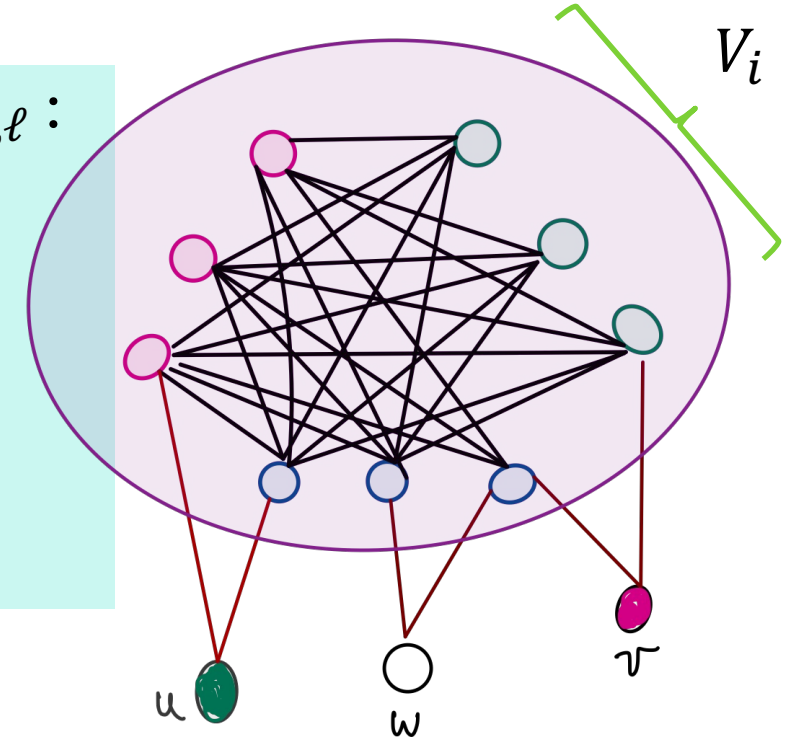
1. There are lots of them
2. There is only one way to color it
   - Due to completeness of $\mathcal{C}_{k,\ell}$

$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

First-level-coloring

First-level-coloring-procedure given $\mathcal{C}_{k,\ell}$:

- For $u \in V$:
  - Check all adjacencies to $\mathcal{C}_{k,\ell}$
  - If has edge to $k-1$ colors in $\mathcal{C}_{k,\ell}$ color u with remaining color
  - Else, u remains uncolored



