Planar Partition Oracles in poly(1/ε) time

- July 31, Simons Institute
- Workshop on Sublinear Graph Simplification
- Akash Kumar (IIT Bombay)

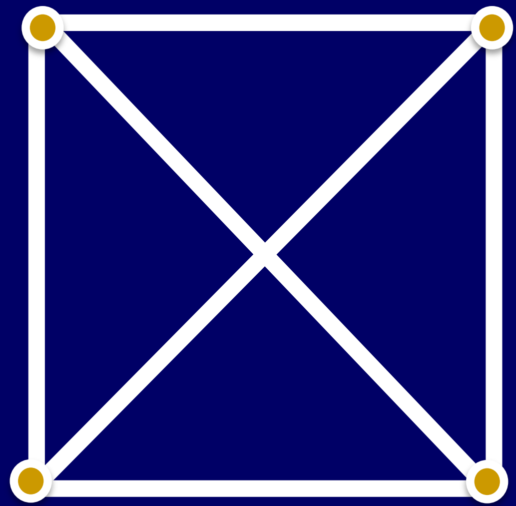
C. Seshadhri

(UCSC)

Andrew Stolman

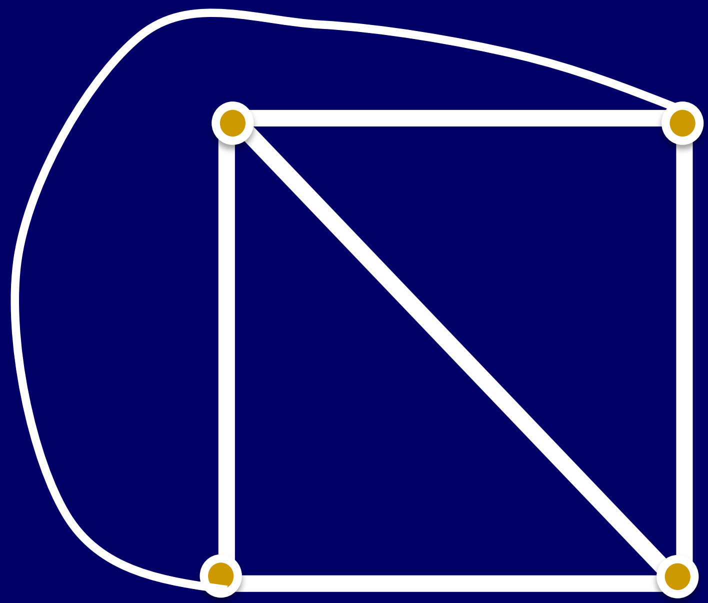
(Katana Labs)

What are planar graphs?

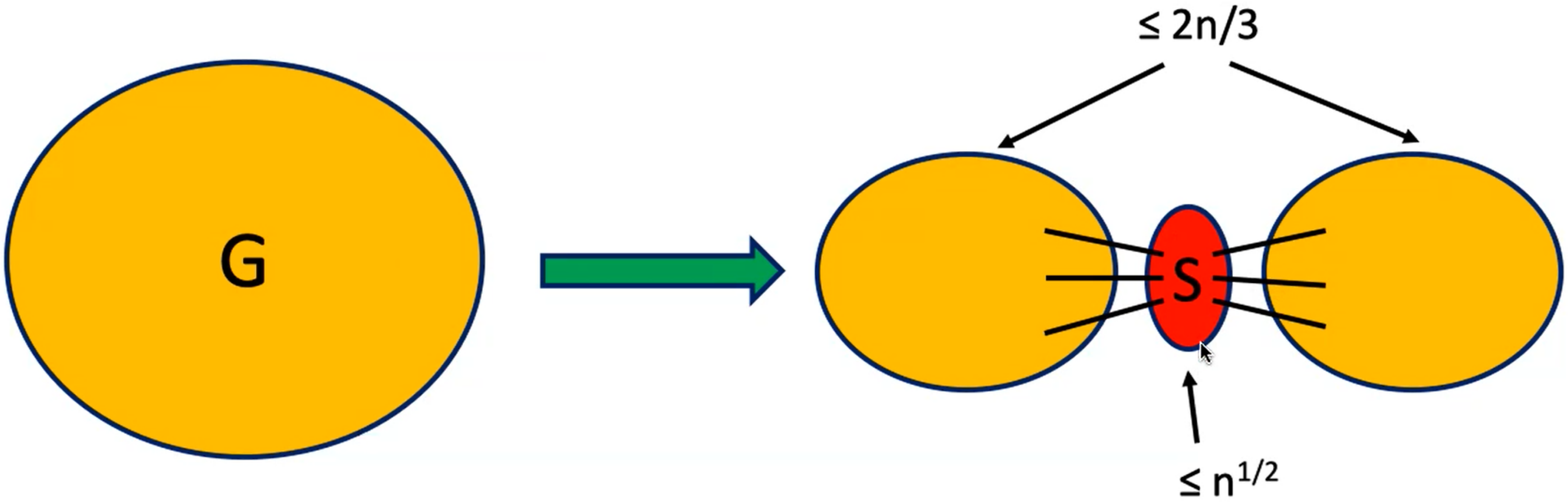


K_4

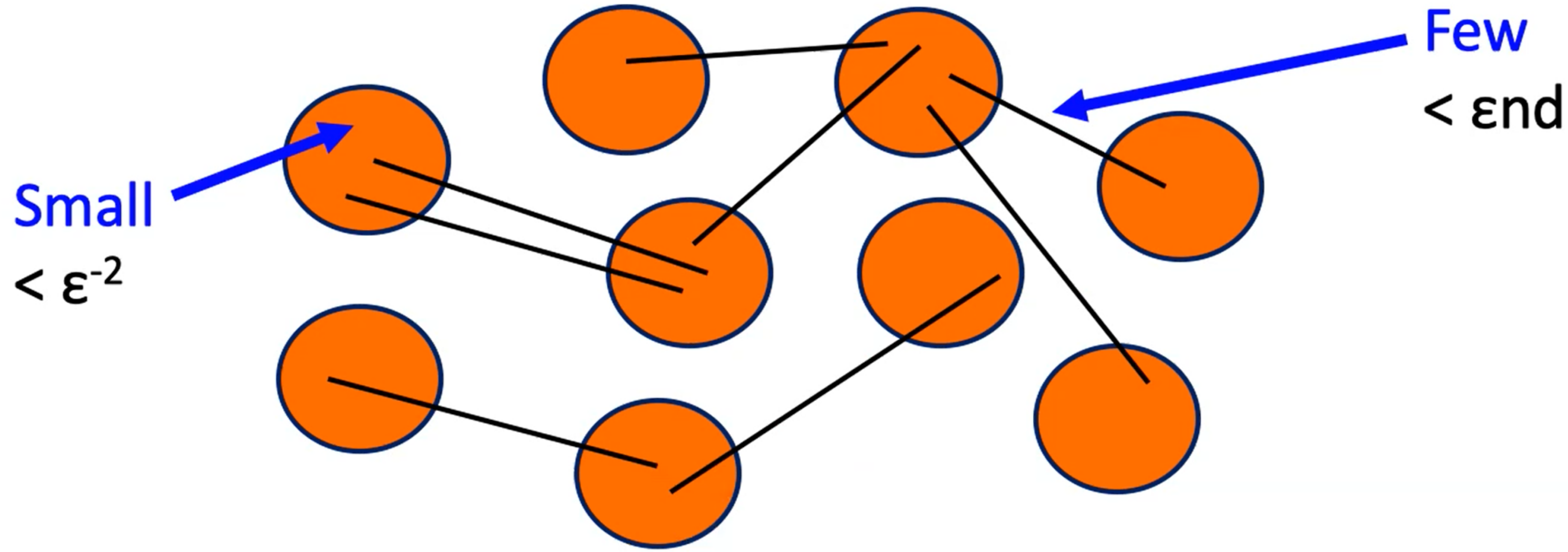
What are planar graphs?



