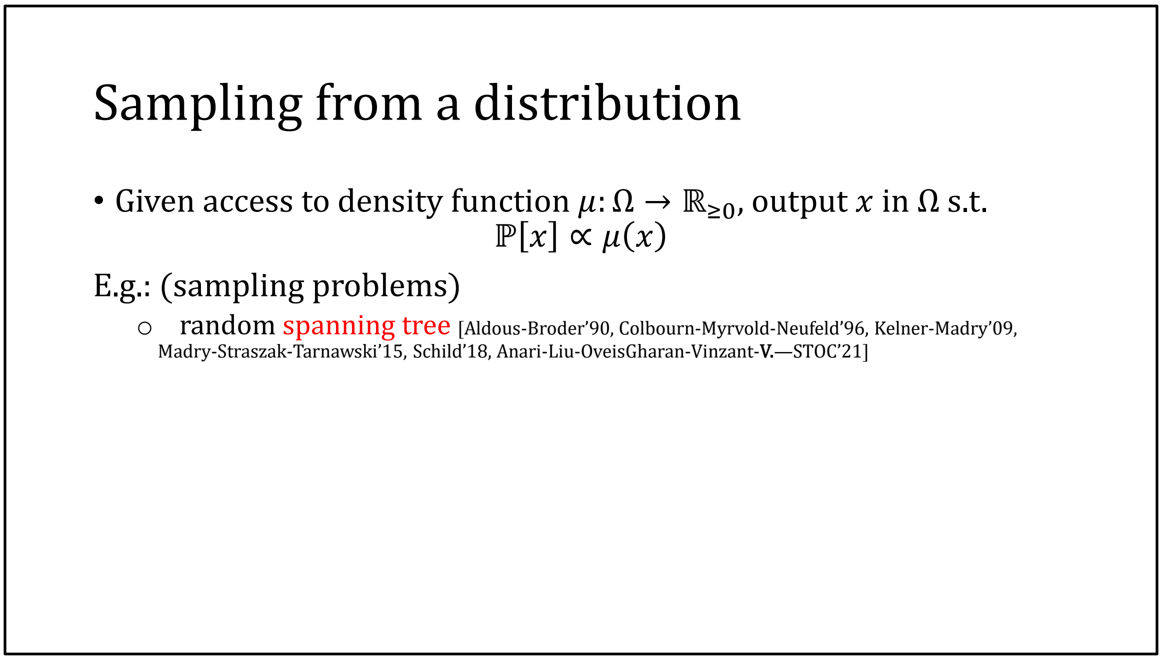
Optimal Sublinear Sampling of Spanning Trees and Determinantal Point Processes via Average-Case Entropic Independence

Thuy-Duong Vuong

Joint with Nima Anari Yang Liu



Sampling from a distribution

- Given access to density function $\mu : \Omega \to \mathbb{R}_{\geq 0}$, output x in Ω s.t.

$$\mathbb{P}[x] \propto \mu(x)$$

E.g.: (sampling problems)

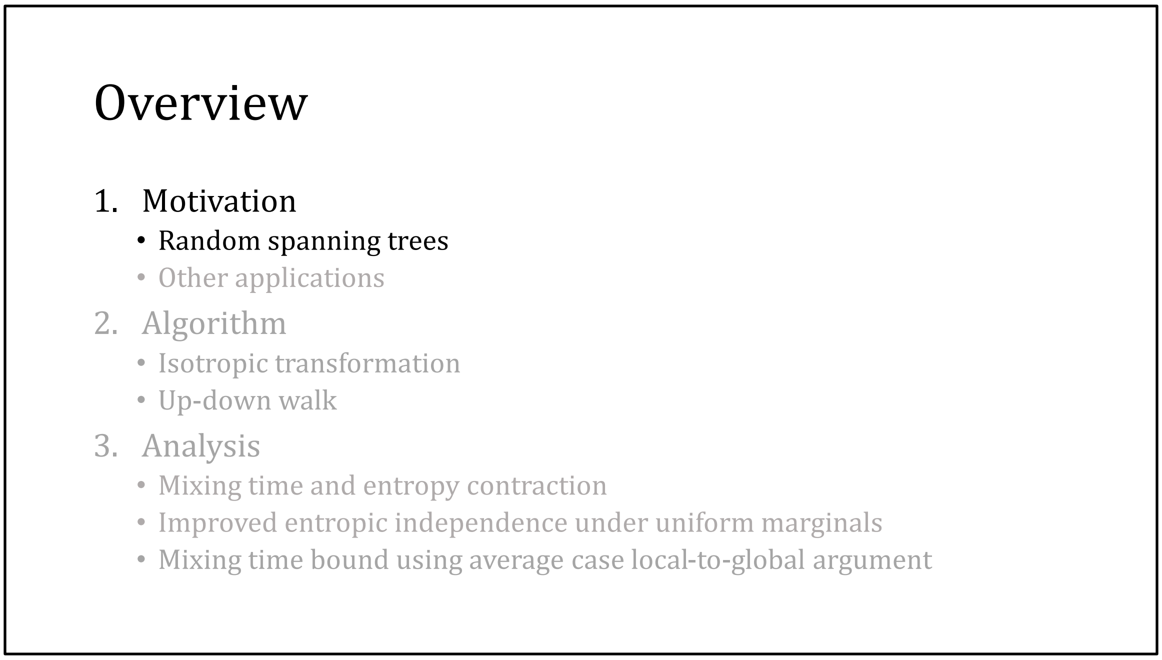
- random spanning tree [Aldous-Broder'90, Colbourn-Myrvold-Neufeld'96, Kelner-Madry'09, Madry-Straszak-Tarnawski'15, Schild'18, Anari-Liu-OveisGharan-Vinzant-**V.**—STOC'21]