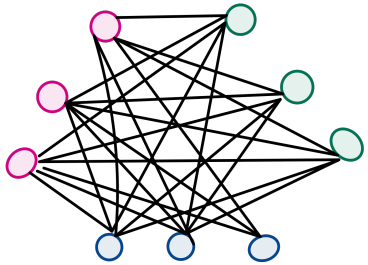
Runtime for u: $O(k\ell) = O(1)$

Claim: $> \frac{n}{2}$ vertices colorable by first-level-coloring-procedure

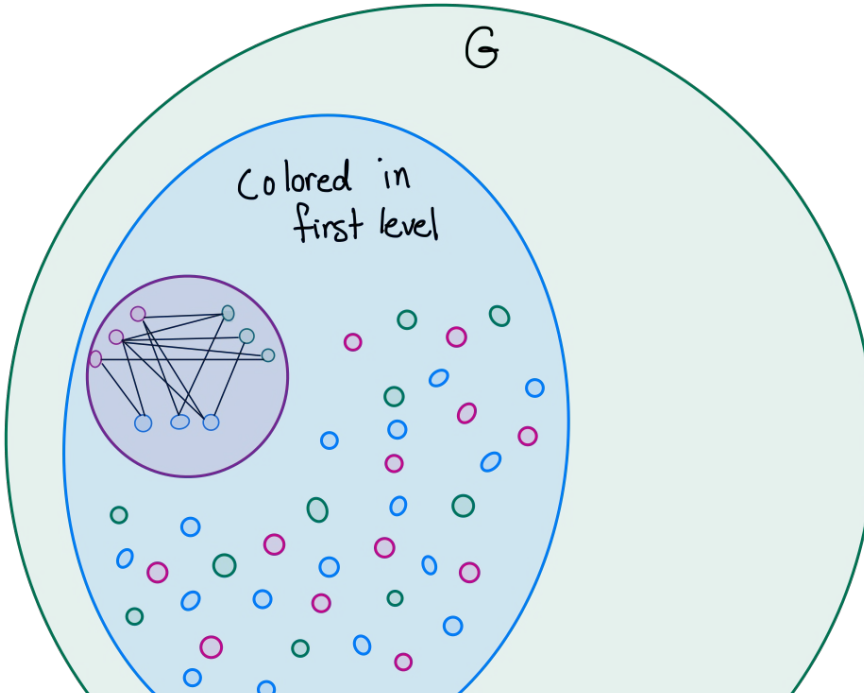
Why? Pick ℓ large enough

$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

Nice properties of core $\mathcal{C}_{k,\ell}$



1. There are lots of them
2. There is only one way to color it
3. Its coloring immediately determines colors of > ½ of vertices $\mathcal{X} \subseteq V$

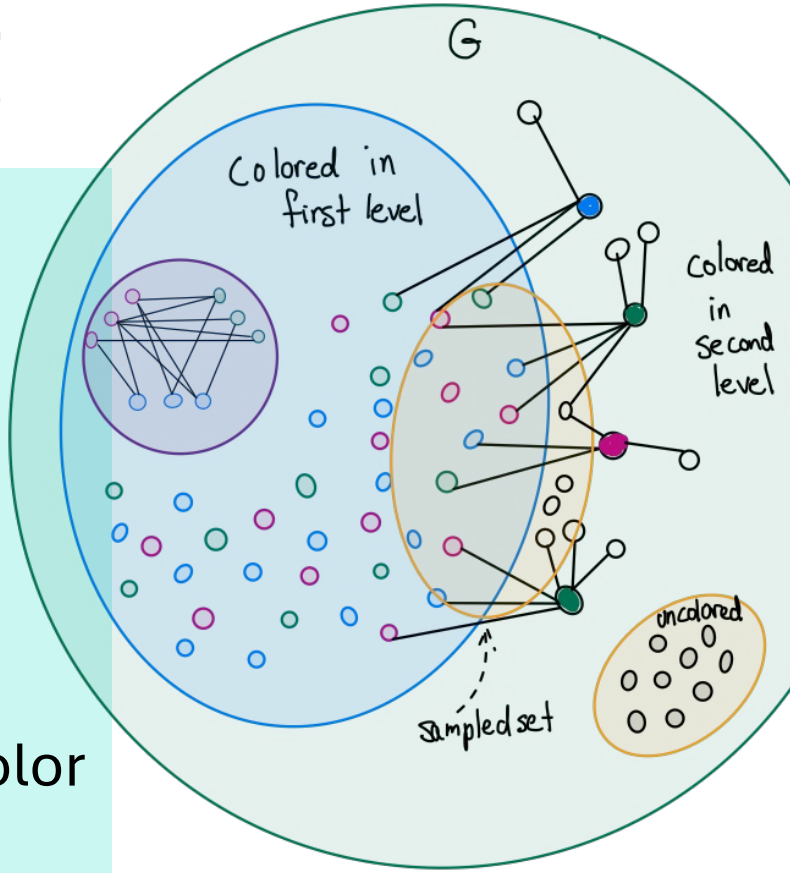


$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

Second-level-coloring

Second-level-coloring-procedure:

- Sample $O\big(k\log(k)\big)$ random nodes
  - For each sampled v
    - run first-level-coloring on v
- For $u \in V$:
  - If u has edge to $k-1$ colors in sample, color u with remaining color
  - Else, u remains uncolored

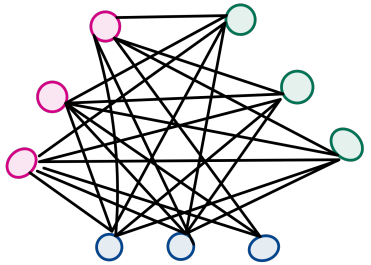


Runtime for u: $O(k\log k) = O(1)$

Claim: usually *all* vertices colored by second-level-coloring, whp at most $O(k\log k)$ remain

$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

Algorithm:



1. Randomly sample nodes until find $\mathcal{C}_{k,\ell}$ copy
   - color it
2. For all u, attempt to color via First-level coloring procedure (determines colors of $> \frac{1}{2}$ of V)
3. For all remaining u , attempt to color via Second-level coloring procedure (determines colors of *all but* $< O(k \log k)$ *nodes* of V)
4. Brute force color the last $<O(k \log k)$ nodes

Both are constant time per node!

