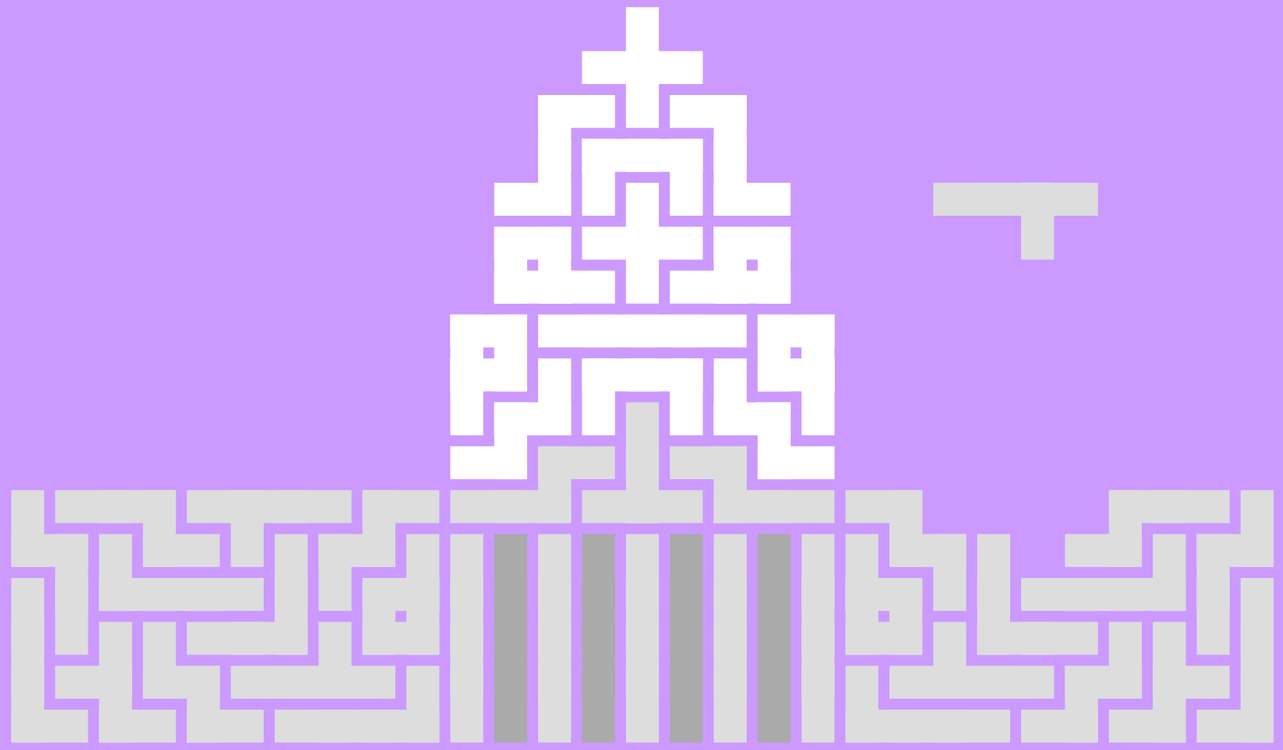


Algorithms For Democratic Decision-Making

Jamie Tucker-Foltz • Yale University • Spring 2026

Lecture 22: **Detecting Gerrymandering**

Announcements

Office hours tomorrow:
13th floor of Kline tower,
Room 1305

Announcements

Final presentation schedule has just been posted on Canvas!

Announcements

Final presentation schedule has just been posted on Canvas!
Format: 20 minutes + 5 minutes for questions (the group of 2 can take a bit longer)

Announcements

Final presentation schedule has just been posted on Canvas!
Format: 20 minutes + 5 minutes for questions (the group of 2 can take a bit longer)

Final project reports are due **May 6th at 11:59pm EDT**. This is the deadline set by the school, and is the latest that assignments can be due.

Announcements

Final presentation schedule has just been posted on Canvas!
Format: 20 minutes + 5 minutes for questions (the group of 2 can take a bit longer)

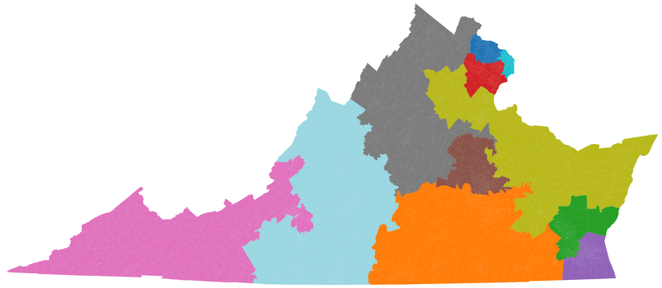
Final project reports are due **May 6th at 11:59pm EDT**. This is the deadline set by the school, and is the latest that assignments can be due.

Please also refresh yourself on the course AI Policy:

1. Take ownership of anything you write in your reports, slides, or say in your presentation, even if AI wrote the first version of it. **You** must fully understand what you are saying, and know that it is correct.
2. For the final project, you must keep track of **everything** you used AI for, in detail, and declare it when you submit your final project. Failing to disclose the use of AI will be considered a violation of Yale's academic integrity policy.
3. Do not let AI replace you as a writer. For instance, if you write a paragraph in your introduction motivating why your problem is interesting, it should be written in your own words using your own ideas. I do not want to read about what AI thinks about a problem. I want to hear what **you** think.

How do you tell if a map is intentionally gerrymandered?

?

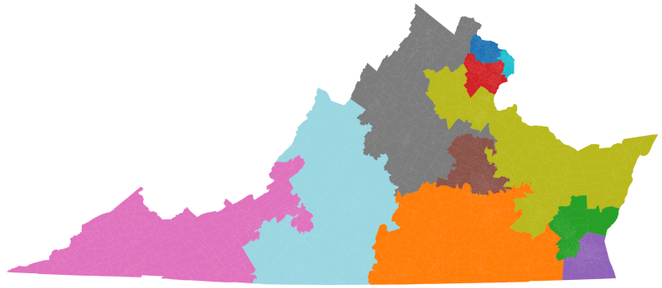


?

How do you tell if a map is intentionally gerrymandered?

Voter Data

+

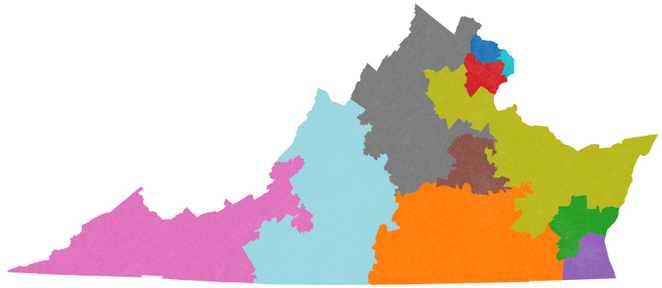


=

How do you tell if a map is intentionally gerrymandered?

Voter Data

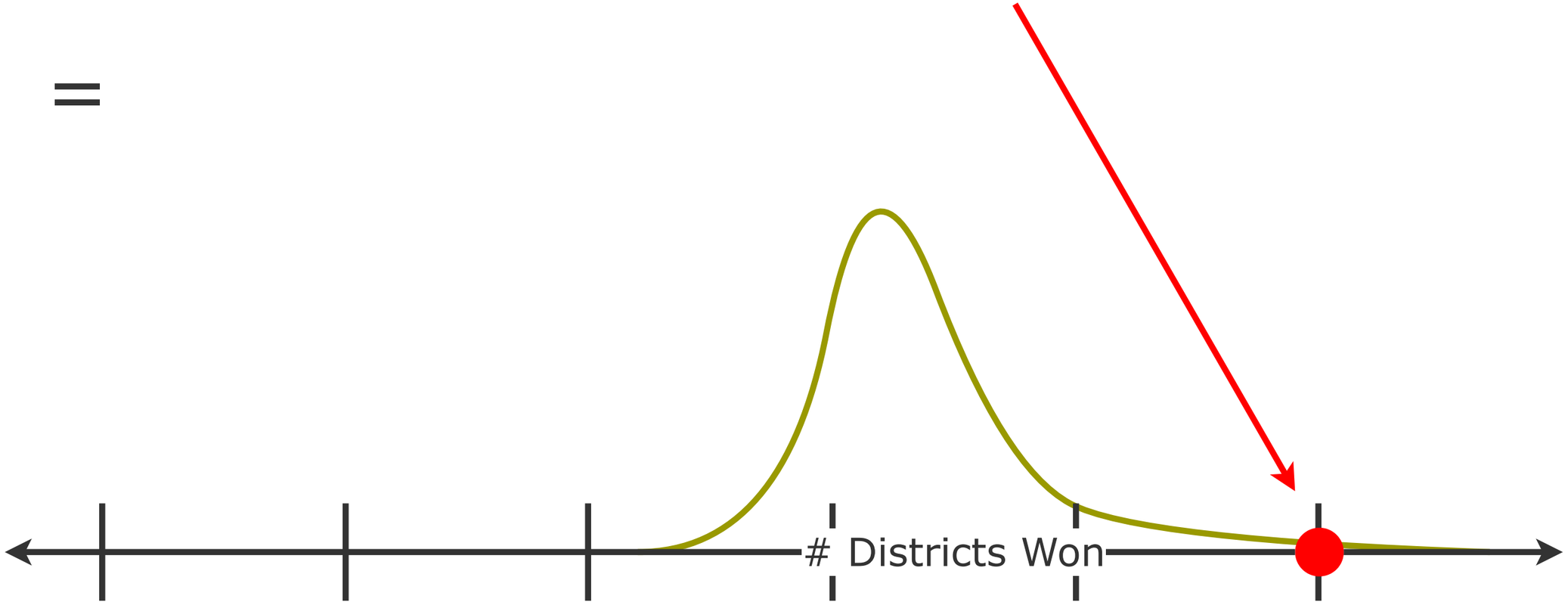
+



+

Ensemble of computed alternatives

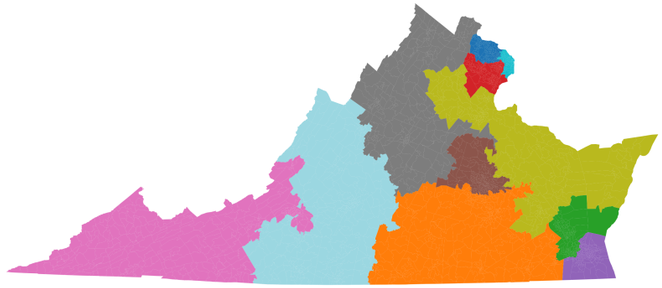
=



How do you tell if a map is intentionally gerrymandered?

Voter Data

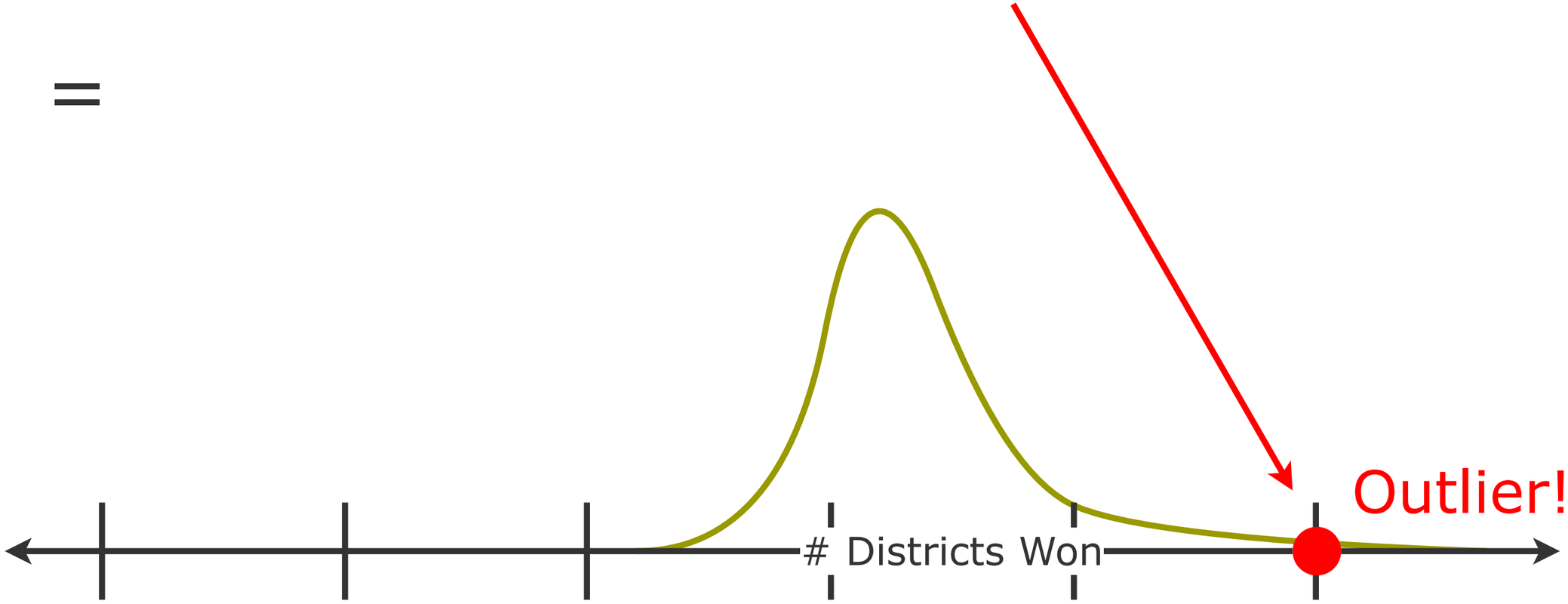
+



+

Ensemble of computed alternatives

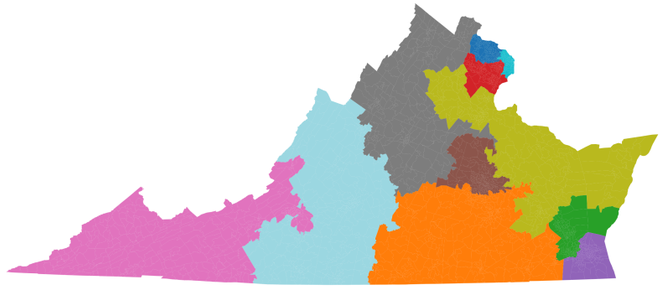
=



How do you tell if a map is intentionally gerrymandered?