K_4



- Balanced Separator Theorem
  - [Lipton-Tarjan 70]
  - Facilitates Divide and Conquer Algorithms



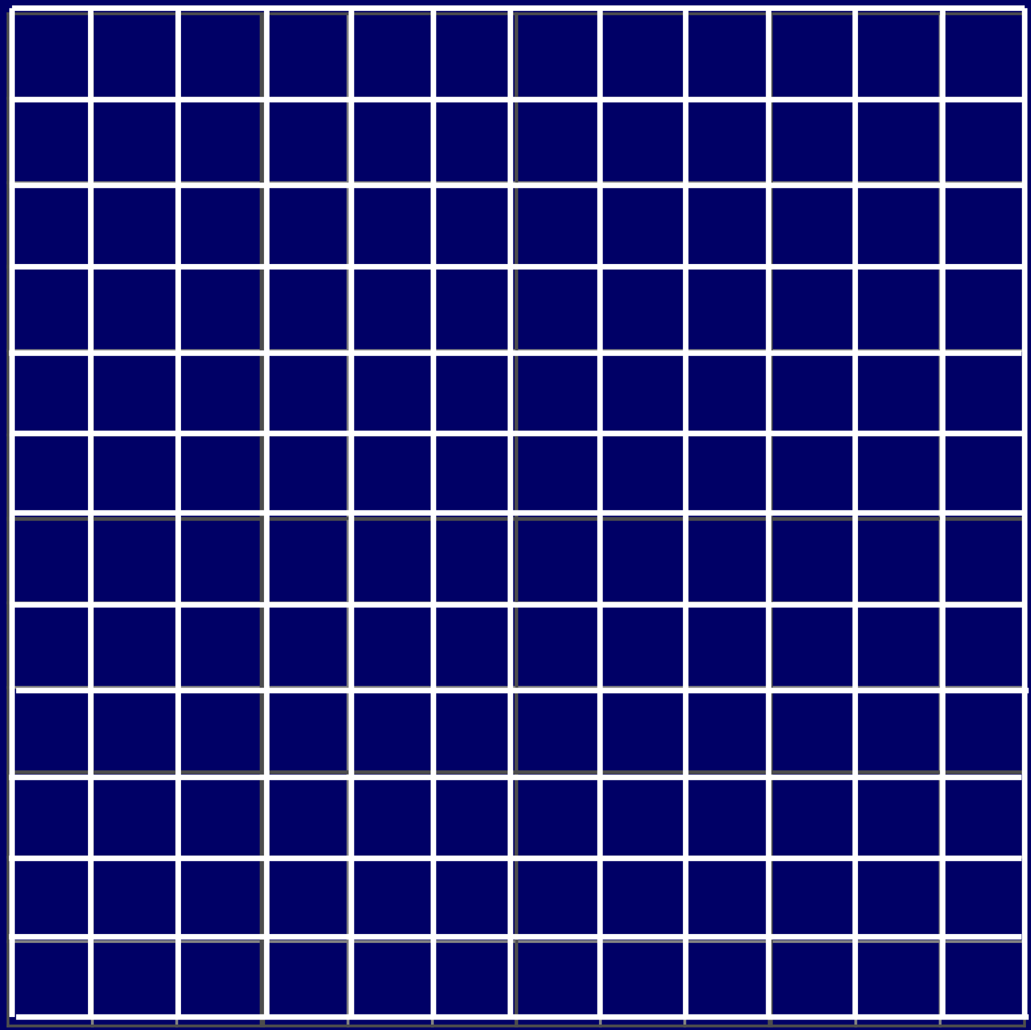
Hyperfinite Decompositions: Recursively use Planar Sep Theorem

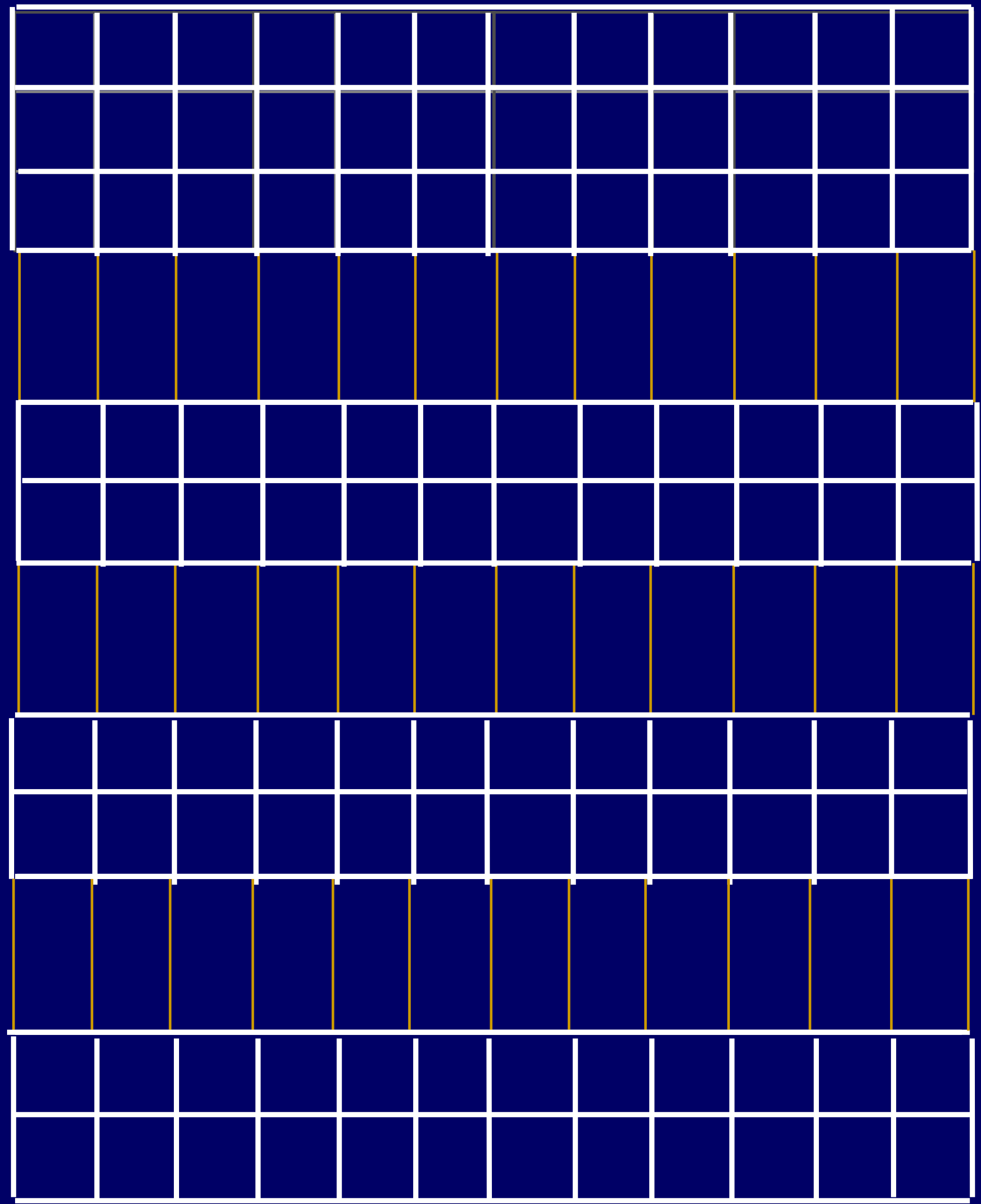
- [Elek 08, BSS 08, AST 94]
- Intuitively, hyperfinite decomposition enables approximation algos for various graph parameters.