Total runtime: $O\left(n \cdot poly(k) + nk^{O(\mathrm{k} \log k)}\right) = O(n)$

$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

Challenges for nonconstant k

For each u, likely to connect to at least one vertex of each color

- Searching for $\mathcal{C}_{k,\ell}$ takes time exponential in k, ℓ
  - only need subgraph with unique, "balanced" coloring!

How to find? Unique?

- Aiming for $O(n \cdot k)$
  - $poly(k)$ implementation to find color of u not fast enough
- $O\left(n \cdot k^{O(k \log k)}\right)$ brute force term

New idea: less restrictive core

Some properties needed from core:

1. Uniquely colorable
2. Many vertices of each color

How do we find core?

Pick random set of size $O(poly(k))$.

How do we color?

Use [Kucera 95] fast expected time algorithm
(if too slow, sample again)

Big win:
$O(1)$ tries

Problem: What if not unique?

Can we detect and sample again?

Certify uniqueness without checking every edge?

Certifying unique colorability

Behavior of certification algorithm on input C:

- Most uniquely colorable graphs certified
- If certify C, then C uniquely colorable

If fails, try new C'
Only do this $O(1)$ times

Huh?

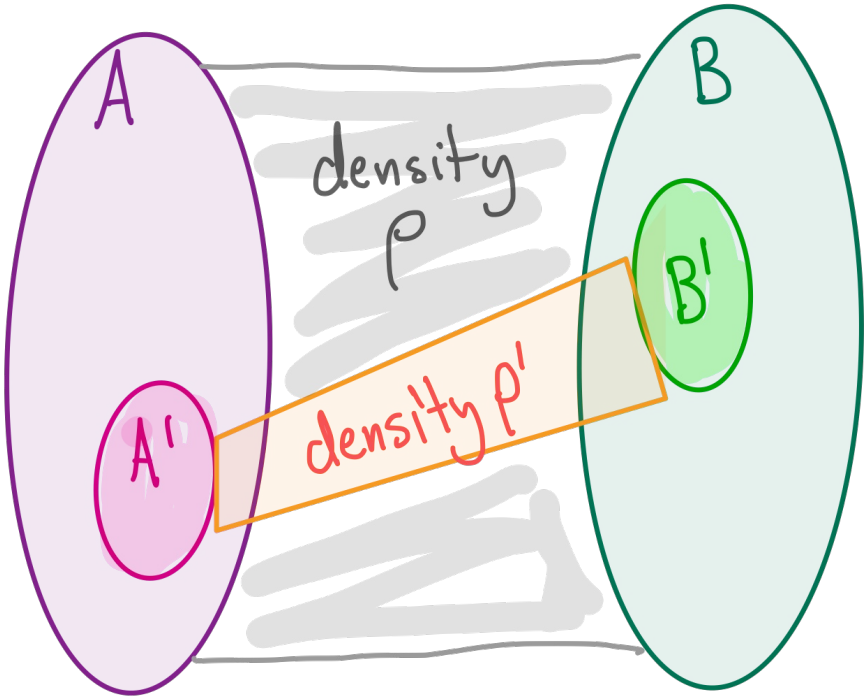
Certification algorithm for subgraph C:

Certify "graph regularity" for every pair of partitions

Regularity

(A, B) is ϵ-regular if for all $A' \subset A, B' \subset B, s.t. |A'| \geq \epsilon|A|$, $|B'| \geq \epsilon|B|$, then $\left|\frac{e(A',B')}{|A'|\cdot|B'|} - \frac{e(A,B)}{|A|\cdot|B|}\right| \leq \epsilon$

random graphs (usually) satisfy this

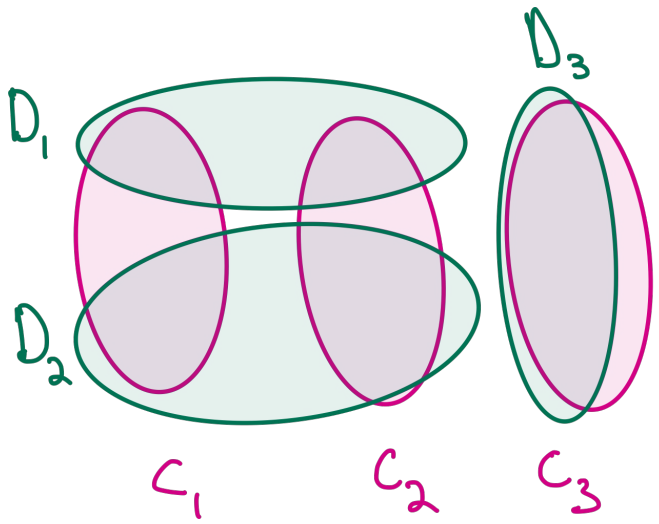


$\rho \approx \rho'$

Regularity is certifiable!
[Alon Duke Lefmann Rodl Yuster]
If degrees, codegrees near expectation, then regular

Regularity implies unique colorability

- Suppose two proper k-colorings $(C_1, \dots, C_k), (D_1, \dots, D_k)$



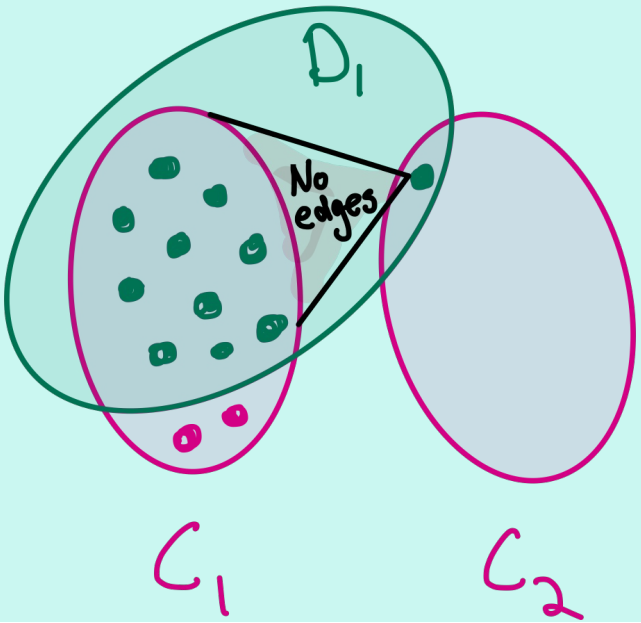
- C's are certified, but D's ???
- Claim: Exists j, j', i s.t. $|C_j \cap D_i|/|C_j| \gtrsim 1/k$ and $|C_{j'} \cap D_i| \geq 1$

D_1 and C_1 have reasonable overlap, D_1 overlaps C_2

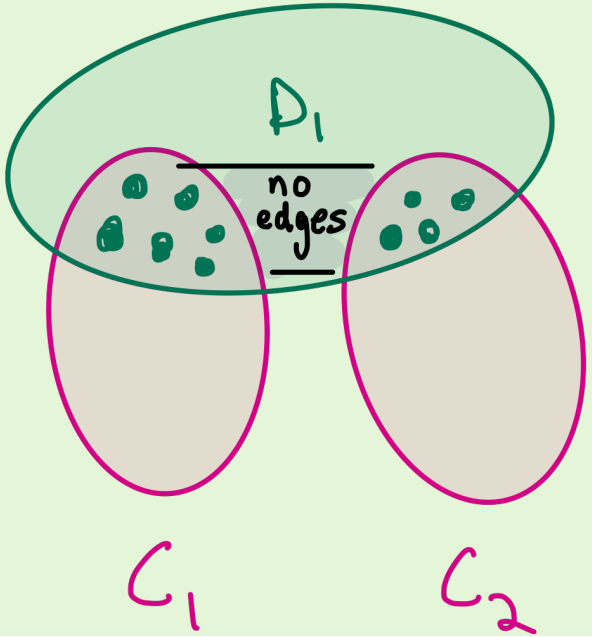
Wlog C_1, C_2, D_1

Regularity implies unique colorability

- $|C_1 \cap D_1|$ really big : Violates "good core" degree requirement



- $|C_1 \cap D_1|$ not really big, but not too small: $|C_2 \cap D_1|$ is not small. Violates regularity