Voter Data

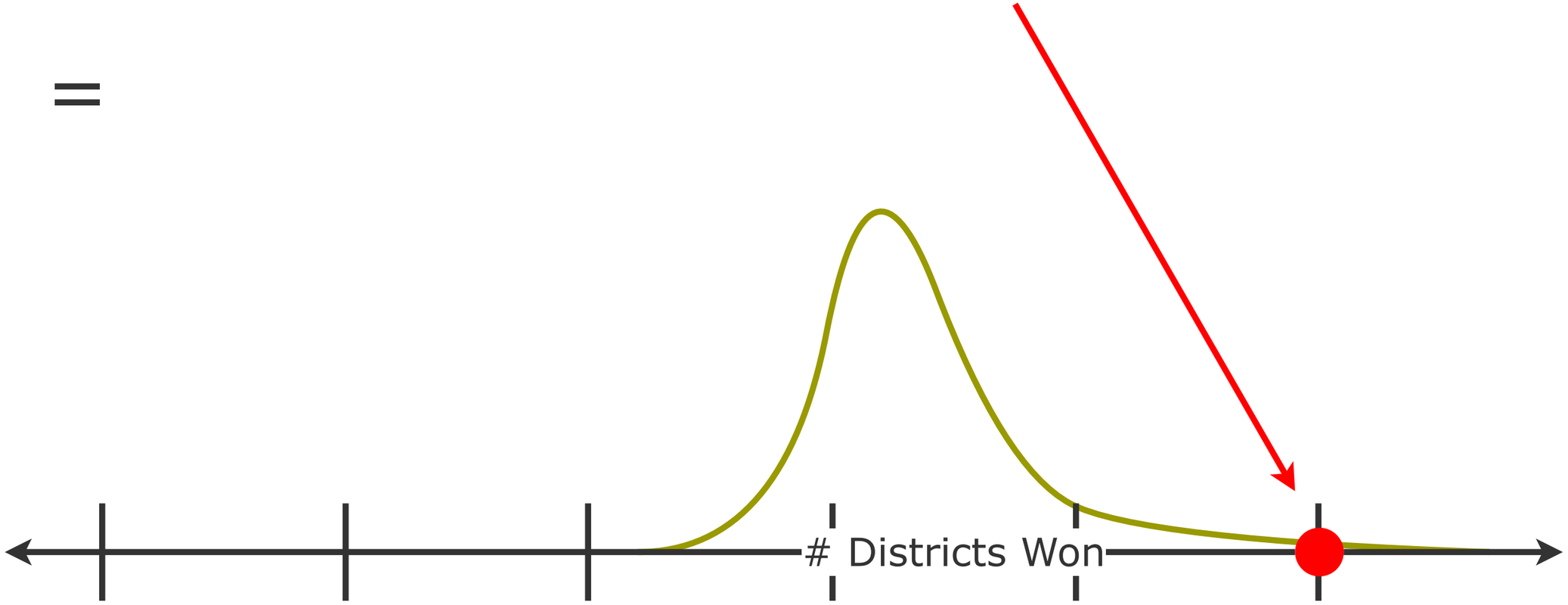
+



+

Ensemble of computed alternatives

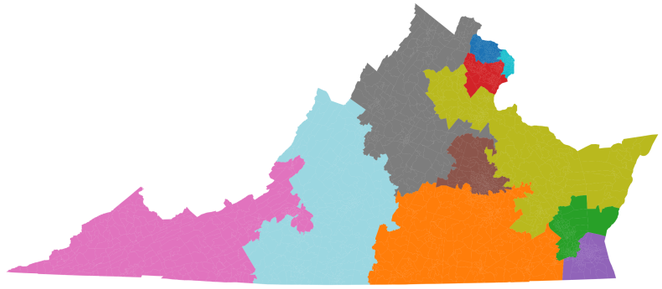
=



How do you tell if a map is intentionally gerrymandered?

Voter Data

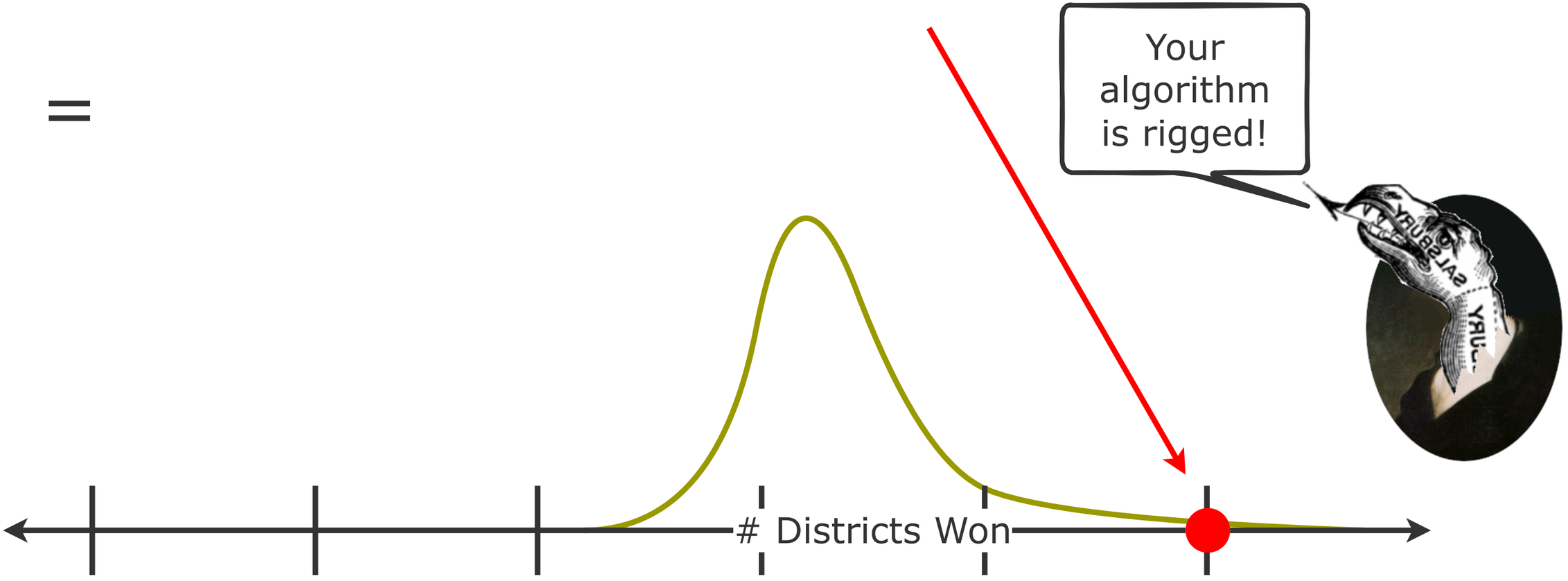
+



+

Ensemble of computed alternatives

=



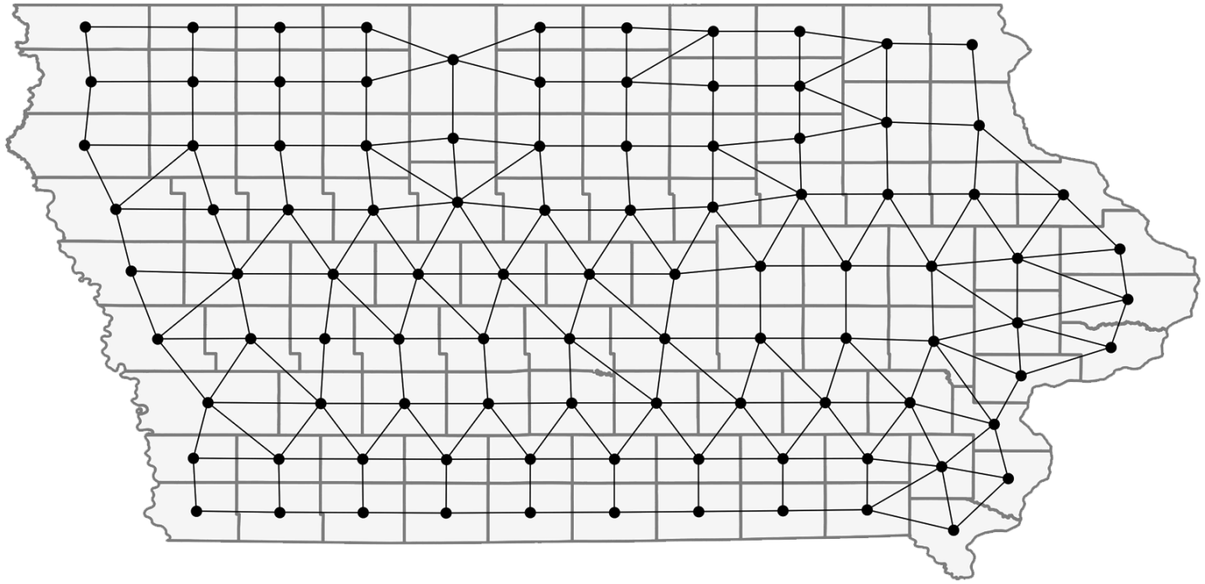
Knowing the distribution matters

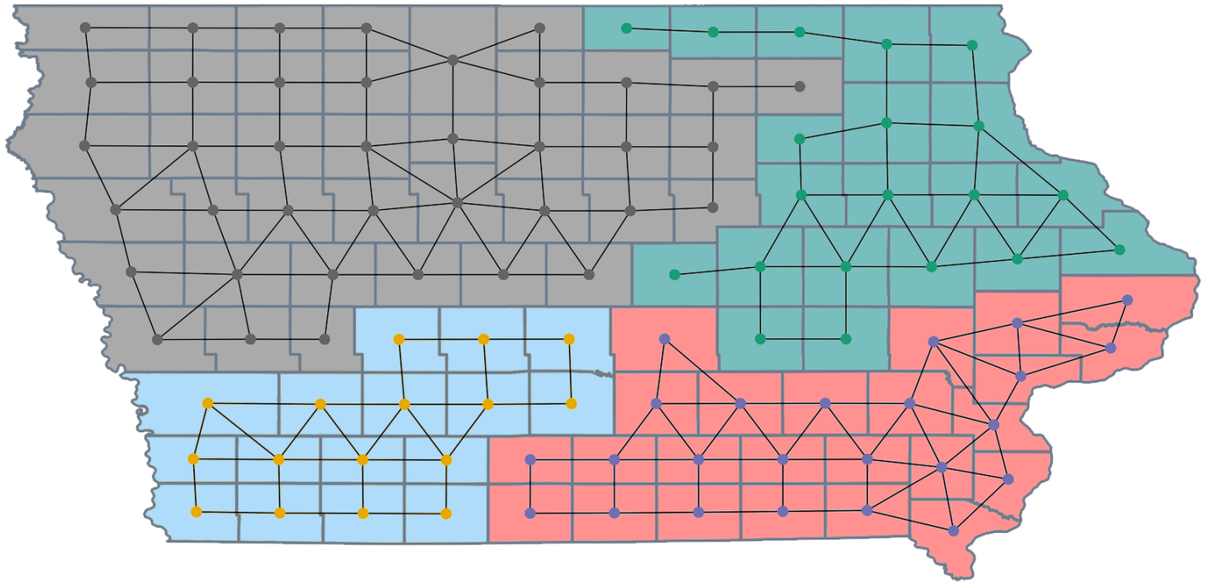
Knowing the distribution matters

"*[The plaintiff's expert witness] purports to have an algorithm that randomly generates maps. He has never evaluated this claim in any rigorous way. In my assessment of this 'random' framework algorithm on a very small toy redistricting data set, I found that the strategy generated a biased set of maps that oversamples some maps while undersampling other maps.*"

—Wendy K. Tam Cho, expert report
League of Women Voters of Pennsylvania
v. Commonwealth of Pennsylvania (2017)