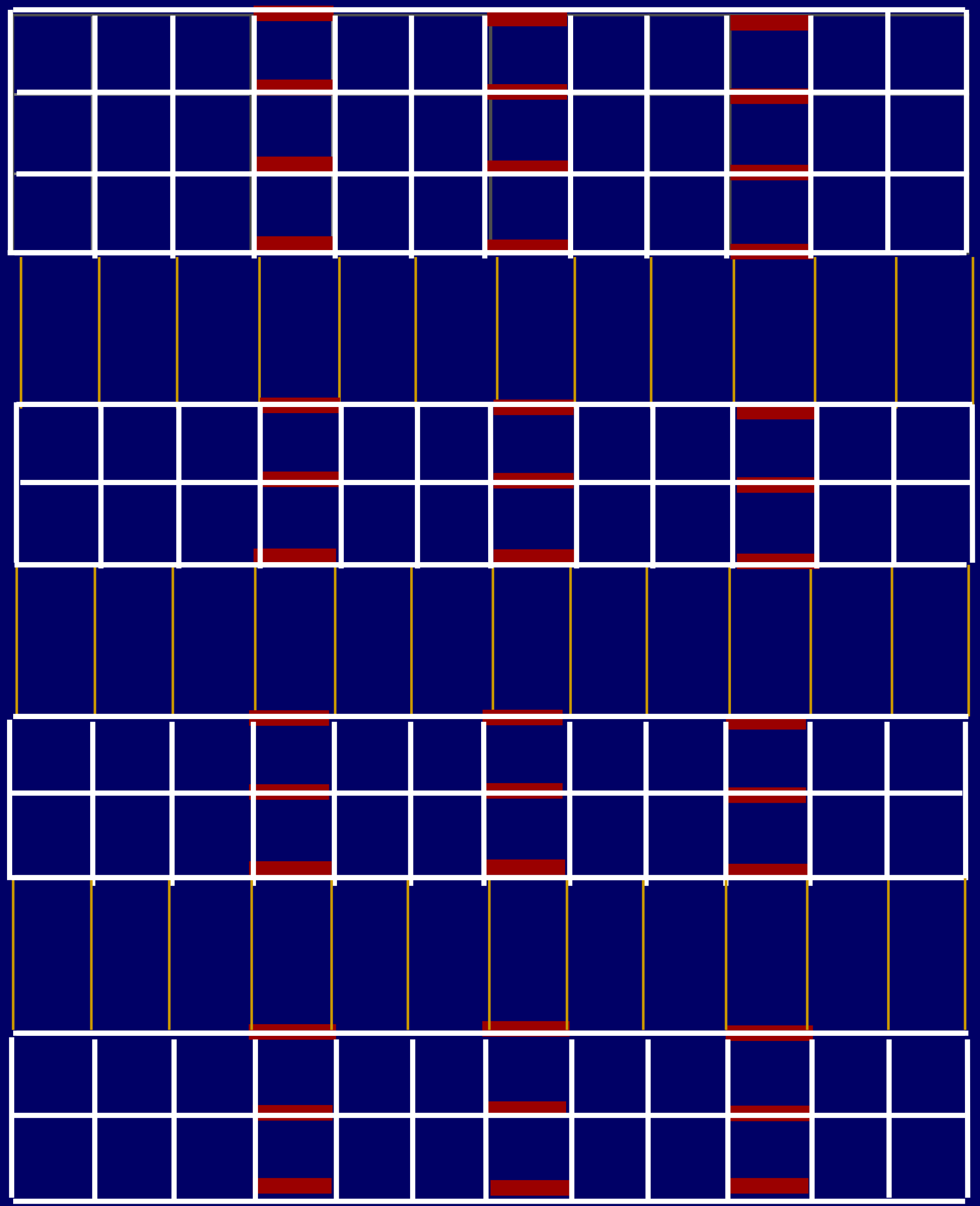
To reiterate:

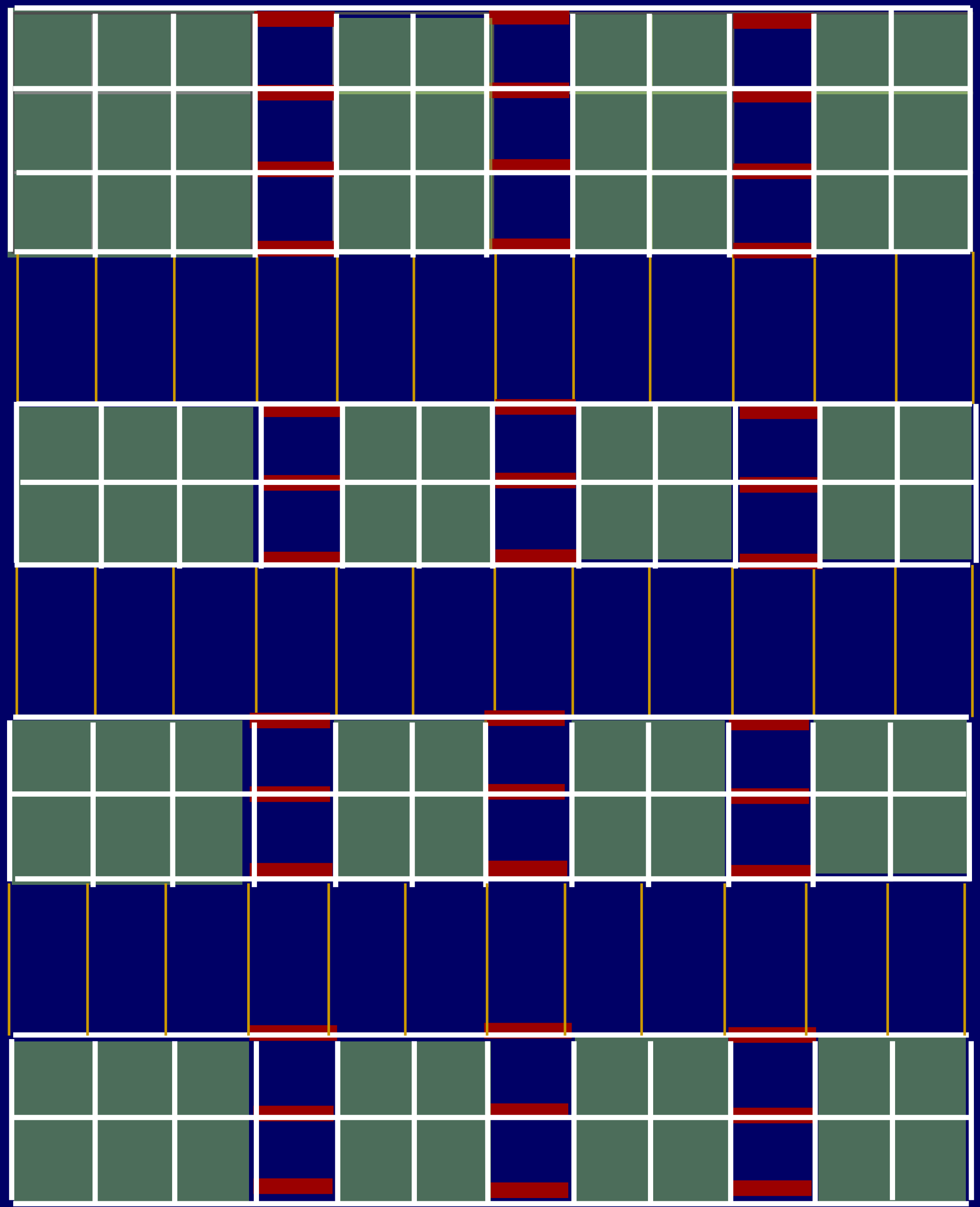
Bdd degree planar graphs are (ε,k)-Hyperfinite *with k = O(1/ε)²*

Can break V(G) into subsets of size $O(1/\varepsilon)^2$ by deleting only an ε-fraction of edges.