Certifying unique colorability

Behavior of certification algorithm on input C:

- Most uniquely colorable graphs certified
- If certify C, then C uniquely colorable

If fails, try new C'
Only do this $O(1)$ times

Certification algorithm for subgraph C: "Certify graph regularity"

- For all u, test degree to each partition in C close to expectation
- For all pairs u, v, test $Codegree(u, v)$ =number of common neighbors in each partition, close to expectation

Only C with "random-looking" properties pass these!

$s \equiv |C| = O(poly(k))$

Summary: less restrictive core

Some properties needed from core:

1. Uniquely colorable
2. Many vertices of each color
3. Degrees and codegrees within core partitions ”look right”

How do we find core?

Pick random set of size $O(poly(k))$. WHP has properties.

How do we color?

Use [Kucera 95] -- if too slow, or degrees/codegrees not nice, sample again

$\mathcal{C}_{k,\ell}$: Complete k −partite graph, ℓ nodes per part

Challenges for nonconstant k

- Searching for $\mathcal{C}_{k,\ell}$ takes time exponential in k, ℓ
  - only need subgraph with unique and "balanced" coloring!
- Aiming for $O(n \cdot k)$
  - $poly(k)$ implementation to find color of u not fast enough
  - Sample from core to determine color of u: $O(k)$ suffices.
- $O\left(nk^{k \log k}\right)$ is too big
  - Rarely need to do brute force.

Derandomization

- Where did we use randomness?
  - Finding core
  - Coloring vertices in level 1
    - Based on subset of core
    - Randomization only saves $\log k$ factor runtime
  - Coloring vertices in level 2

$\tilde{O}(nk)$

Analyze over equivalence classes of isomorphic graphs via symmetry properties

Lower bound

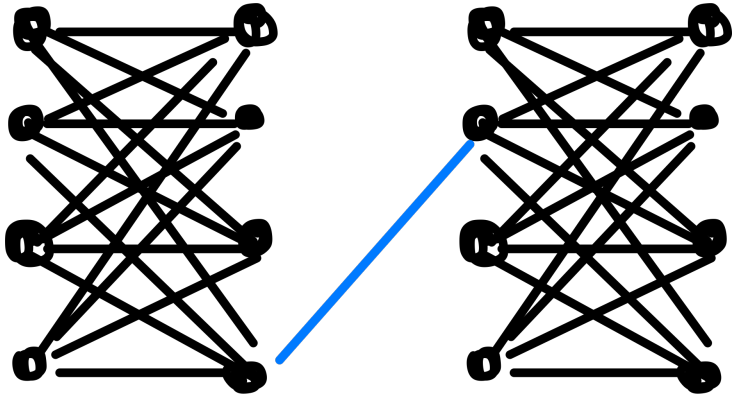
- Any k-coloring algorithm makes $\Omega(nk)$ queries on any input graph (implies average case $\Omega(nk)$ lower bound)

Locality

- Def. Local computation algorithm (LCA) for coloring:
  - Answer query "what is color of u?"
  - Local memory wiped between queries
  - Answers must be consistent with previous, simultaneous, future queries
  - Average case runtime is expected (over graphs) of maximum (over u) time to answer query.

Locality

Worst case 2-coloring needs a lot of queries!



Locality

- Theorem: there is an LCA that answers k-color queries to any given node in average case $O(k^9)$ time
  - Algorithm given query u:
    - Find core, and sample as in deterministic algorithm
    - Try to color u via first-level-coloring
    - Else try to color u via second-level-coloring
    - Else, take a lot of time.