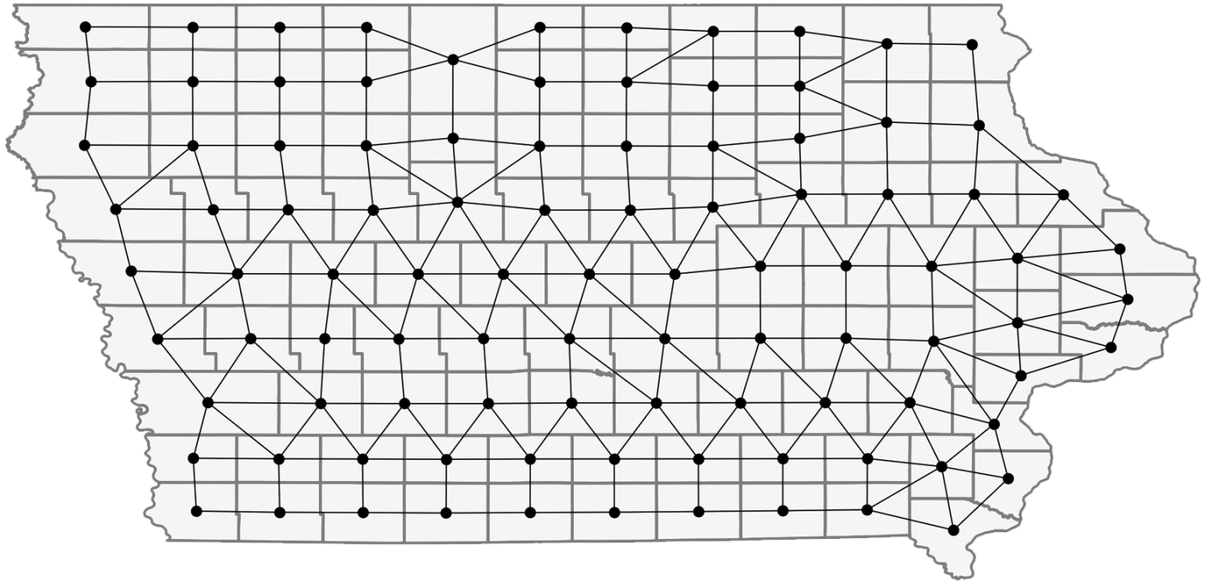
The computational problem

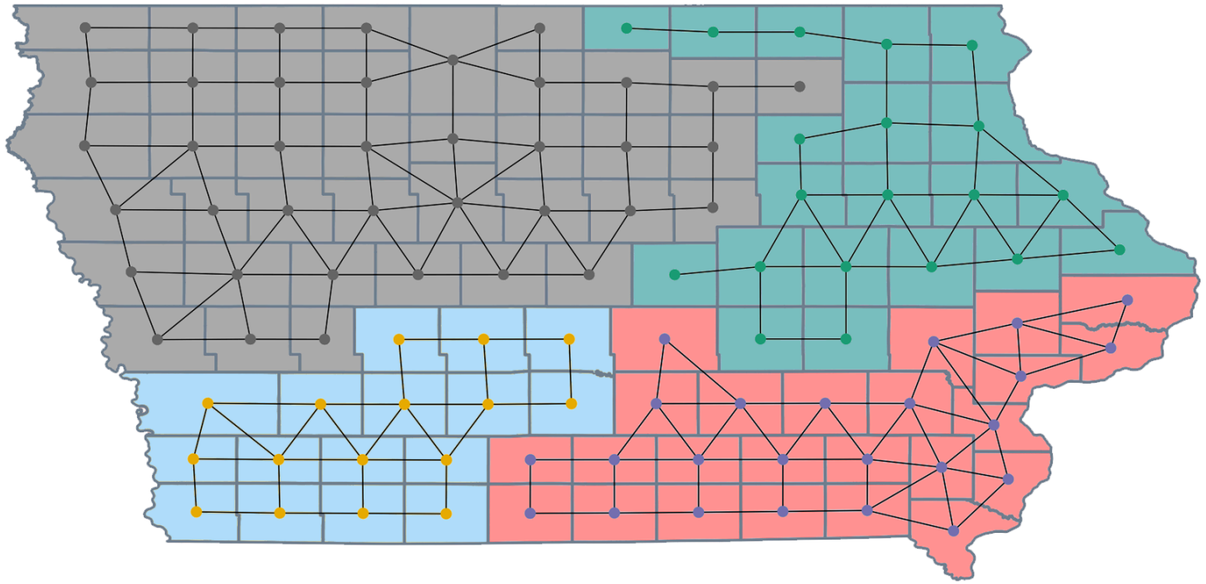




Input: A vertex-weighted undirected graph *G* and a positive integer *k*.
Redistricting map: Partition of *G* into equal-weight connected subgraphs.

The computational problem





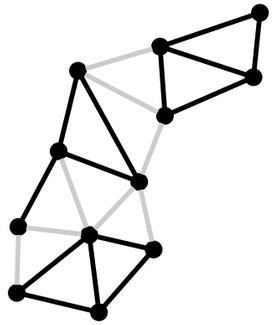
Input: A vertex-weighted undirected graph *G* and a positive integer *k*.
Redistricting map: Partition of *G* into equal-weight connected subgraphs.

The technical question: **How can we efficiently perform statistical analysis on the space of all balanced, connected graph partitions of *G*?**

The state of the art: MCMC

Algorithm (ReCom)

Starting from an arbitrary balanced k-partition, repeatedly:

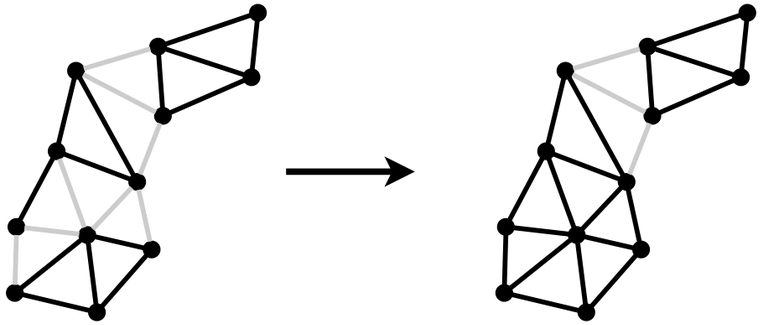


The state of the art: MCMC

Algorithm (ReCom)

Starting from an arbitrary balanced k-partition, repeatedly:

1. Merge two random districts.

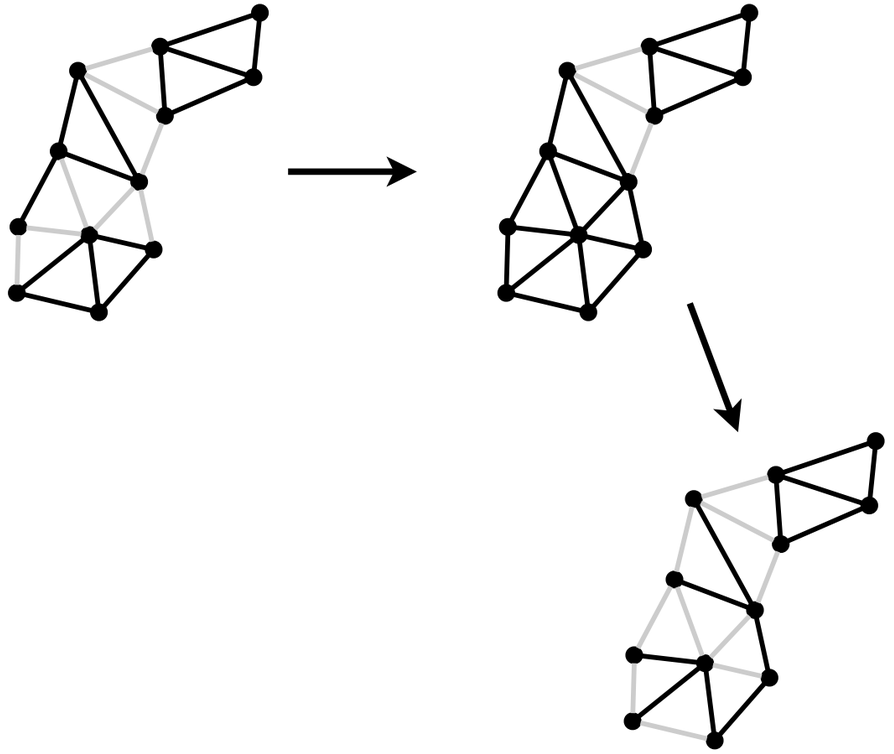


The state of the art: MCMC

Algorithm (ReCom)

Starting from an arbitrary balanced k-partition, repeatedly:

1. Merge two random districts.
2. Sample a random spanning tree T on the merged district.

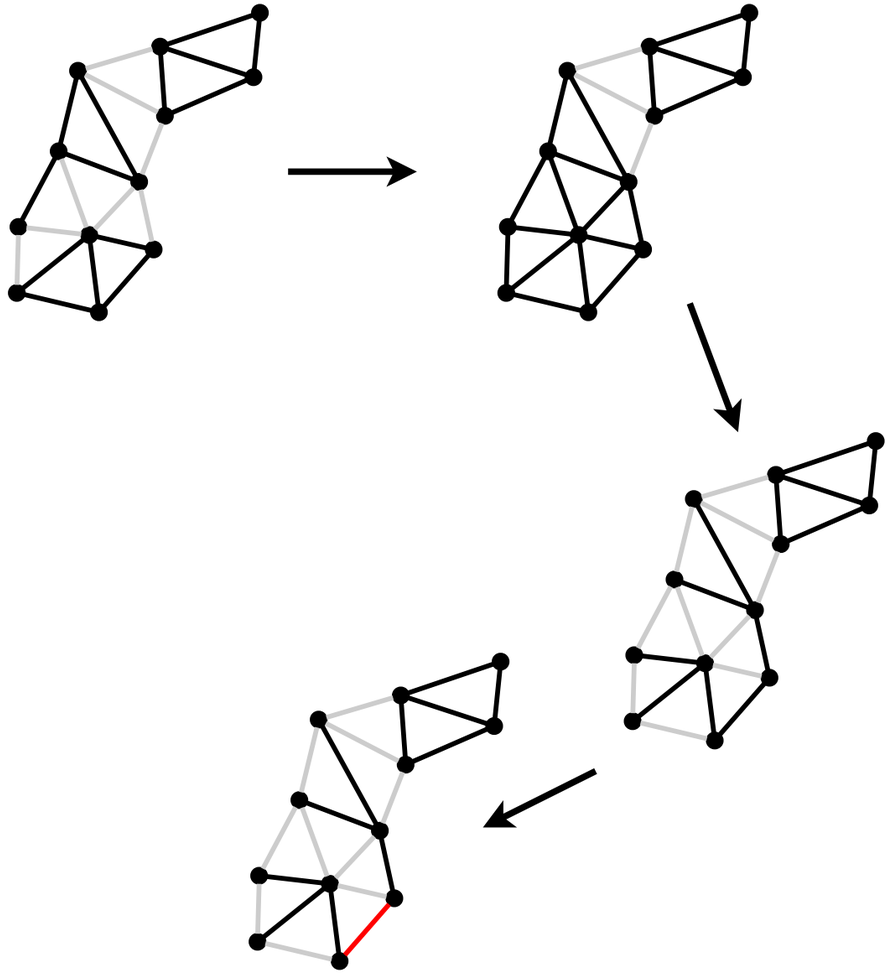


The state of the art: MCMC

Algorithm (ReCom)

Starting from an arbitrary balanced k-partition, repeatedly:

1. Merge two random districts.
2. Sample a random spanning tree T on the merged district.
3. Try to split T on a single edge into balanced subtrees. If not possible, start over.

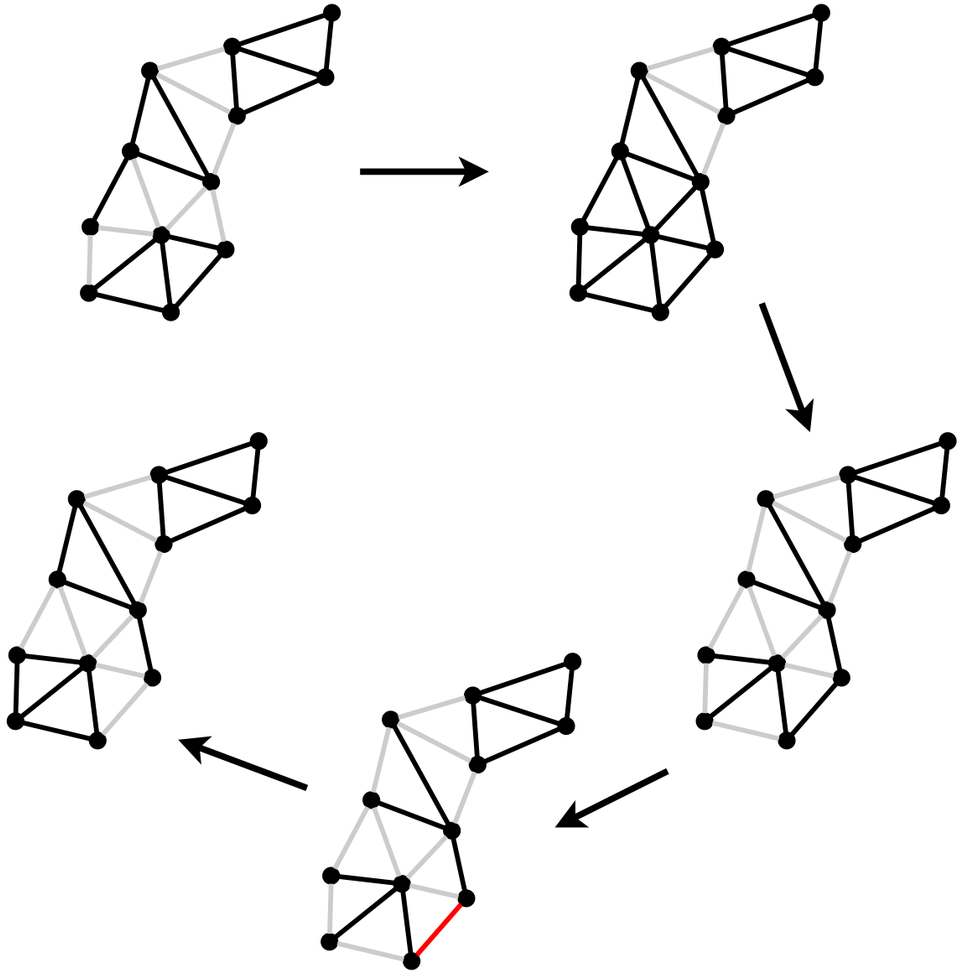


The state of the art: MCMC

Algorithm (ReCom)

Starting from an arbitrary balanced k-partition, repeatedly:

1. Merge two random districts.
2. Sample a random spanning tree T on the merged district.
3. Try to split T on a single edge into balanced subtrees. If not possible, start over.
4. Update the k-partition, new parts induced by subtrees.