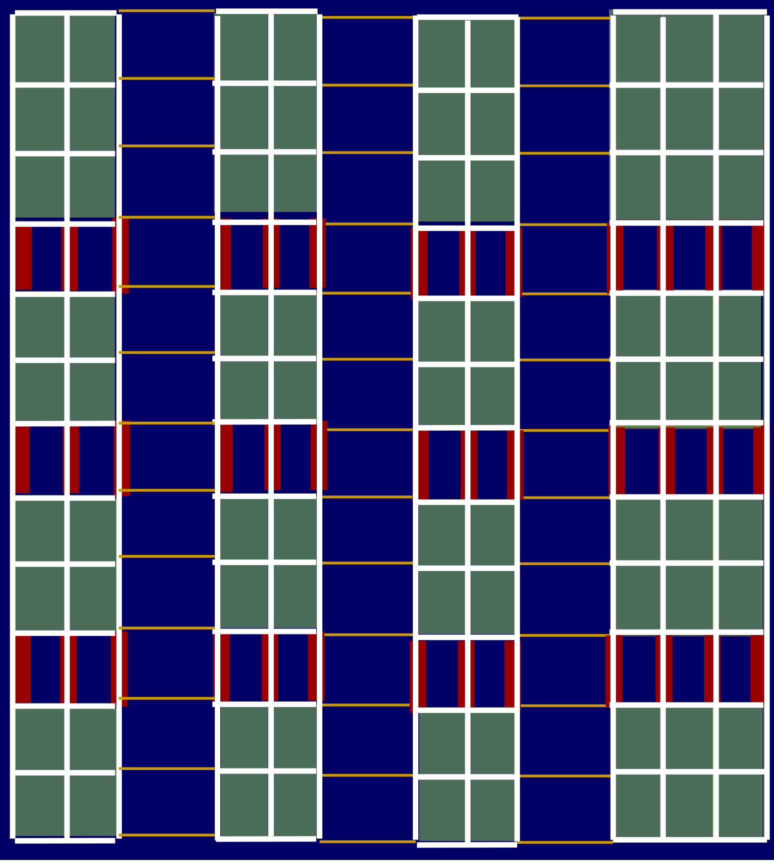




Goal: Assuming adjacency list query model

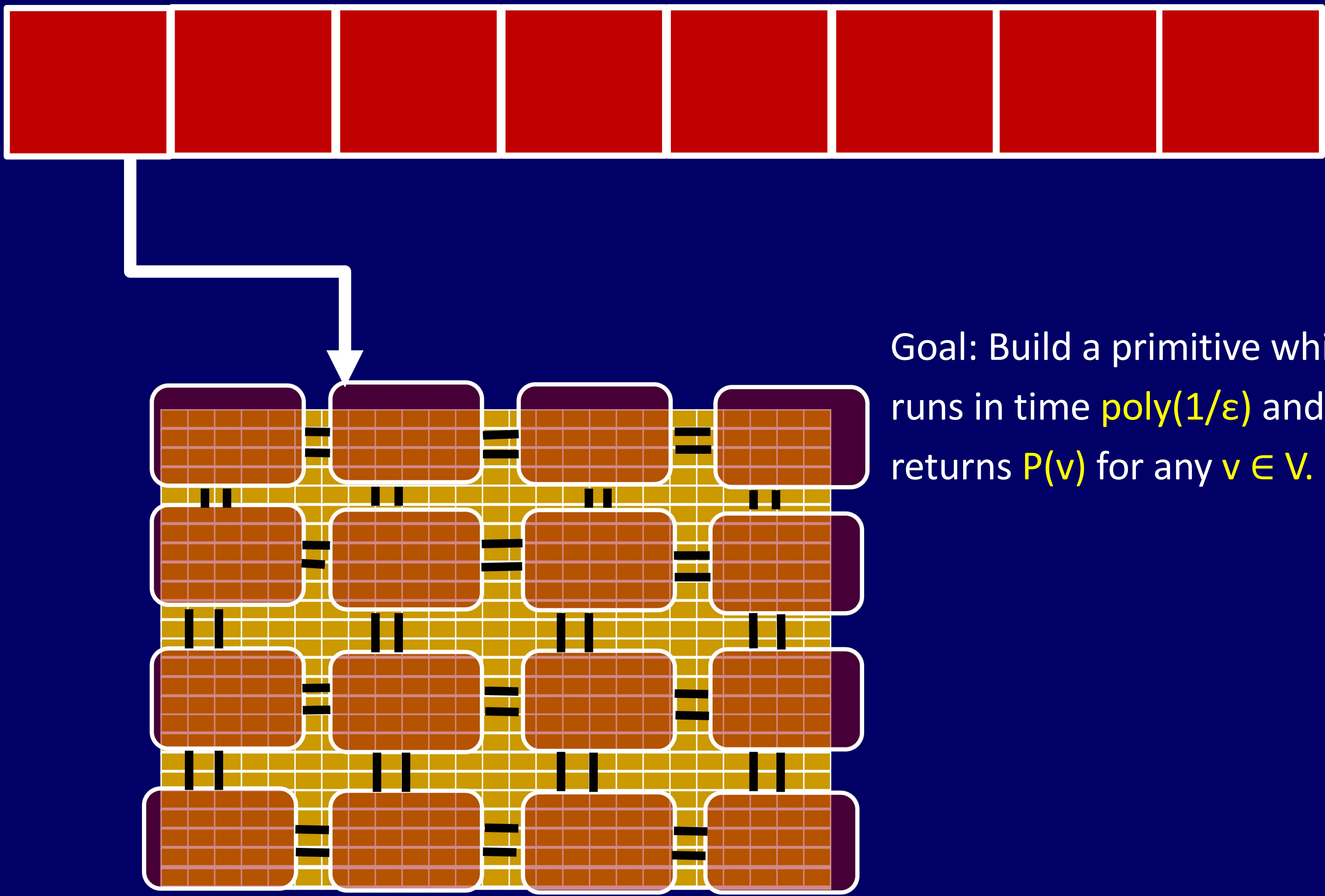
Build an Oracle which returns the component a vertex v belongs to with respect to *some hyperfinite* decomposition.

Annoyance: Multiple hyperfinite decompositions



Goal: A little more detailed

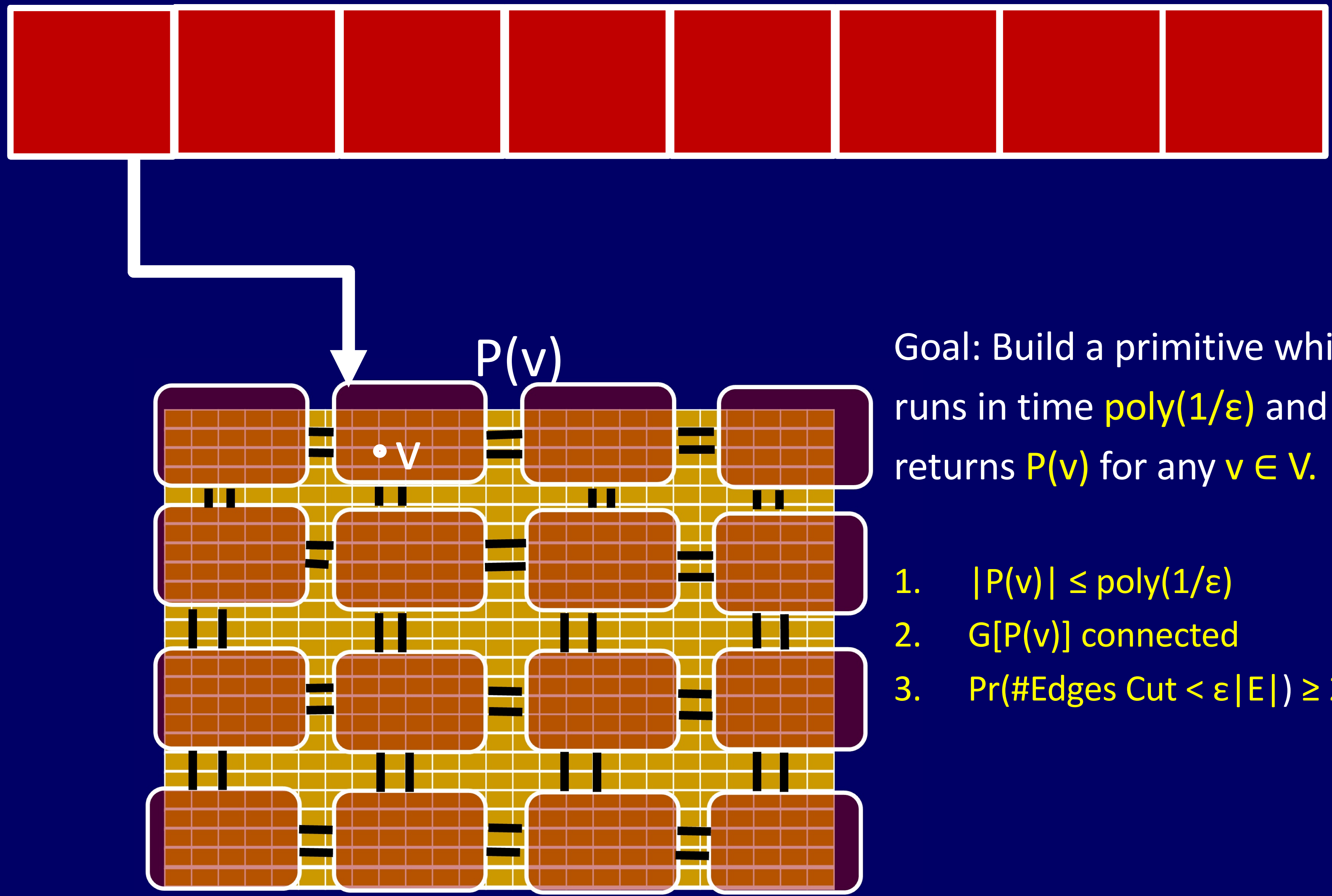
(Read only random tape)



Goal: Build a primitive which runs in time poly(1/ε) and returns P(v) for any v ∈ V.