Our results for k-coloring

- Randomized k-coloring with average runtime $O(nk)$ for $k \leq n^{\frac{1}{37}}$
- Deterministic k-coloring with average runtime $\tilde{O}(nk)$ for $k \leq n^{\frac{1}{37}}$
- For $\mathrm{k} \leq n^{\frac{1}{3}}$, any algorithm needs average runtime $\Omega(nk)$
- For $\mathrm{k} \leq n^{1/37}$, LCA answers coloring queries in $\tilde{O}(k^9)$ probes and time $\tilde{O}(k^{26})$

Open questions on k-coloring

- $k \geq n^{\frac{1}{37}}$?
  - Current limitation comes from size of core needed to certify
- Deterministic average runtime $O(nk)$?

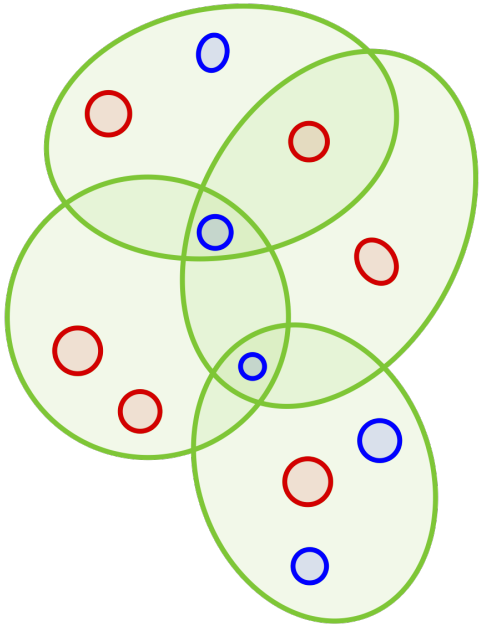
Hypergraph 2-coloring

[Marcussen Pyne R. Shapira Tauber ‘25]

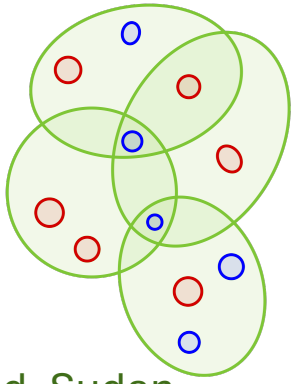
2-coloring hypergraphs

- r-uniform hypergraph: all hyper-edges of size r

- 2-coloring of hypergraph: each edge has at least one blue and one red node.
- Bipartite hypergraph: has 2-coloring
- Let $\mathcal{H}$ = set of r − uniform bipartite hypergraphs



Can we 2-color bipartite hypergraphs?



- NP-hard, even with extra colors [Lovasz '73],[Guruswami, Hastad, Sudan '02]
- Lots of algorithms… [Czumaj Scheideler '00a,b], [Erdos-Lovasz], [Radhakrishnan Srinivasan 98], [Lu 98], [Alon Kelsen Mahajan Ramesh 96], [Chen Frieze 96], [Krivelevich Nathaniel Sudakov 2001], [Krivelevich Sudakov 2003]…
  - number of colors depends on size of hypergraph
- $O(n^r)$ expected time 2-coloring algorithm for $H \in_U \mathcal{H}$ [Person Schacht '11] [Lee Molla Nagle '24]
  - Uses Szemeredi Regularity Lemma
- Here:
  - Randomized 2-coloring algorithm in average case time O(n)
    - without SRL - no hidden tower-type constant

Algorithm

- Similar outline but somewhat simpler:
  - Core = constant size complete bipartite hypergraph
  - 2-coloring vs. k-coloring
  - Random hypergraphs have stronger connectivity properties

Average case time LCAs on graphs

[Biswas Cao Marcussen Pyne R. Shapira Tauber]

- In Erdos Renyi and preferential attachment graphs, provably faster than worst case for consistently answering:
  - Spanners – "is edge (u, v) in the spanner?"
  - Sparse connected subgraphs – "is edge (u, v) in the sparse connected subgraph?"
  - Vertex cover – "Is node v in the small vertex cover?"

Open

- Other problems/distributions where average case algorithms provably beat worst case?
- Role of sublinear analysis in designing average case algorithms

Thank you!