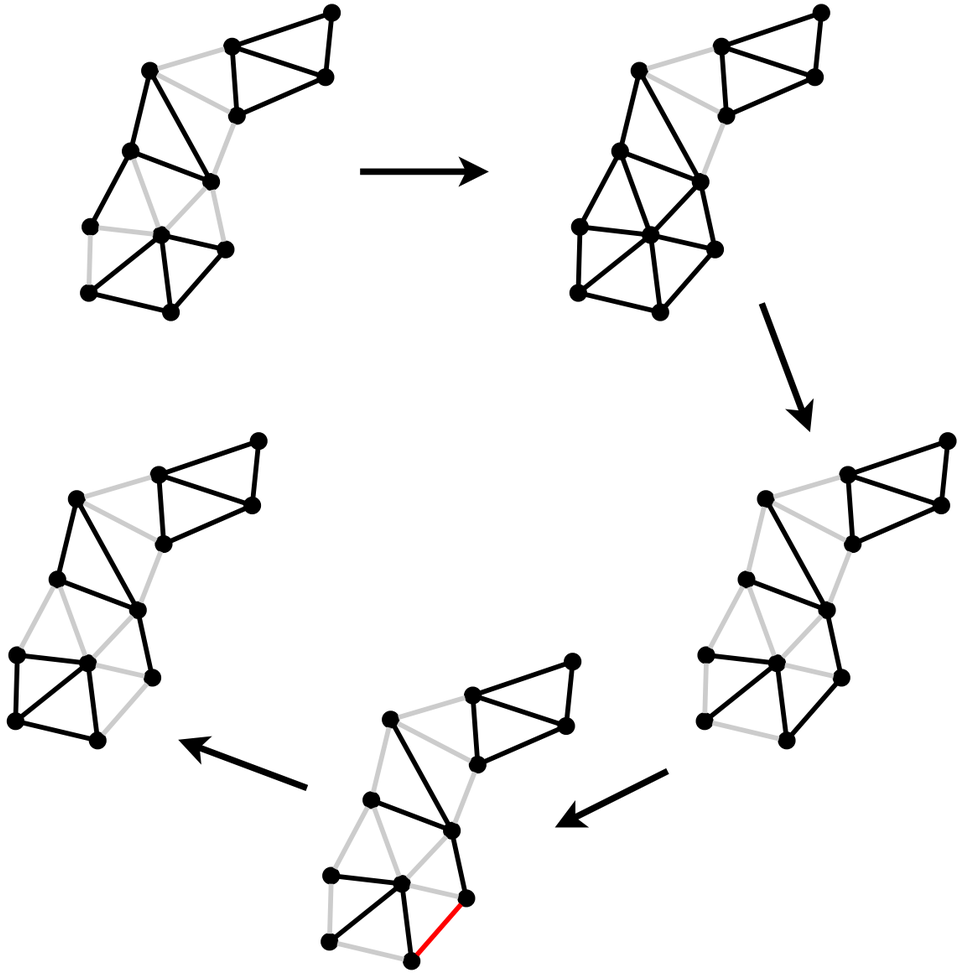
The state of the art: MCMC

Algorithm (ReCom)

Starting from an arbitrary balanced k-partition, repeatedly:

1. Merge two random districts.
2. Sample a random spanning tree T on the merged district.
3. Try to split T on a single edge into balanced subtrees. If not possible, start over.
4. Update the k-partition, new parts induced by subtrees.

Run for many steps, then output the final partition.



ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

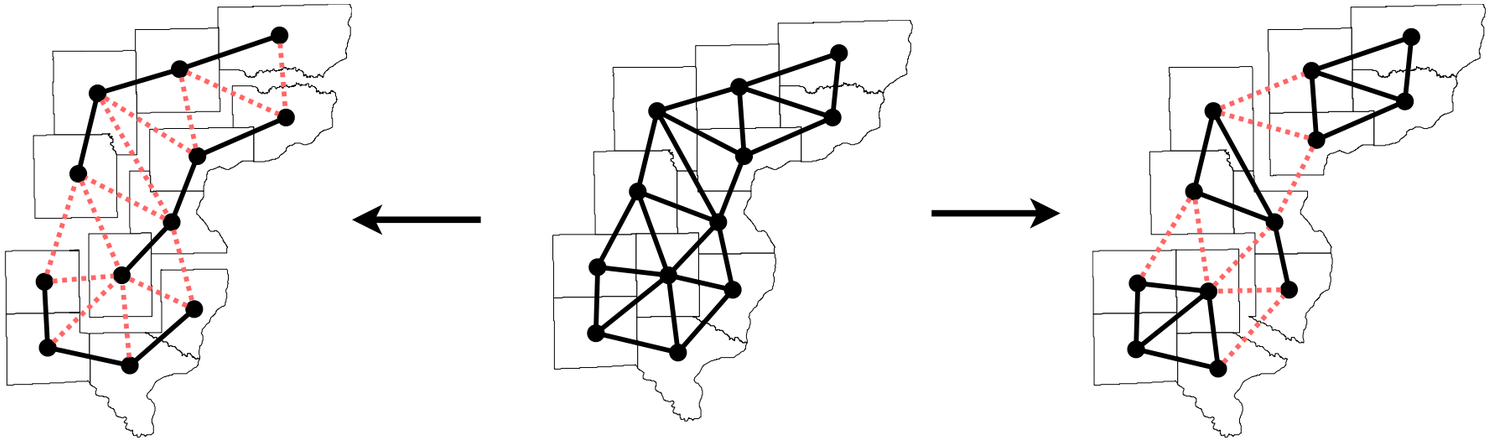
$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$



ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

- Empirical and theoretical work has argued it's a good distribution for redistricting
  - e.g., [Clelland, Bossenbroek, Heckmaster Nelson, Rock, VanAusdall, 2021], [Procaccia and T-F, 2022], [Tapp, 2021]

ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

- Empirical and theoretical work has argued it's a good distribution for redistricting
  - e.g., [Clelland, Bossenbroek, Heckmaster Nelson, Rock, VanAusdall, 2021], [Procaccia and T-F, 2022], [Tapp, 2021]
- Multiple algorithms have been developed to (exactly/approximately) sample from it
  - e.g., [DeFord, Duchin, Solomon, 2021], [McCartan, Imai, 2020], [Akitaya, Cannon, Herschlag, Schoenbach, Tapp, T-F]

ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

- Empirical and theoretical work has argued it's a good distribution for redistricting
  - e.g., [Clelland, Bossenbroek, Heckmaster Nelson, Rock, VanAusdall, 2021], [Procaccia and T-F, 2022], [Tapp, 2021]
- Multiple algorithms have been developed to (exactly/approximately) sample from it
  - e.g., [DeFord, Duchin, Solomon, 2021], [McCartan, Imai, 2020], [Akitaya, Cannon, Herschlag, Schoenbach, Tapp, T-F]

Complexity questions:

1. How fast is the ReCom **transition function**?

2. How fast does the Markov chain **mix**?

ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

- Empirical and theoretical work has argued it's a good distribution for redistricting
  - e.g., [Clelland, Bossenbroek, Heckmaster Nelson, Rock, VanAusdall, 2021], [Procaccia and T-F, 2022], [Tapp, 2021]
- Multiple algorithms have been developed to (exactly/approximately) sample from it
  - e.g., [DeFord, Duchin, Solomon, 2021], [McCartan, Imai, 2020], [Akitaya, Cannon, Herschlag, Schoenbach, Tapp, T-F]

Complexity questions:

1. How fast is the ReCom **transition function**?
2. How fast does the Markov chain **mix**?
3. Is there some **other way** to efficiently sample from the Spanning Tree Distribution?

ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

- Empirical and theoretical work has argued it's a good distribution for redistricting
  - e.g., [Clelland, Bossenbroek, Heckmaster Nelson, Rock, VanAusdall, 2021], [Procaccia and T-F, 2022], [Tapp, 2021]
- Multiple algorithms have been developed to (exactly/approximately) sample from it
  - e.g., [DeFord, Duchin, Solomon, 2021], [McCartan, Imai, 2020], [Akitaya, Cannon, Herschlag, Schoenbach, Tapp, T-F]

Complexity questions:

1. How fast is the ReCom **transition function**?
2. How fast does the Markov chain **mix**?
3. Is there some **other way** to efficiently sample from the Spanning Tree Distribution?
   - Yes, for bounded k

ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

- Empirical and theoretical work has argued it's a good distribution for redistricting
  - e.g., [Clelland, Bossenbroek, Heckmaster Nelson, Rock, VanAusdall, 2021], [Procaccia and T-F, 2022], [Tapp, 2021]
- Multiple algorithms have been developed to (exactly/approximately) sample from it
  - e.g., [DeFord, Duchin, Solomon, 2021], [McCartan, Imai, 2020], [Akitaya, Cannon, Herschlag, Schoenbach, Tapp, T-F]

Complexity questions:

1. How fast is the ReCom **transition function**?
2. How fast does the Markov chain **mix**?
   - This is a huge open question.
3. Is there some **other way** to efficiently sample from the Spanning Tree Distribution?
   - Yes, for bounded k

ReCom and the Spanning Tree Distribution

Theorem (Cannon, Duchin, Randall, Rule, 2024)

With some tweaks, the stationary distribution of ReCom is the Spanning Tree Distribution:

$$\Pr[P = (D_1, D_2, \ldots, D_k)] \sim \prod_{i=1}^{k} (\#\text{ spanning trees of } D_i)$$

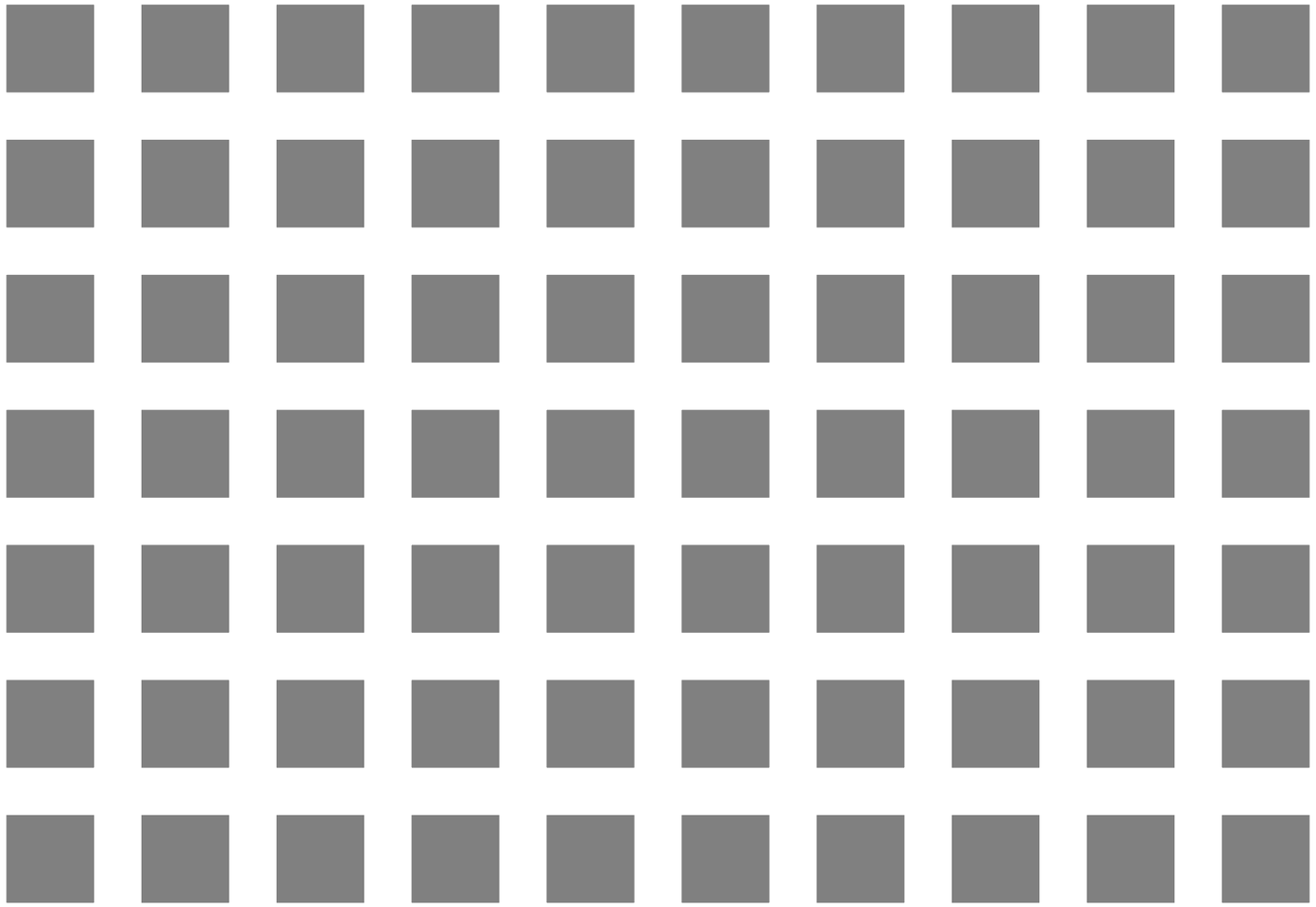
- Empirical and theoretical work has argued it's a good distribution for redistricting
  - e.g., [Clelland, Bossenbroek, Heckmaster Nelson, Rock, VanAusdall, 2021], [Procaccia and T-F, 2022], [Tapp, 2021]
- Multiple algorithms have been developed to (exactly/approximately) sample from it
  - e.g., [DeFord, Duchin, Solomon, 2021], [McCartan, Imai, 2020], [Akitaya, Cannon, Herschlag, Schoenbach, Tapp, T-F]

Complexity questions:

1. How fast is the ReCom **transition function**?
   - Micah has 2 weeks left to solve this otherwise he fails this course
2. How fast does the Markov chain **mix**?
   - This is a huge open question.
3. Is there some **other way** to efficiently sample from the Spanning Tree Distribution?
   - Yes, for bounded k

Can we sample from the Spanning Tree Distribution in polynomial time?

Can we sample from the Spanning Tree Distribution in polynomial time?



Until a few years ago, this was unknown even on grid graphs!

Can we sample from the Spanning Tree Distribution in polynomial time?

Conjecture (Charikar, Liu, Liu, Vuong, 2022)

The following algorithm runs in polynomial time on grid graphs, for constant k:

1. *Sample a uniformly random k-component forest*
2. *Reject and try again if it's not balanced*

Can we sample from the Spanning Tree Distribution in polynomial time?

Conjecture (Charikar, Liu, Liu, Vuong, 2022)

The following algorithm runs in polynomial time on grid graphs, for constant k:

1. *Sample a uniformly random k-component forest*
2. *Reject and try again if it's not balanced*

Theorem (Cannon, Pegden, T-F, 2024)

The following algorithm runs in polynomial time on grid graphs, for constant k:

1. *Sample a uniformly random **spanning tree***
2. *Try to find $k-1$ edges to remove to get a balanced forest, otherwise restart*

Can we sample from the Spanning Tree Distribution in polynomial time?