Goal: A little more detailed

(Read only random tape)



Goal: Build a primitive which runs in time poly($1/\varepsilon$) and returns $P(v)$ for any $v \in V$.

1. $|P(v)| \le \text{poly}(1/\varepsilon)$
2. $G[P(v)]$ connected
3. $\Pr(\#\text{Edges Cut} < \varepsilon|E|) \ge 2/3$

Why care about this primitive/goal?

- Gives property testers for planarity [HKNO 10, LR 15].
- Can estimate various graph parameters eg, max independent set, min vertex cover etc.
- Yields $\exp(1/\varepsilon)^2$ testers for all planar properties [NS 13, BKS 22].

Previous Implementations of Partition Oracles.

- [HKNO 10]: An $\exp(d/\varepsilon)$ time implementation.
- [LR 15]: $\exp(O(\log^2(d/\varepsilon))$ time implementation.

(Refines HKNO 10 analysis)

Key principle in [HKNO 10, LR 15]

- Describe a *global* poly(n)-time algo first to find a hyperfinite decomposition.
- Simulate Locally.

Previous Implementations of Partition Oracles.

Global (ε,k) Partition Oracle of [LR 15]. (A rough sketch)

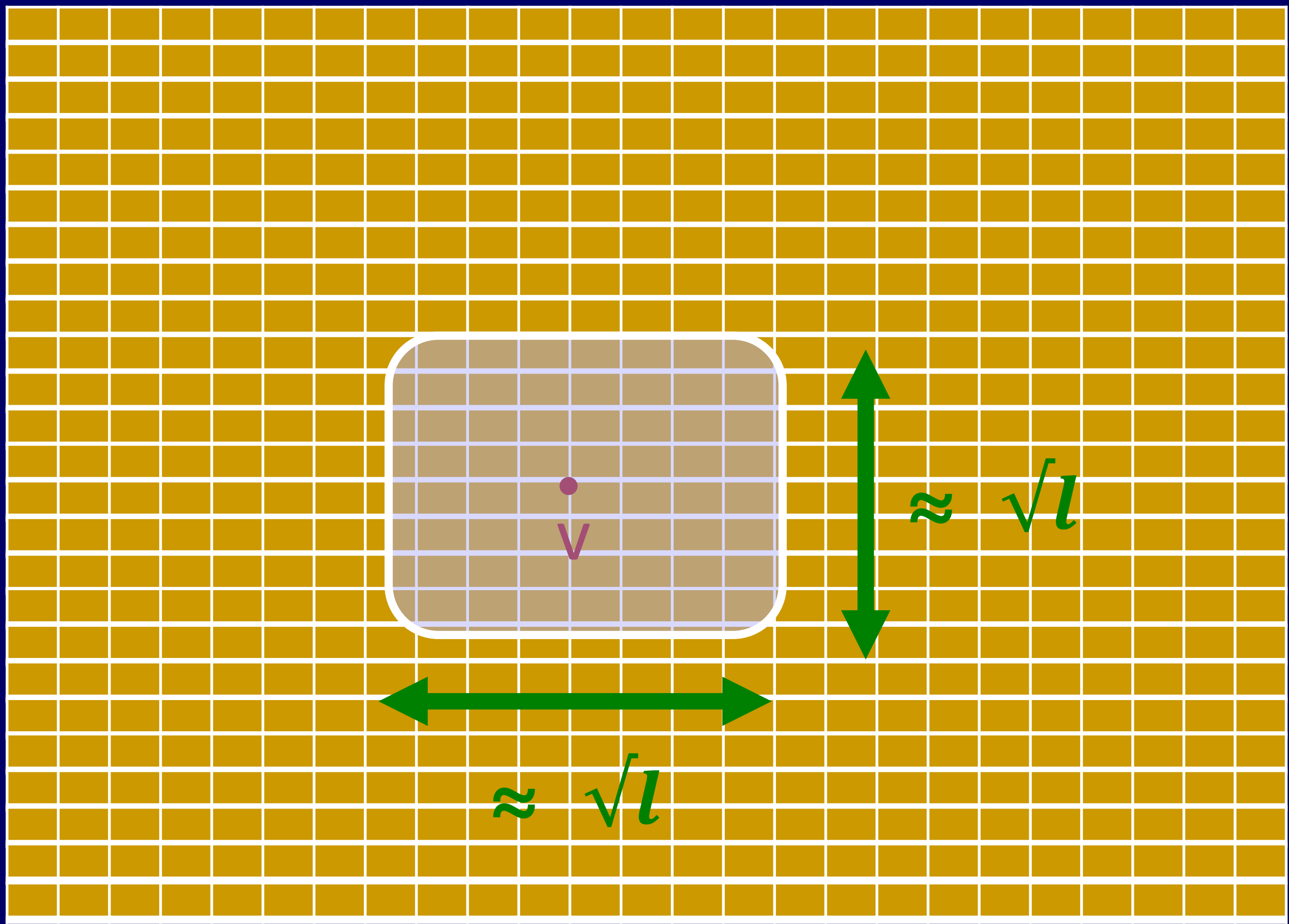
1. Set $l = O(\log(d/\varepsilon))$.
2. Set $G^0 = G$.
3. Perform l iterations. In i^{th} iteration

- Pick some random edges and contract. Merge end points.
- If any piece violates the size bound, unmerge.

Sketch of [LR 15] analysis.

1. #Edges Cut in $G^i \leq 0.99 \times$ #Edges Cut in G^{i-1}
2. After l iterations, #Edges Cut in $G^l \leq O(\varepsilon dn)$
3. Time taken to find piece of v in G^i: $O(k^{2i})$

Random Walks on planar graphs: A different starting point



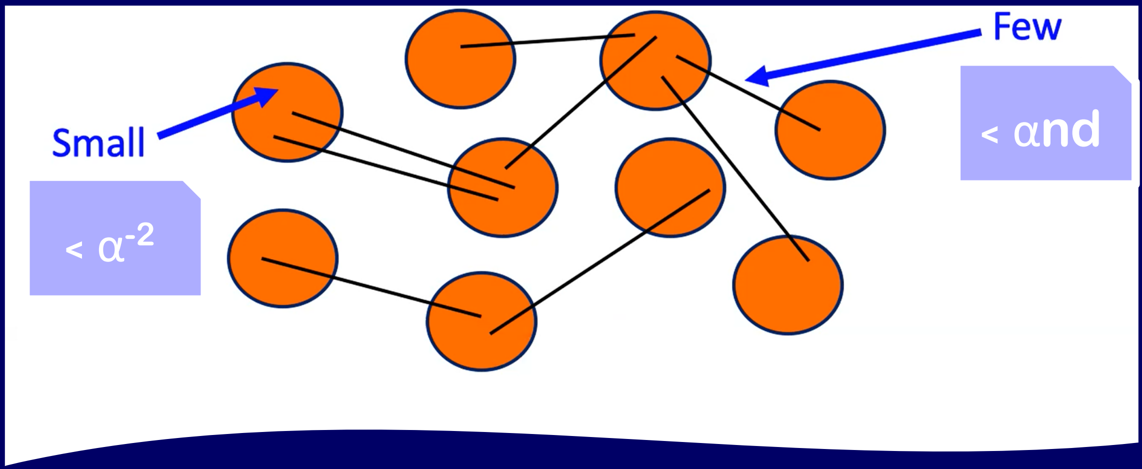
Let $p_{v,\,l} = M^{l}.\mathbb{1}_v$

What is $||p_{v,\,l}||_2^2$?