Conjecture (Charikar, Liu, Liu, Vuong, 2022)

The following algorithm runs in polynomial time on grid graphs, for constant k:

1. *Sample a uniformly random k-component forest*
2. *Reject and try again if it's not balanced*

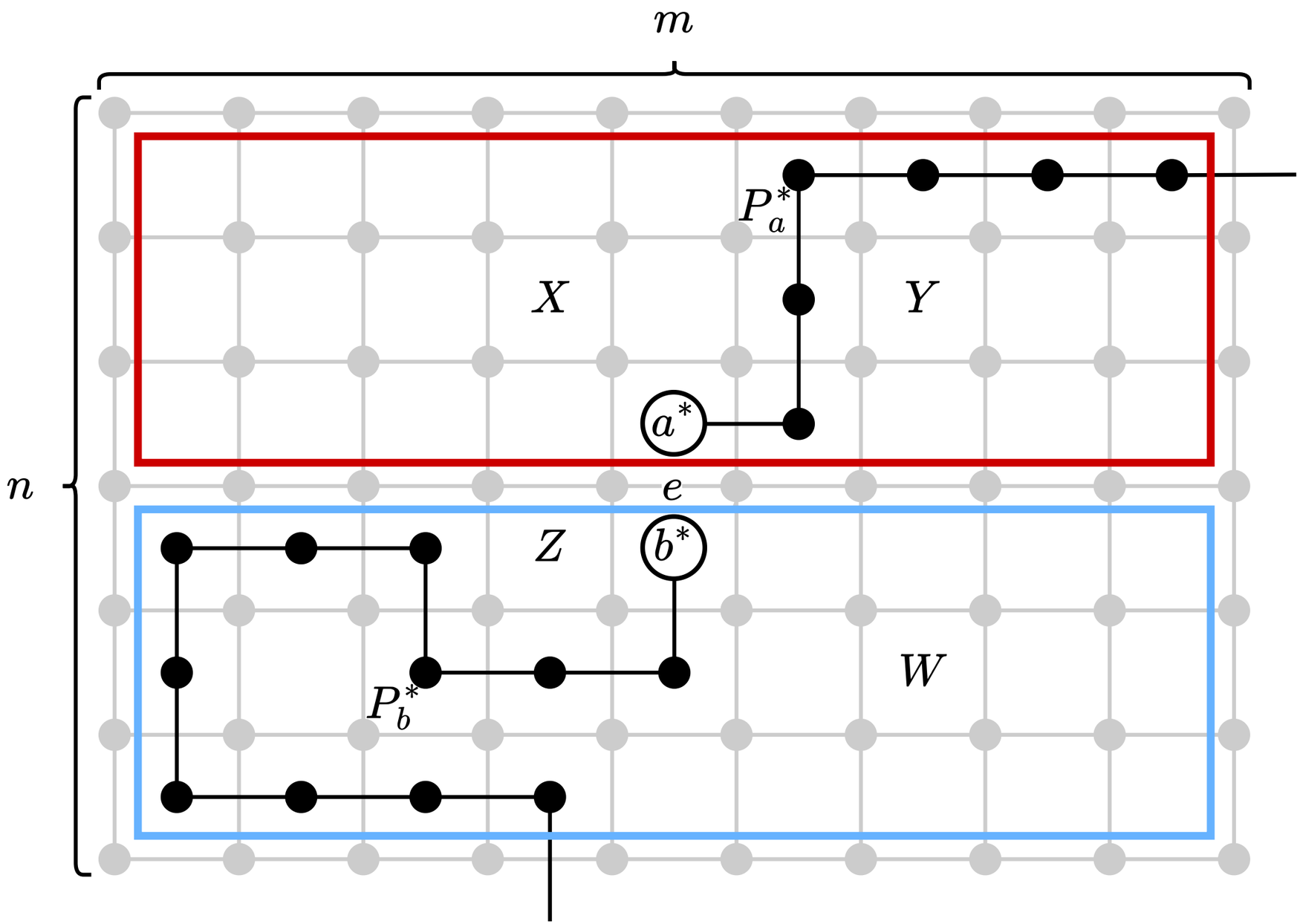
Theorem (Cannon, Pegden, T-F, 2024)

The following algorithm runs in polynomial time on grid graphs, for constant k:

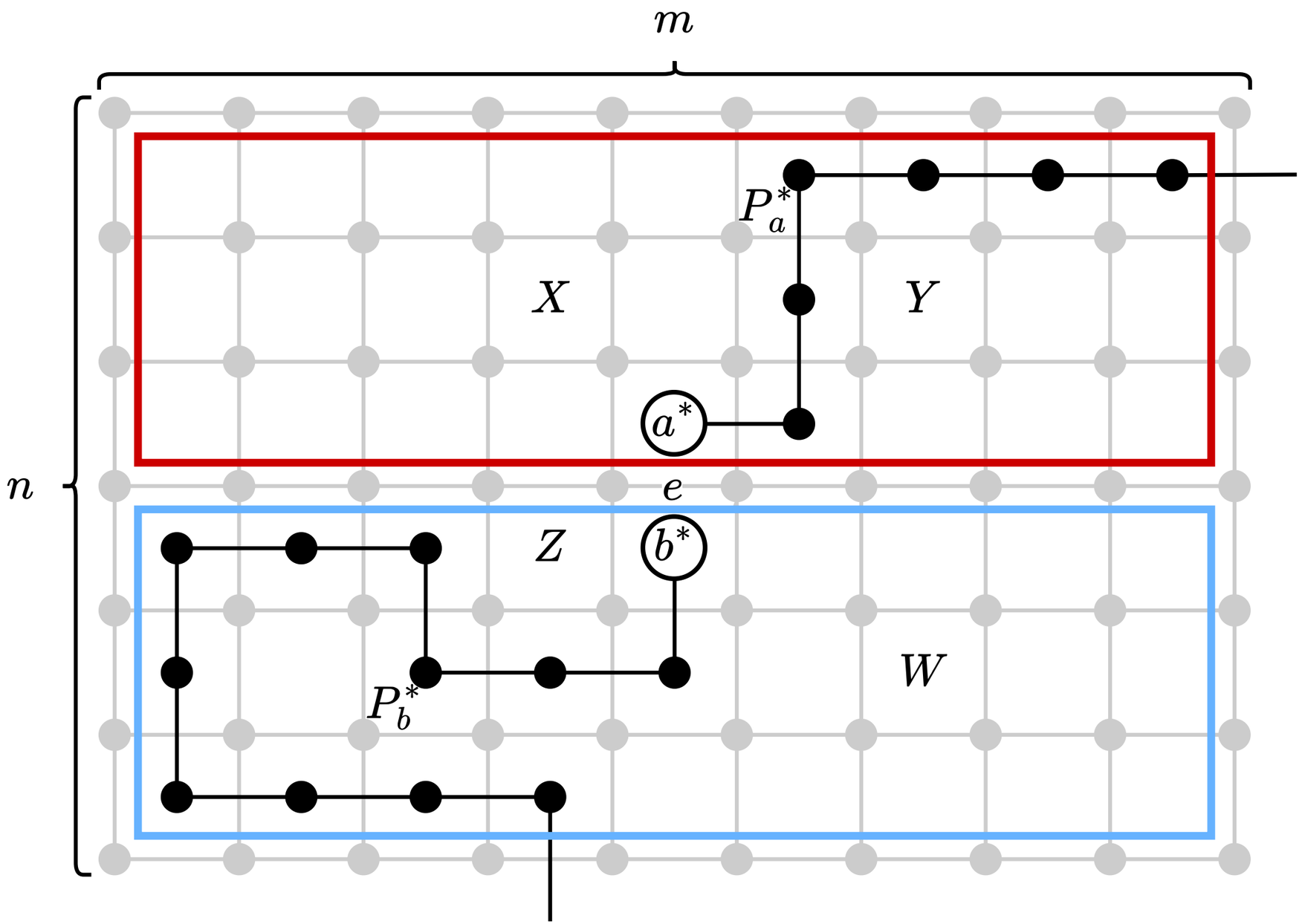
1. *Sample a uniformly random **spanning tree***
2. *Try to find $k-1$ edges to remove to get a balanced forest, otherwise restart*

Note: This implies the conjecture! Rejection rates are related by polynomial factors for any fixed k.

Proof of probable splittability for $k = 2$

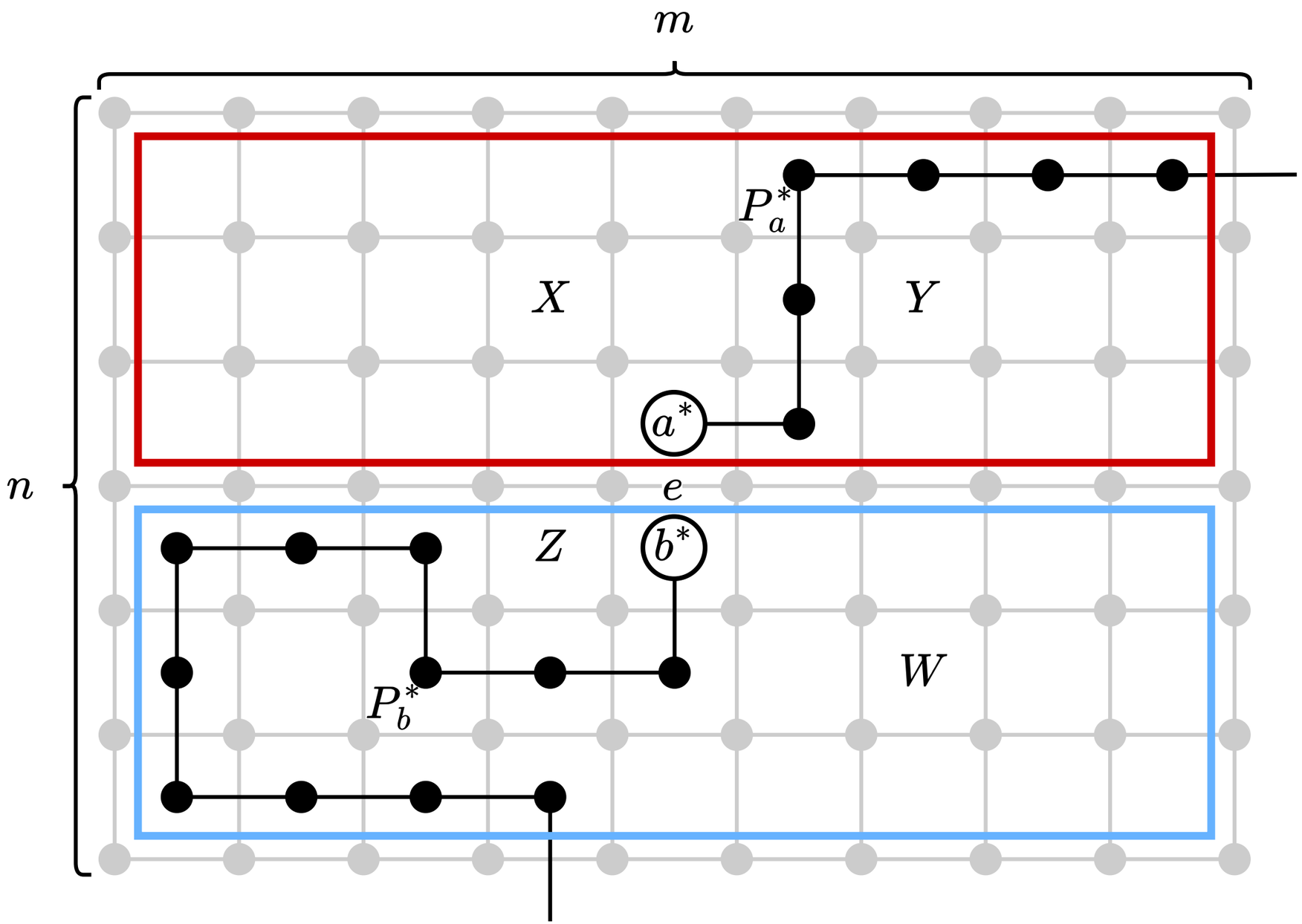


Proof of probable splittability for $k = 2$



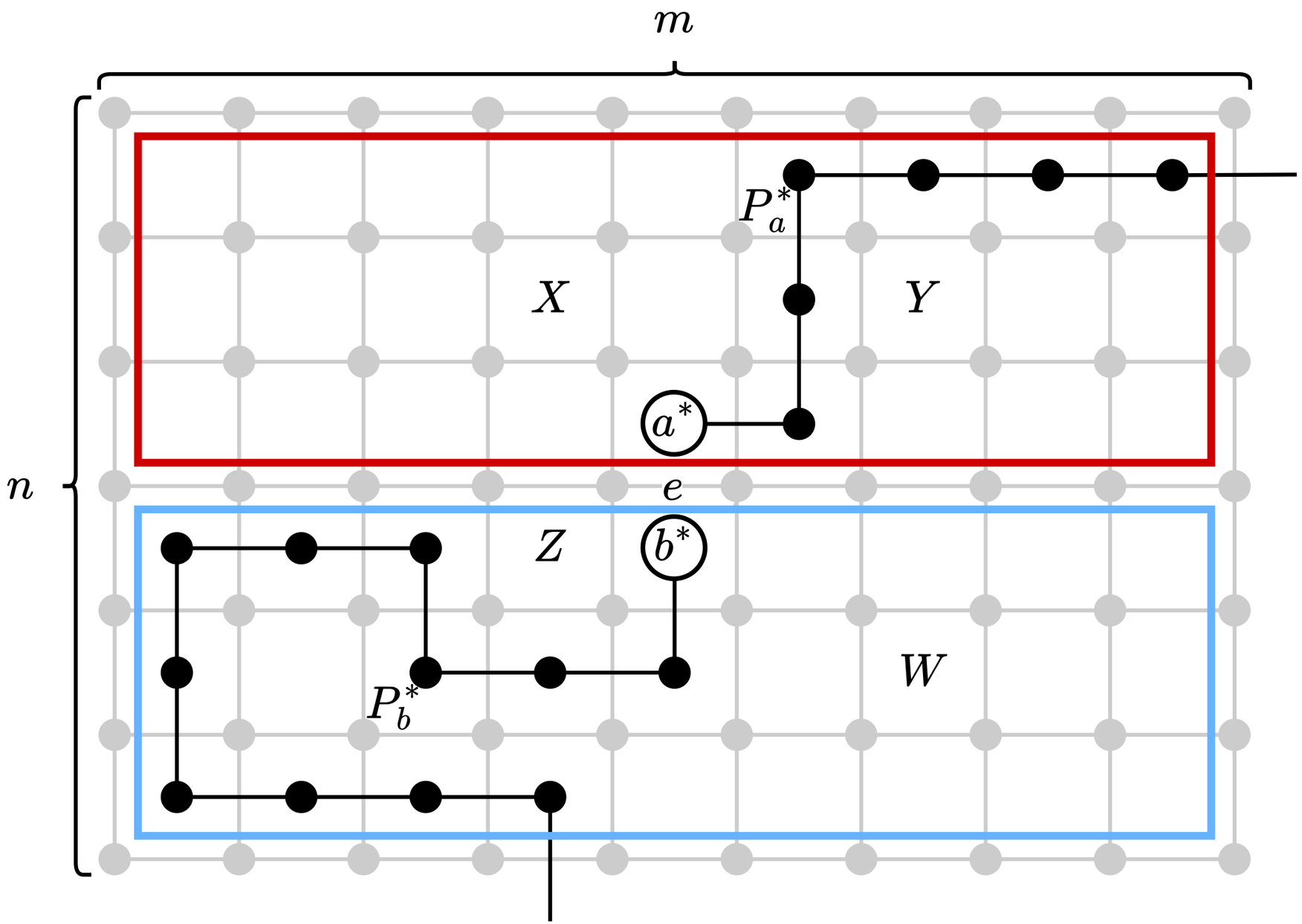
- With probability $\geq \frac{1}{n}$, the first walk stays within the top rectangle until it hits the boundary.

Proof of probable splittability for $k = 2$



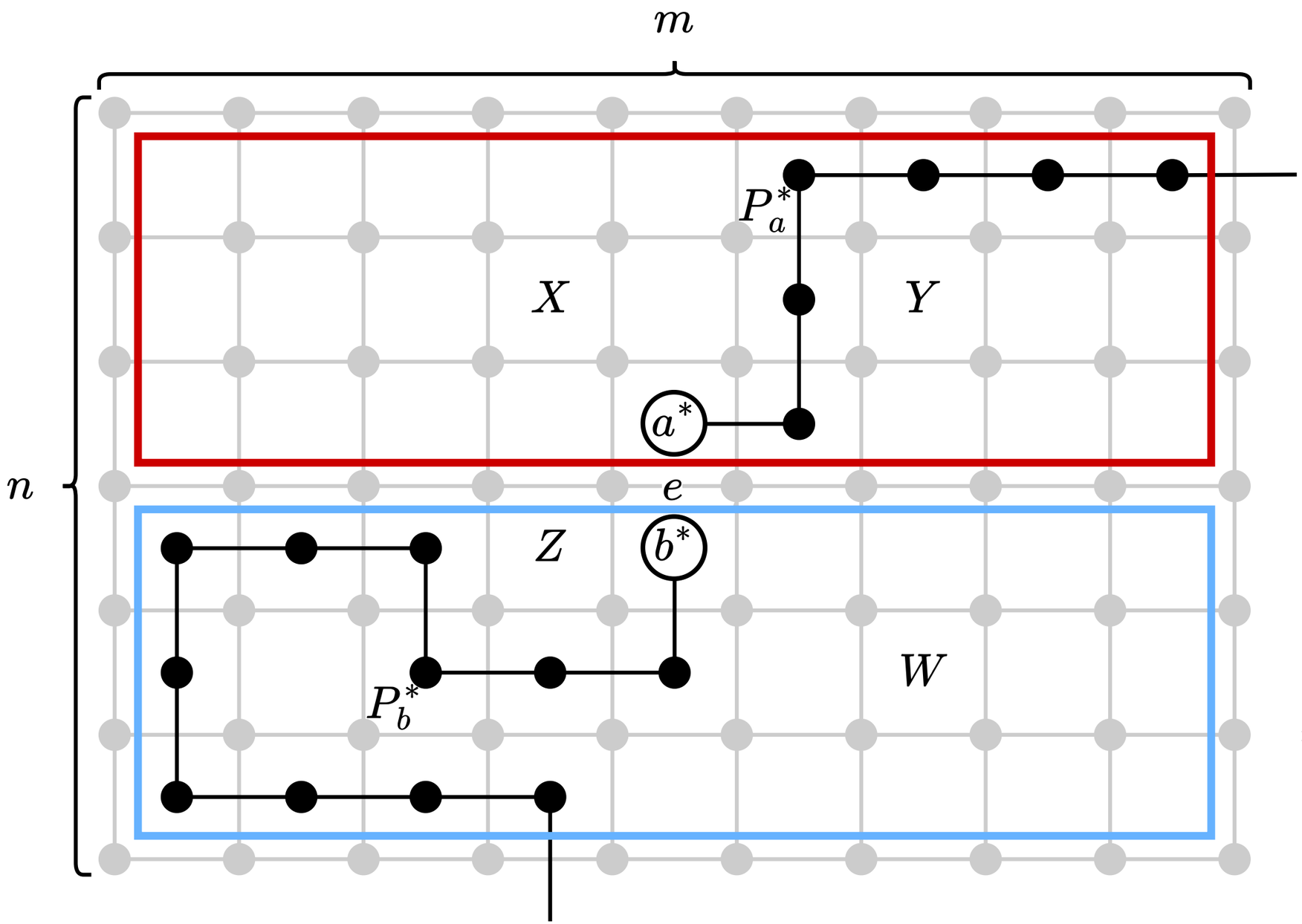
- With probability $\geq \frac{1}{n}$, the first walk stays within the top rectangle until it hits the boundary.
- With probability $\geq \frac{1}{n}$, the second walk stays within the bottom rectangle until it hits the boundary.

Proof of probable splittability for $k = 2$



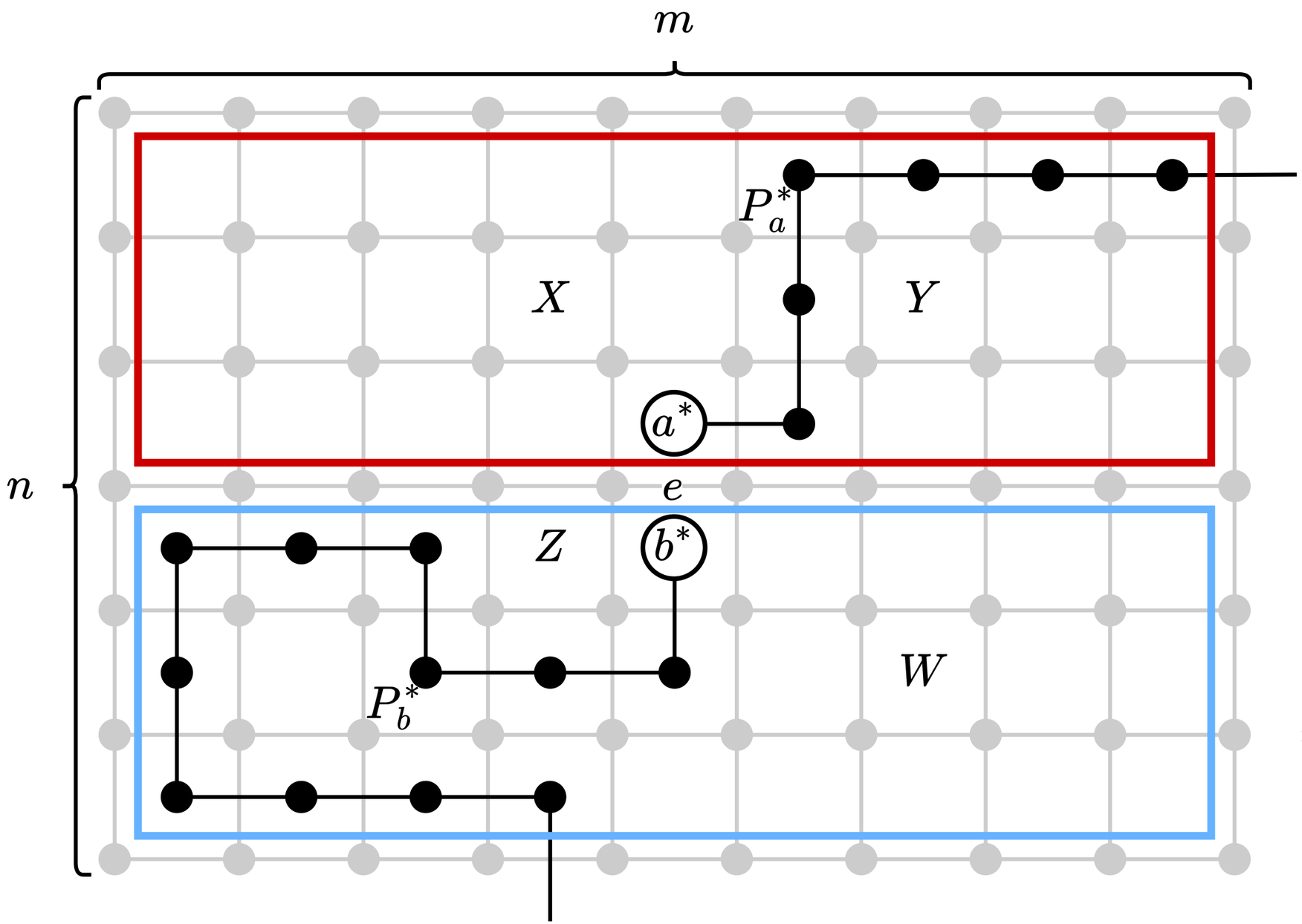
- With probability $\geq \frac{1}{n}$, the first walk stays within the top rectangle until it hits the boundary.
- With probability $\geq \frac{1}{n}$, the second walk stays within the bottom rectangle until it hits the boundary.
- With probability $\geq \frac{1}{mn}$, we have $X - Y = W - Z$

Proof of probable splittability for $k = 2$



- With probability $\geq \frac{1}{n}$, the first walk stays within the top rectangle until it hits the boundary.
- With probability $\geq \frac{1}{n}$, the second walk stays within the bottom rectangle until it hits the boundary.
- With probability $\geq \frac{1}{mn}$, we have $X - Y = W - Z$ $\implies X + Z = Y + W$

Proof of probable splittability for $k = 2$



- With probability $\geq \frac{1}{n}$, the first walk stays within the top rectangle until it hits the boundary.
- With probability $\geq \frac{1}{n}$, the second walk stays within the bottom rectangle until it hits the boundary.
- With probability $\geq \frac{1}{mn}$, we have $X - Y = W - Z$ $\Longrightarrow X + Z = Y + W$

Overall probability: $\frac{1}{mn^3}$ ■

Proof credit: STOC conference hotel lobby carpet

Speeding up tree proposal

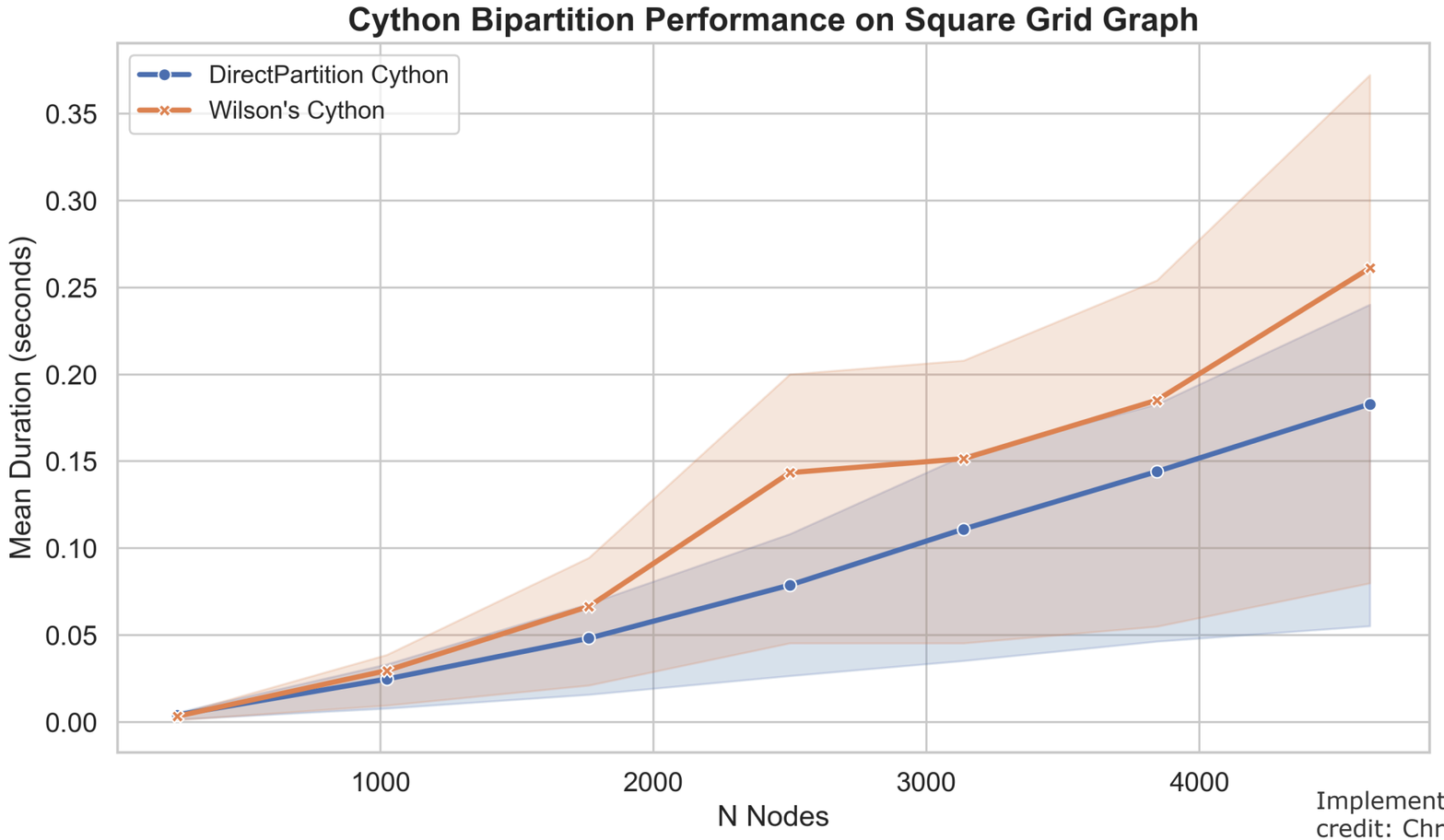
Theorem (Cannon, Pegden, T-F, working paper)

Over any family of n-vertex "grid-like graphs" we can simulate drawing a random spanning tree and finding the bipartition it generates (if one exists) in time $O(n)$.