One notes that $||p_{v,\,l}||_2^2 \approx 1/l$.

Random Walks on planar graphs: A different starting point

Lemma: Let G be a bdd degree planar graph. Then for at least $(1 - 1/l)$ fraction of vertices $v \in V$, it holds that $||p_{v,l}||_2^2 \geq l^{-10}$

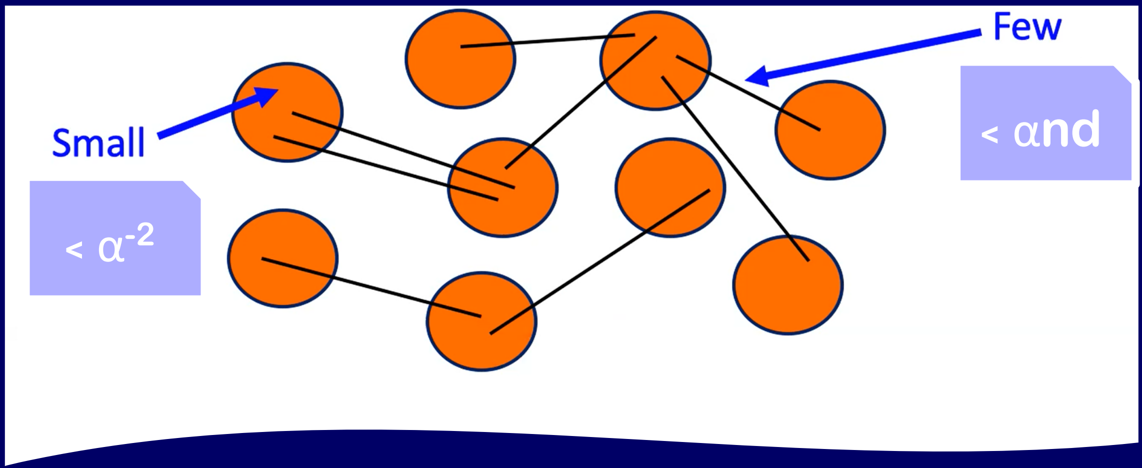


Think $\alpha = \varepsilon^2/100\, l$

$$\mathop{\mathbb{E}}_{v \sim \pi} \; \mathop{\mathbb{E}}_{W \sim W_{v,l}} [\text{Number of Cut Edges on } W] \leq \alpha\, l = \varepsilon^2/100$$

Random Walks on planar graphs: A different starting point

Lemma: Let G be a bdd degree planar graph. Then for at least $(1 - 1/l)$ fraction of vertices $v \in V$, it holds that $||p_{v,l}||_2^2 \geq l^{-10}$



Think $\alpha = \varepsilon^2/100\, l$

$\Rightarrow ||p_{v,l}||_2^2 \geq \text{poly}(\varepsilon)$

$$\mathop{\mathbb{E}}_{v \sim \pi} \; \mathop{\mathbb{E}}_{W \sim W_{v,l}} [\text{Number of Cut Edges on } W] \leq \alpha\, l = \varepsilon^2/100$$

By a Markov Bound, for at least $(1 - \varepsilon)$ fraction of vertices, the probability W never leaves is at least $1\text{-}\varepsilon$.

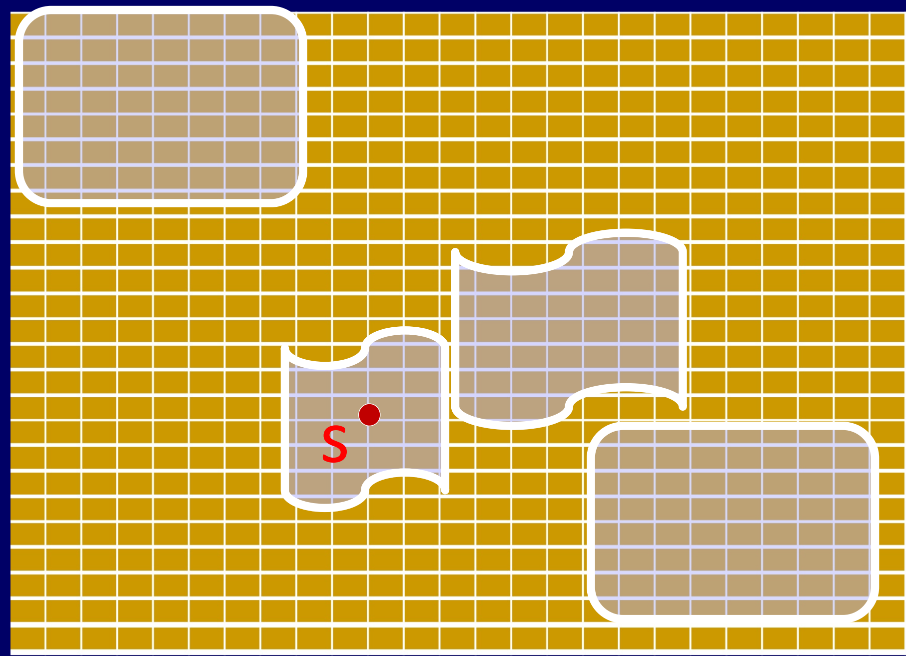
Random Walks on planar graphs: A different starting point

Call such vertices non-leaky ($||p_{v,l}||_2^2 \geq poly(\varepsilon)$).

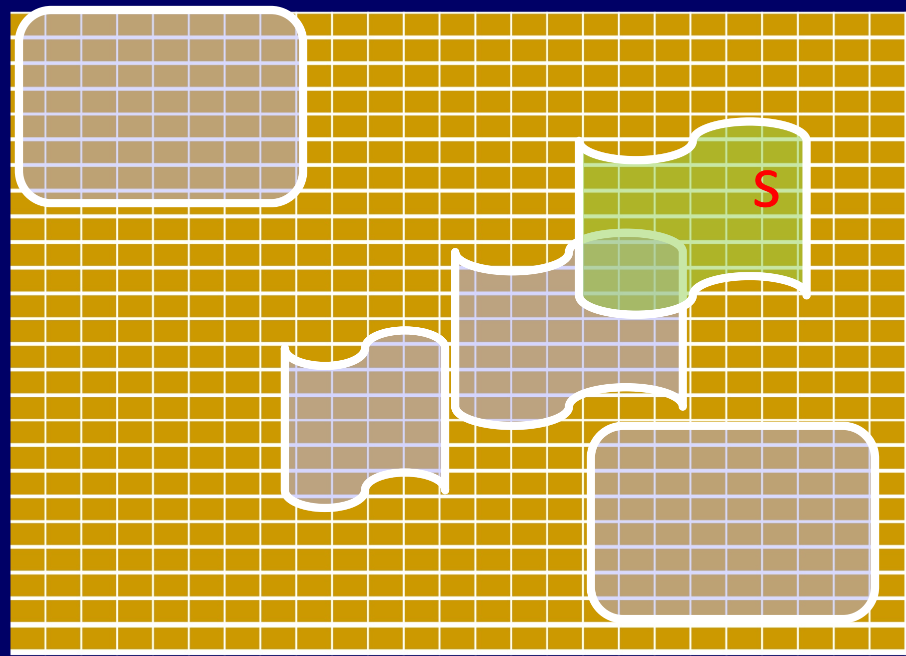
Define procedure C(s) = $\begin{cases} s, \ if\ s\ is\ leaky \\ l_1\ heavy\ hitters\ of\ pv_{,l}\ o/w \end{cases}$

We are ready for our global algorithm.