Speeding up tree proposal

Theorem (Cannon, Pegden, T-F, working paper)

Over any family of n-vertex "grid-like graphs" we can simulate drawing a random spanning tree and finding the bipartition it generates (if one exists) in time $O(n)$.



Implementation and image credit: Christopher Pankow

State-space connectivity

A Markov chain *mixes* when the total variation distance between its stationary distribution and the distribution after running from any given initial state is less than some small constant.

State-space connectivity

A Markov chain *mixes* when the total variation distance between its stationary distribution and the distribution after running from any given initial state is less than some small constant.

Open Question: For a constant number of districts, does ReCom mix on sufficiently large grid graphs in polynomial time?

State-space connectivity

A Markov chain *mixes* when the total variation distance between its stationary distribution and the distribution after running from any given initial state is less than some small constant.

Open Question: For a constant number of districts, does ReCom mix on sufficiently large grid graphs in polynomial time?

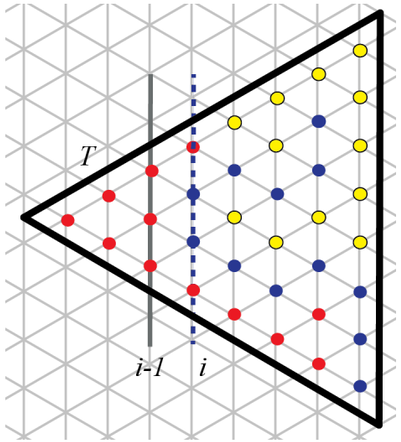
Open Question (should be easier??):
For three districts, does ReCom mix on sufficiently large grid graphs at all?

State-space connectivity

A Markov chain *mixes* when the total variation distance between its stationary distribution and the distribution after running from any given initial state is less than some small constant.

Open Question: For a constant number of districts, does ReCom mix on sufficiently large grid graphs in polynomial time?

Open Question (should be easier??):
For three districts, does ReCom mix on sufficiently large grid graphs at all?



Some positive results:

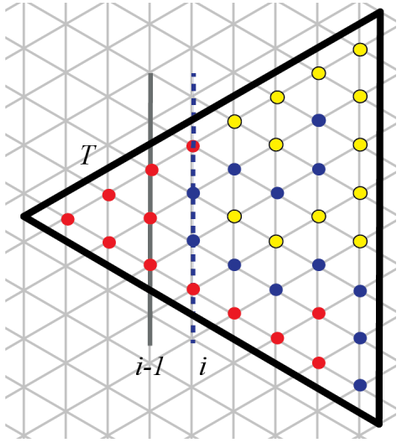
- (Cannon, 2023) This is true on a triangular subset of the triangular lattice, with 3 simply connected districts and ± 1 deviation from exact balance

State-space connectivity

A Markov chain *mixes* when the total variation distance between its stationary distribution and the distribution after running from any given initial state is less than some small constant.

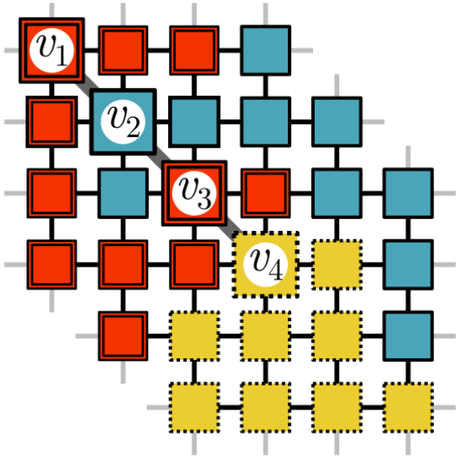
Open Question: For a constant number of districts, does ReCom mix on sufficiently large grid graphs in polynomial time?

Open Question (should be easier??):
For three districts, does ReCom mix on sufficiently large grid graphs at all?

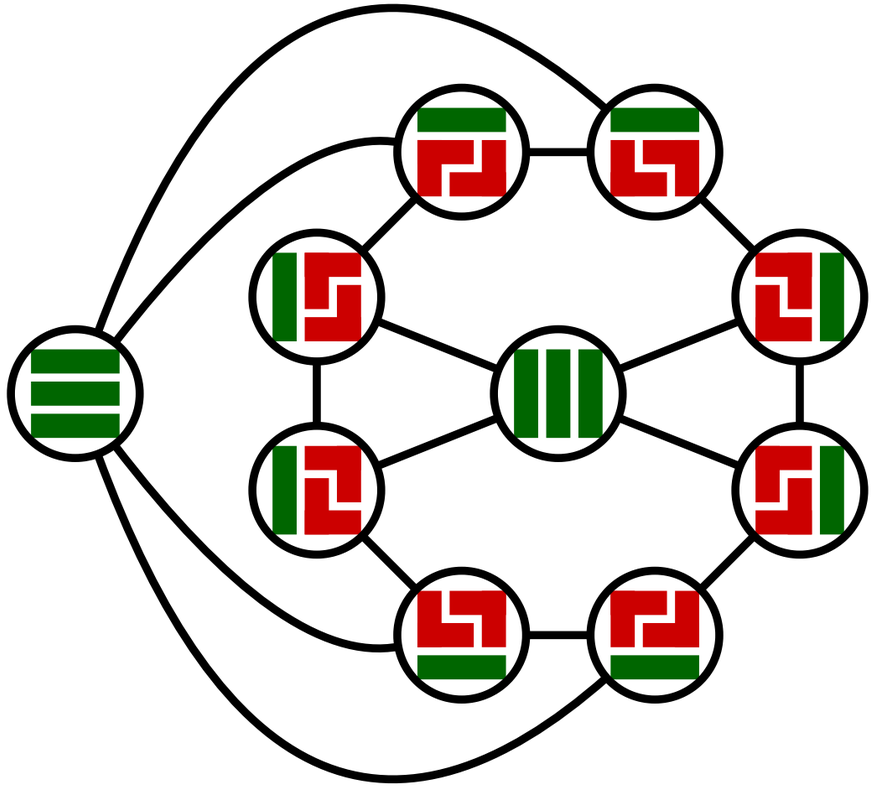


Some positive results:

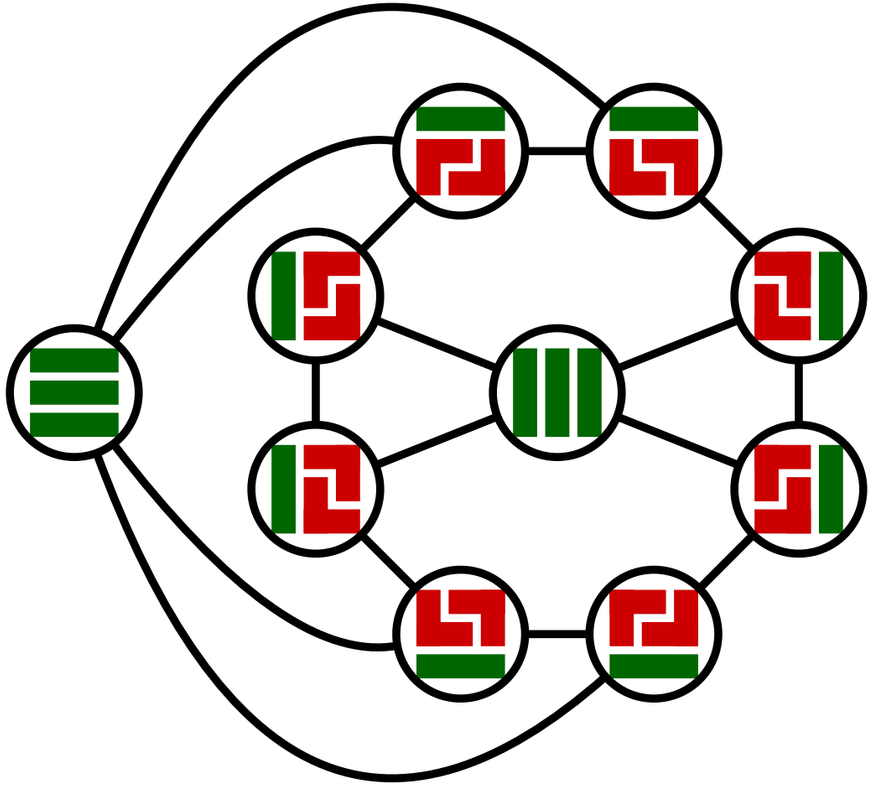
- (Cannon, 2023) This is true on a triangular subset of the triangular lattice, with 3 simply connected districts and ± 1 deviation from exact balance
- (Akitaya, Dituro, Gonczi, Korman, Souvaine, Stock, Tóth, 2026) Simpler proof, extends to square grids