Few Leaky Vertices Lemma → A global algorithm



Few Leaky Vertices Lemma → A global algorithm



Few Leaky Vertices Lemma → A global algorithm

Our Algorithm (crudely)

/* Random Tape holds a random permutation of V */

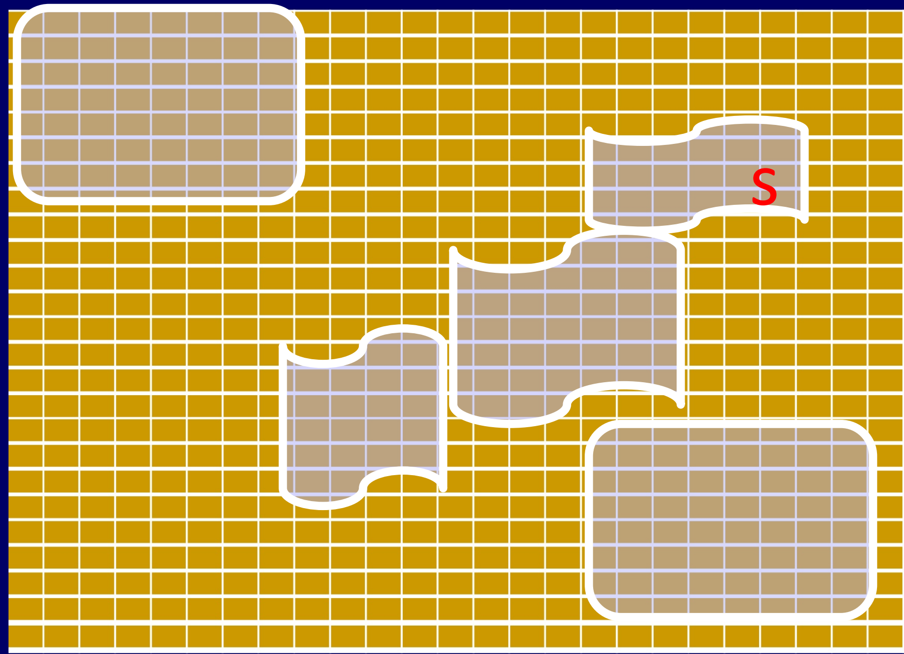
Let F = free vertices in i-th iteration.

Pick smallest s ∈ F

Find C(s)

Assign P(s) = C(s) ∩ F

Update F = F \ P(s)



Few Leaky Vertices Lemma → A global algorithm

Our Algorithm (crudely)

/* Random Tape holds a random permutation of V */

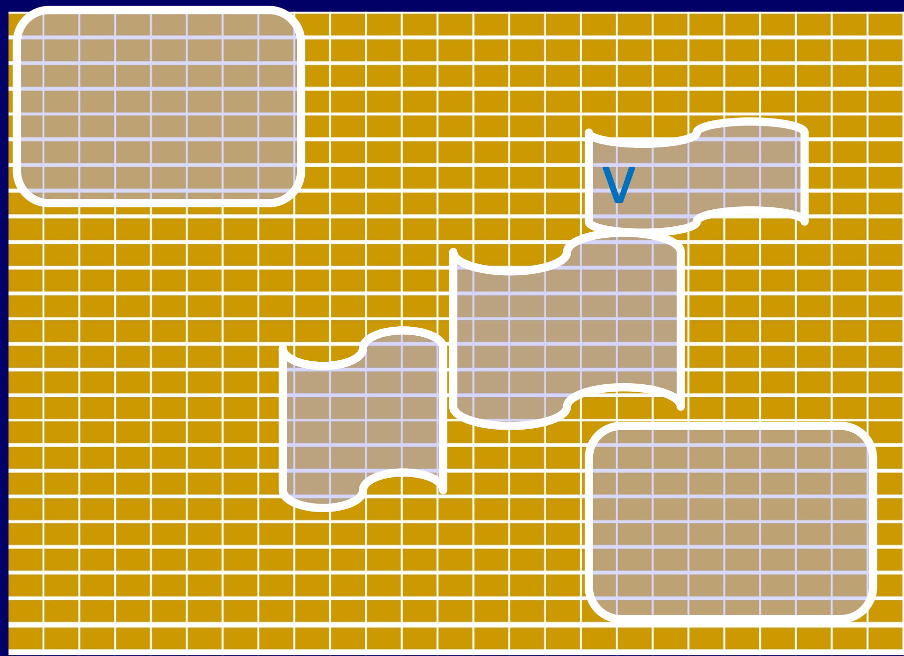
Let F = free vertices in i-th iteration.

Pick smallest s ∈ F

Find C(s)

Assign P(s) = C(s) ∩ F

Update F = F \ P(s)

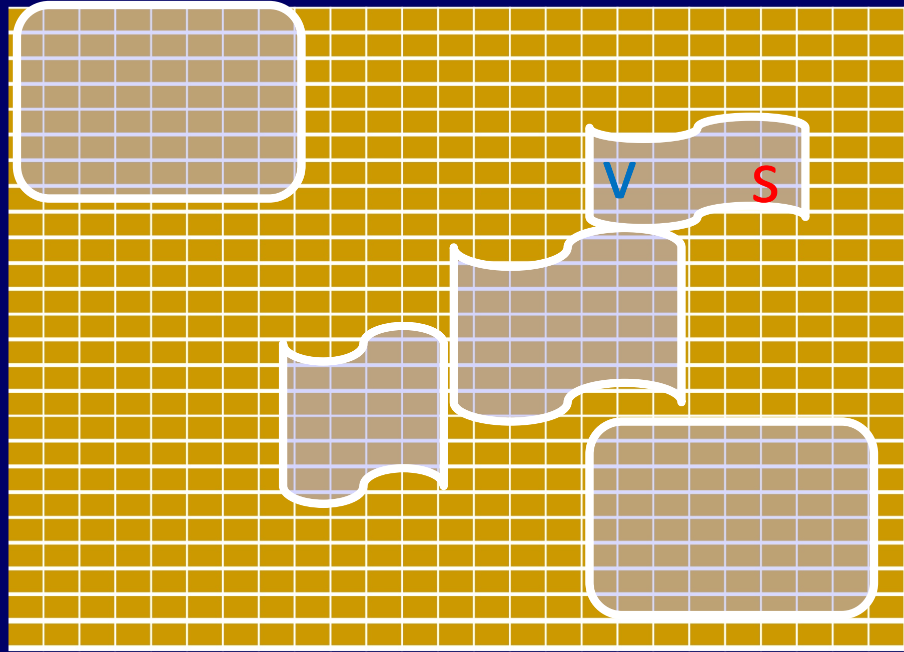


Few Leaky Vertices Lemma → A global algorithm

Our Algorithm (crudely)

/* Random Tape holds a random permutation of V */

Let F = free vertices in i-th iteration.
Pick smallest s ∈ F
Find C(s)
Assign P(s) = C(s) ∩ F
Update F = F \ P(s)



Challenges in local impl.

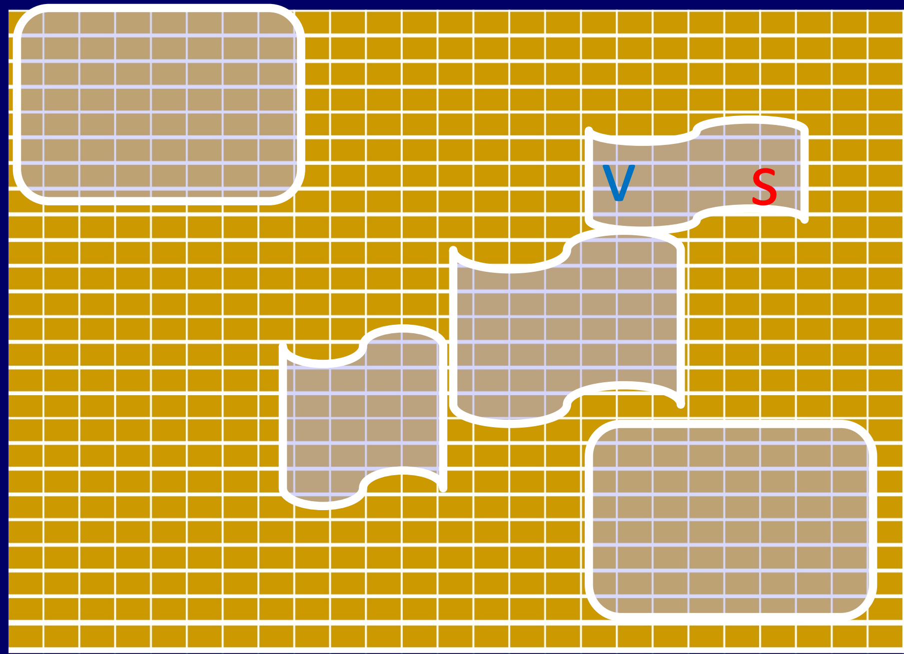
Def: We say anchor(v) = s where v ∈ P(s)

Need to find the “free vertex” s which captures v.
Need to find C(s) ∩ F

Few Leaky Vertices Lemma → A global algorithm

Suppose you can find anchor(v) for any $v \in V$.

Can handle both challenges now.



Challenges in local impl.

Def: We say anchor(v) = s where $v \in P(s)$

Need to find the "free vertex" s which captures v.

Need to find $C(s) \cap F$

Few Leaky Vertices Lemma → A global algorithm

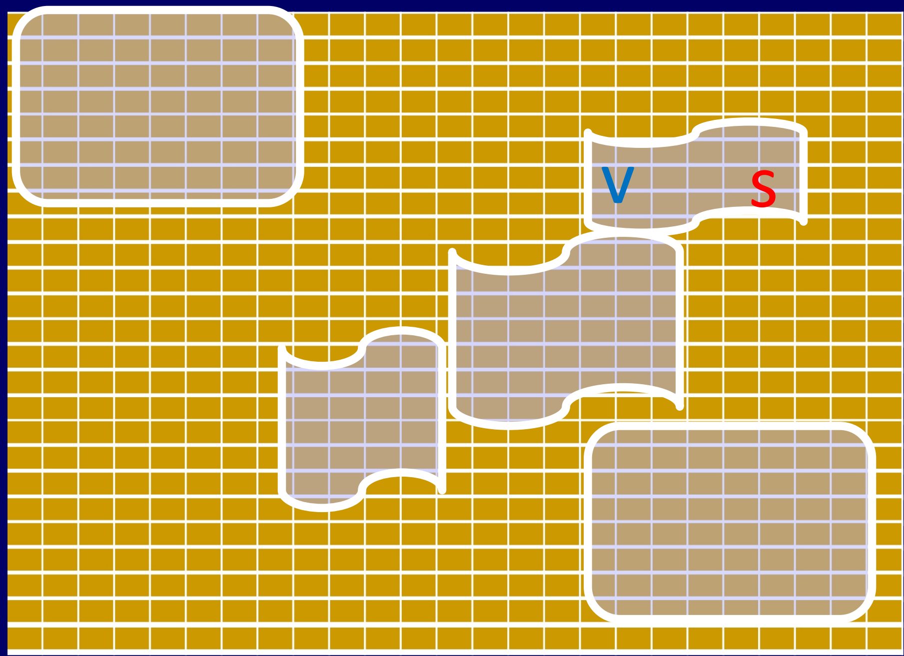
Annoyance:

How do I find anchor(v)?

v has a list of potential anchors. Need to know if any of those were already captured.

Need to determine anchors recursively.

We could not analyze this process.



Challenges in local impl.

Def: We say anchor(v) = s where $v \in P(s)$

Need to find the "free vertex" s which captures v.

Need to find $C(s) \cap F$

A New global algorithm

Our Algorithm (still crudely)

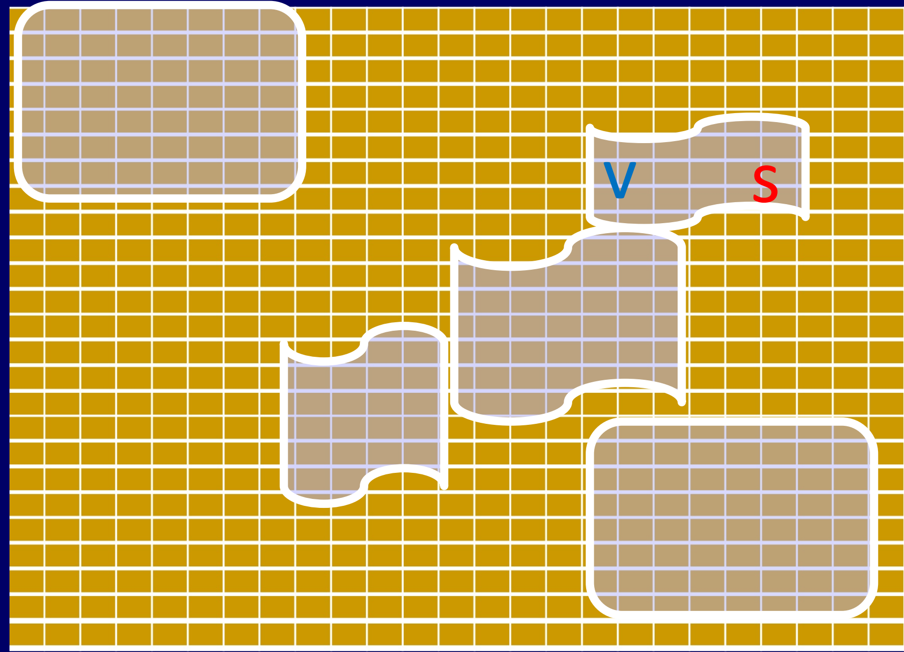
/* Random Tape holds a random permutation of V */

Let F = V

For s = 1 to n

- Find C(s)
- Assign P(s) = C(s) ∩ F
- Update F = F \ P(s)

Easy to implement locally.



A danger

While C(s) has a small edge boundary.

P(s) need not!

Way out: Amortize.

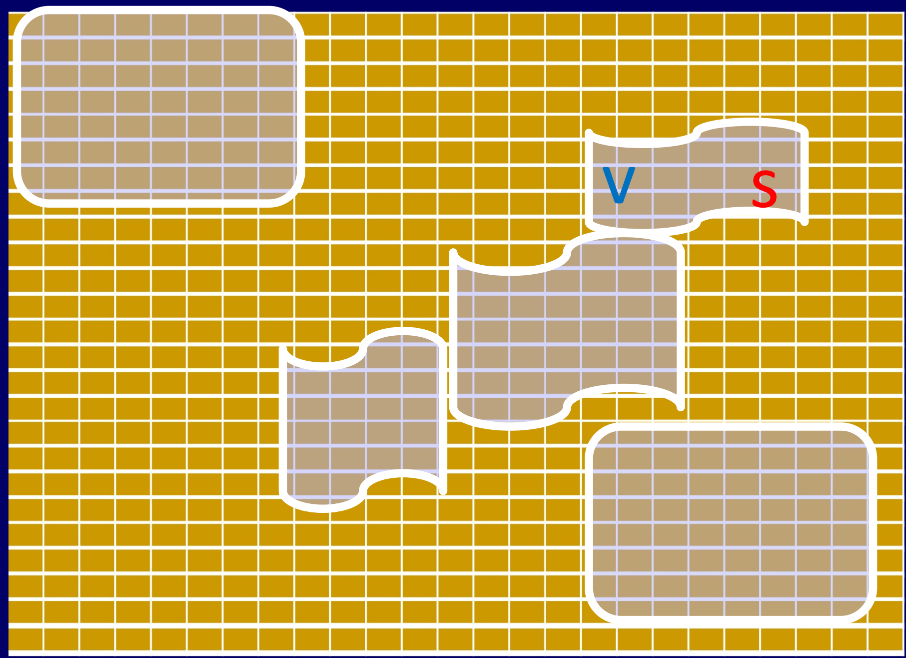
A New global algorithm

Handling Danger via Isoperimetry

Theorem: Suppose $|F| \geq \varepsilon n$. Then for at least $\varepsilon^2 n$ vertices $s \in F$, it holds that

$$|C(s) \cap F| \geq \varepsilon^3 |C(s)|$$

Using this theorem, you can show that for every clustered vertex, on an average you cut εd edges.



A danger

While C(s) has a small edge boundary.
P(s) need not!
Way out: Amortize.

More applications?

[Levi-Shoshan 21] Gave a tester for Hamiltonicity on Planar Graphs using Partition Oracles.

[Basu-K-Seshadhri 22] Gave an $\exp(1/\varepsilon^2)$ query tester for all planar properties.

Open Problems

- Extending to unbounded degree planar graphs.
- Distributed applications?
- LCA for Hyperfinite Decompositions?

Thanks for your time