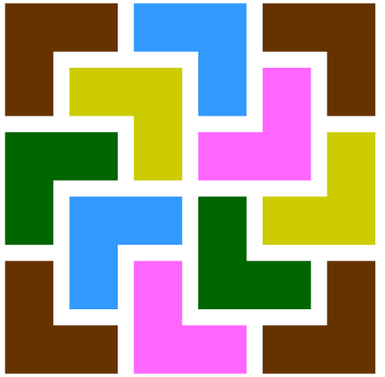


Metagraphs and locked tilings

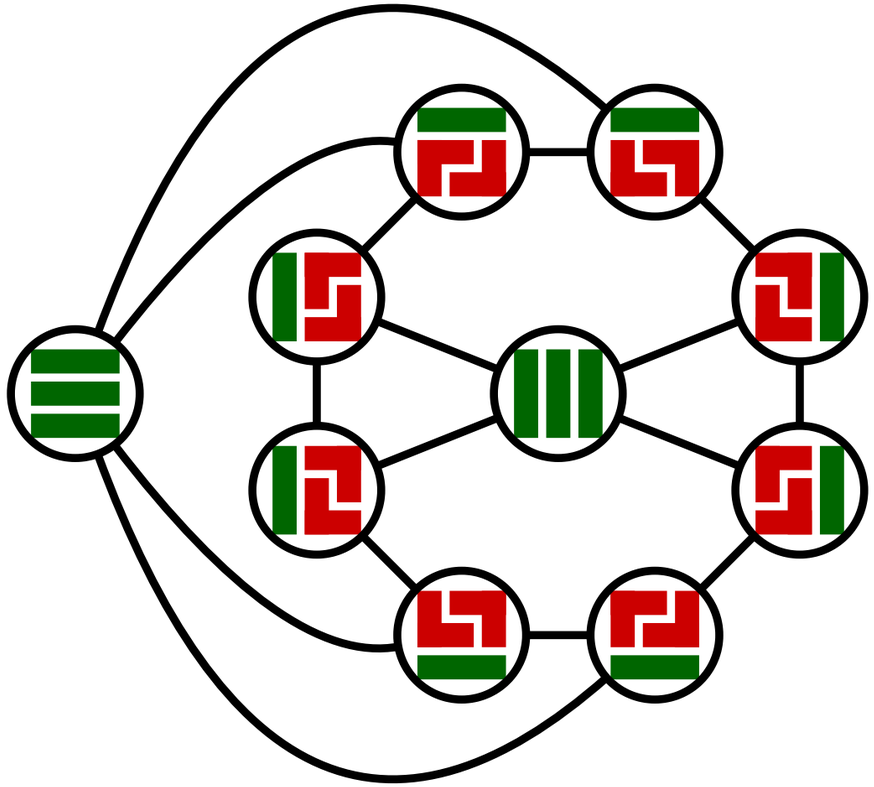


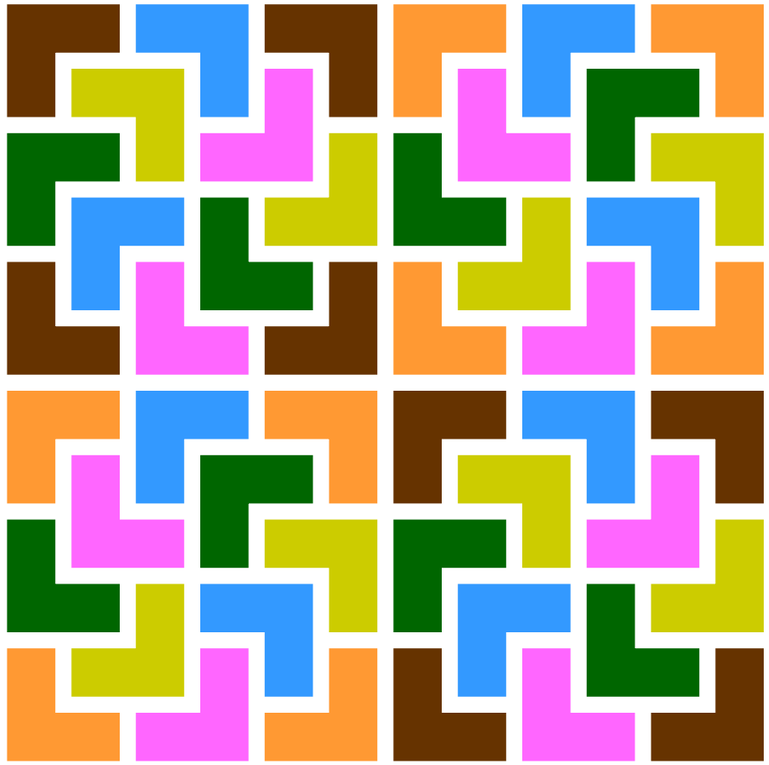
Metagraphs and locked tilings



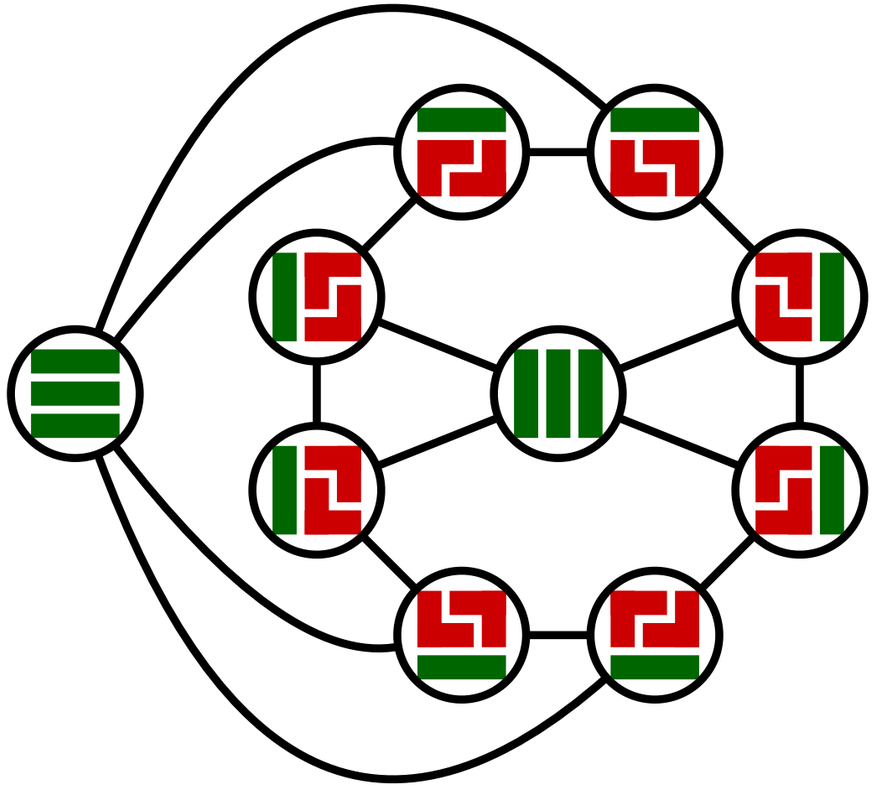


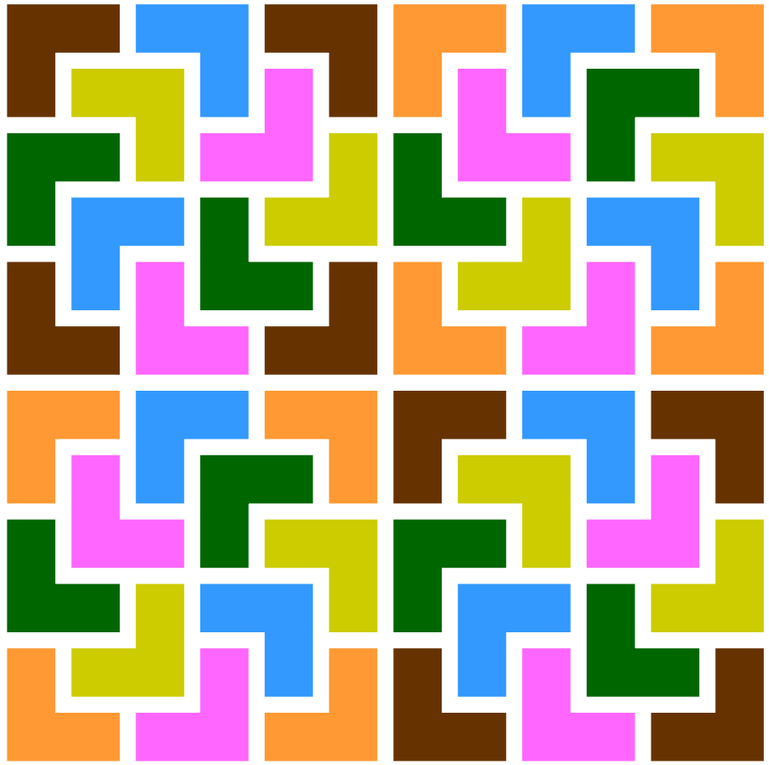
Metagraphs and locked tilings





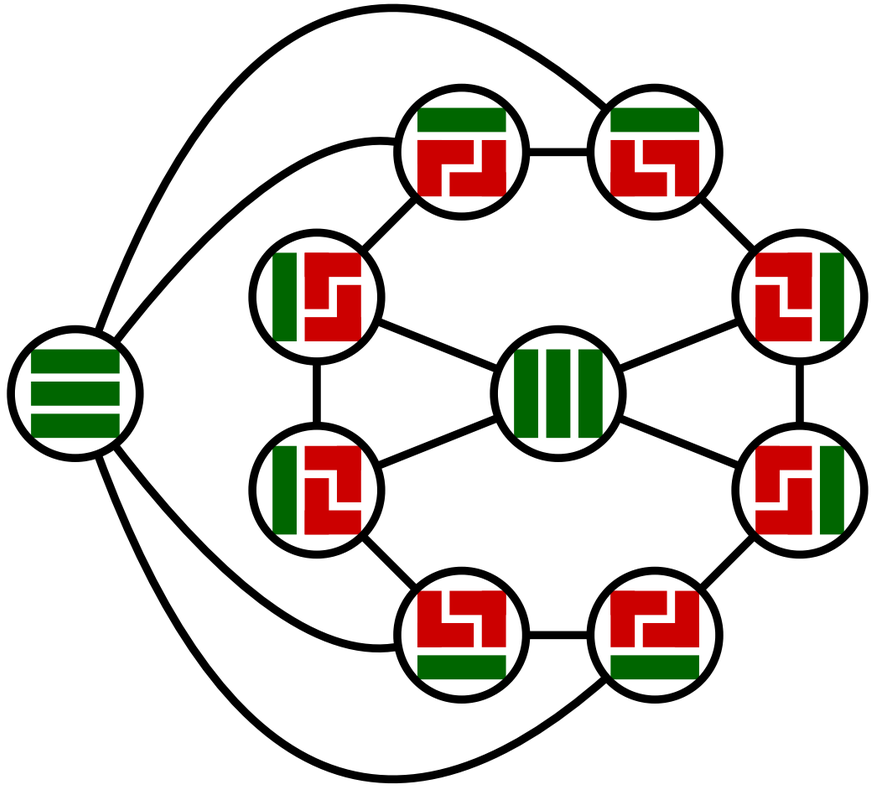
Metagraphs and locked tilings

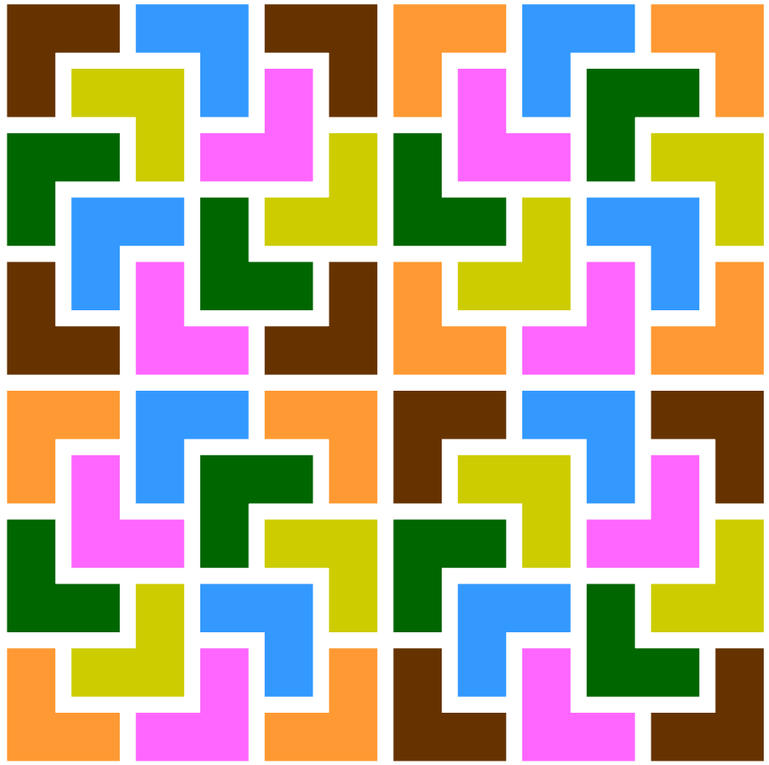




The best math puzzle of all time: Find a locked 4-omino tiling on a square grid.

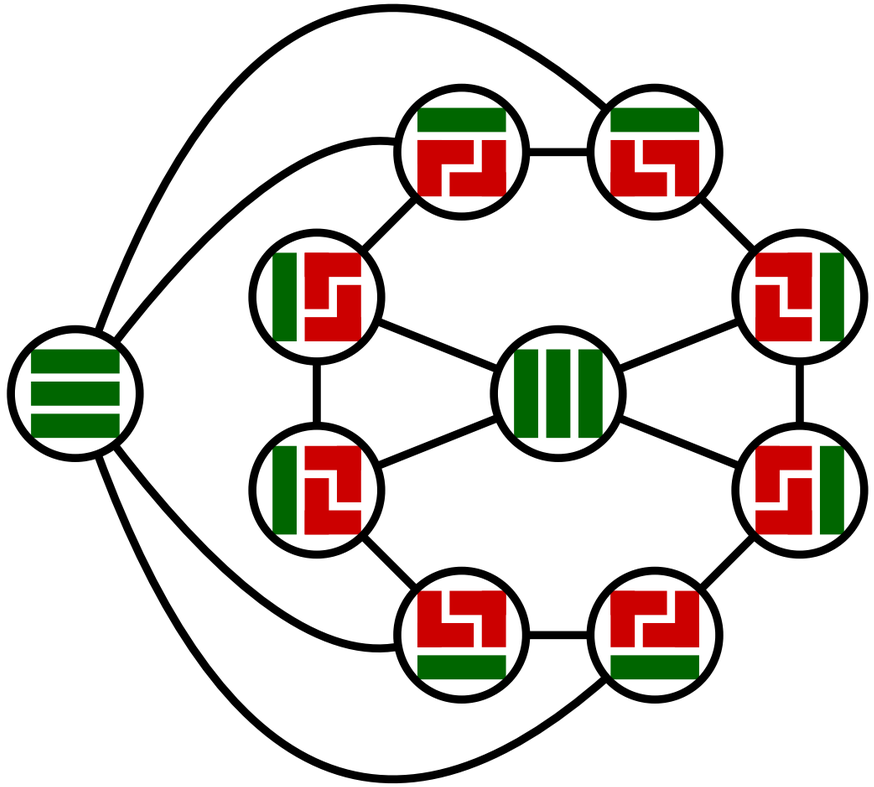
Metagraphs and locked tilings

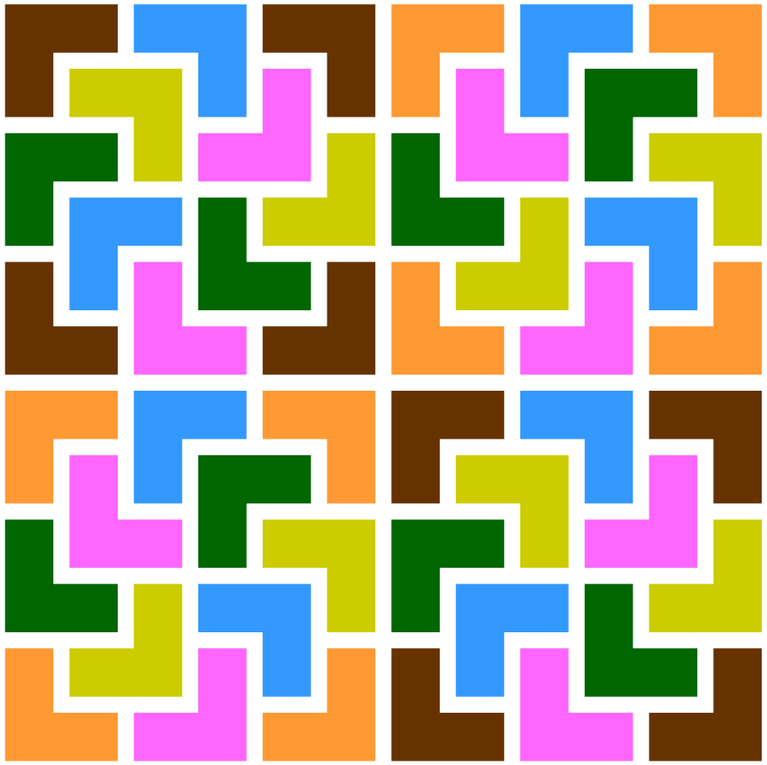




The best math puzzle of all time: Find a locked 4-omino tiling on a square grid.
In other words, the tiling should have the property that:
If you pick up any two 4-ominoes, the only way to fill those 8 spaces is to use the same two tiles in the same configuration.

Metagraphs and locked tilings



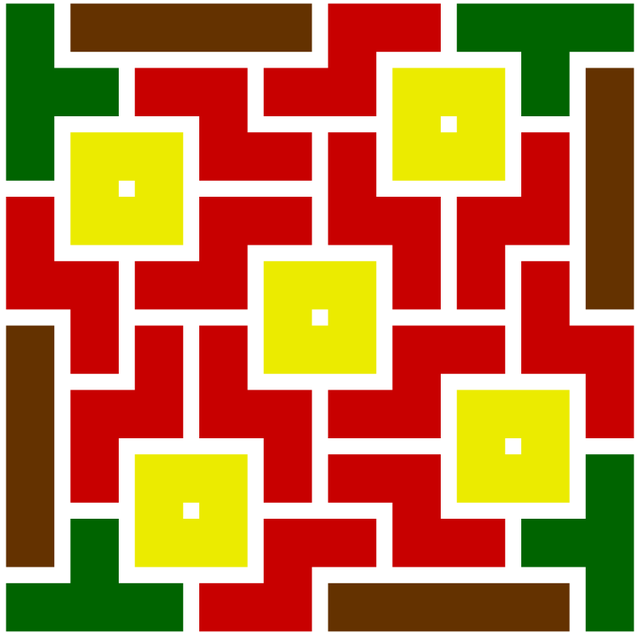


The best math puzzle of all time: Find a locked 4-omino tiling on a square grid.
In other words, the tiling should have the property that:
If you pick up any two 4-ominoes, the only way to fill those 8 spaces is to use the same two tiles in the same configuration.
Hint: The smallest example is on a 10x10 grid, and it's **unique** up to symmetries.

The smallest locked 4-omino tiling

(Spoiler alert: Answer on next slide)

The smallest locked 4-omino tiling



The second smallest (known) locked 4-omino tiling