Today we will focus on the problem of sampling from a given distribution. By given, I mean that you have oracle access to the density function of that distribution. Examples of sampling problem include sampling a random spanning tree in a graph. This is also the main problem we will focus on today.

Sampling from a distribution

- Given access to density function $\mu: \Omega \to \mathbb{R}_{\geq 0}$, output x in Ω s.t.
$$\mathbb{P}[x] \propto \mu(x)$$
- Sufficient to approximately sample i.e. output x according to $\widehat{\mathbb{P}}$ s.t.
$$d_{TV}(\mathbb{P}, \widehat{\mathbb{P}}) = \sum |\widehat{\mathbb{P}}(x) - \mathbb{P}(\mathrm{x})| < 0.01$$

It is usually enough to approximately sample, meaning having the output distribution P hat close in TV distance to the target distribution P.

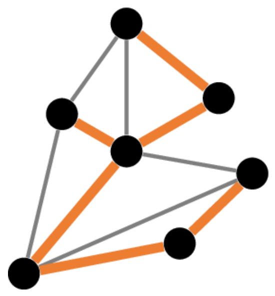


Overview

1. Motivation
   - Random spanning trees
   - Other applications
2. Algorithm
   - Isotropic transformation
   - Up-down walk
3. Analysis
   - Mixing time and entropy contraction
   - Improved entropic independence under uniform marginals
   - Mixing time bound using average case local-to-global argument

Spanning tree

Given graph G = G(V,E), $T \subseteq E$ is a spanning tree of G if T has no cycle and $|T| = |V| - 1$.

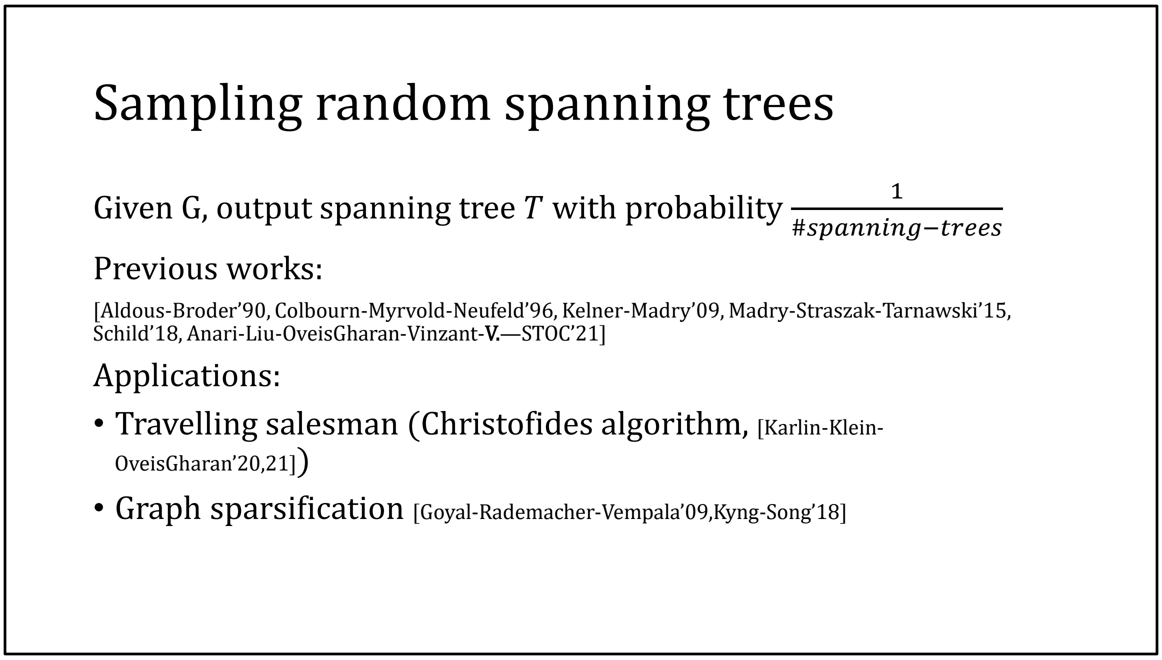


Given a graph, a spanning tree is a subset T of edges in the graph that form a connected and cycle-free subgraph. This implies the size of T is exactly number of vertices - 1

Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning-trees}$

We consider the problem of sampling from the uniformly random distribution over spanning trees.



Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning-trees}$

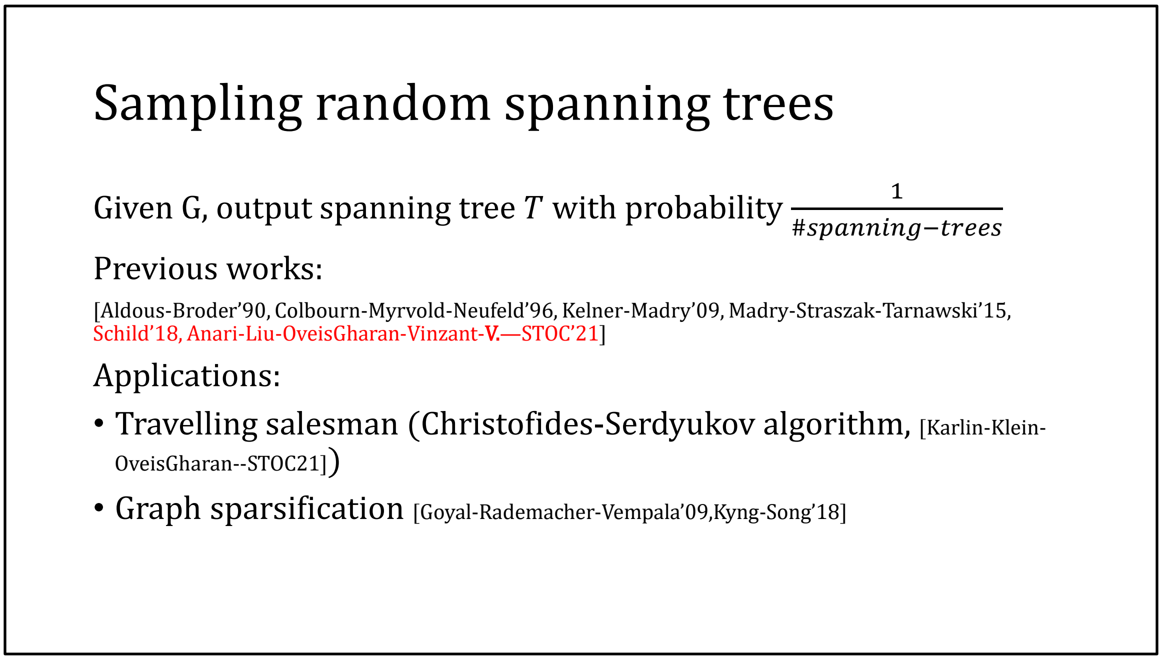
Previous works:

[Aldous-Broder'90, Colbourn-Myrvold-Neufeld'96, Kelner-Madry'09, Madry-Straszak-Tarnawski'15, Schild'18, Anari-Liu-OveisGharan-Vinzant-**V.**—STOC'21]

Applications:

- Travelling salesman (Christofides algorithm, [Karlin-Klein-OveisGharan'20,21])
- Graph sparsification [Goyal-Rademacher-Vempala'09,Kyng-Song'18]

This problem has been very well studied since the 1990s. Theres a long line of work starting from Aldous-Broder in the 90s to now.



Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning{-}trees}$

Previous works:

[Aldous-Broder'90, Colbourn-Myrvold-Neufeld'96, Kelner-Madry'09, Madry-Straszak-Tarnawski'15, Schild'18, Anari-Liu-OveisGharan-Vinzant-V.—STOC'21]

Applications:

- Travelling salesman (Christofides-Serdyukov algorithm, [Karlin-Klein-OveisGharan--STOC21])
- Graph sparsification [Goyal-Rademacher-Vempala'09,Kyng-Song'18]

This problem has been very well studied since the 1990s. Theres a long line of work starting from Aldous-Broder in the 90s to now. Note that the last two work achieve linear runtime. Aaron Schild's work in 2018 follows from the long line of work starting from the 90s that involves like hundred of pages of proof, why ours work use a very simple algorithm that can be stated in 3 lines, and the analysis is like two pages.

The problem of sampling random spanning tree has also found applications in other area of CS, for example, in approximation for the travelling salesman problem. I hope you are familiar with TSP? So given a graph, the travelling salesman problem ask you to find the shortest possible route that visits each city exactly once and returns to the origin city.
One approximation alg for the TSP problem is called the christofides algorithm, where first you find a minimum weight spanning tree of the graph, when you do some modification to the tree to get a tour. This gives you a 3/2 approximation for the travelling salesman problem, and very recently Karlin-Klein-OveisGharan modify the algorithm to sample spanning tree according to some distribution (instead of finding just one spanning tree), and through that they get an improved approximation factor of 3/2-10^{-40}
Another application is in graph sparsification, where given G we want to create a

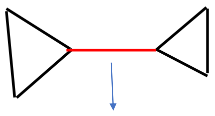
much smaller subgraph of G that preserves properties of G like size of cuts.

Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning{-}trees}$

To find one spanning tree, need $\Omega(|E|)$ time.

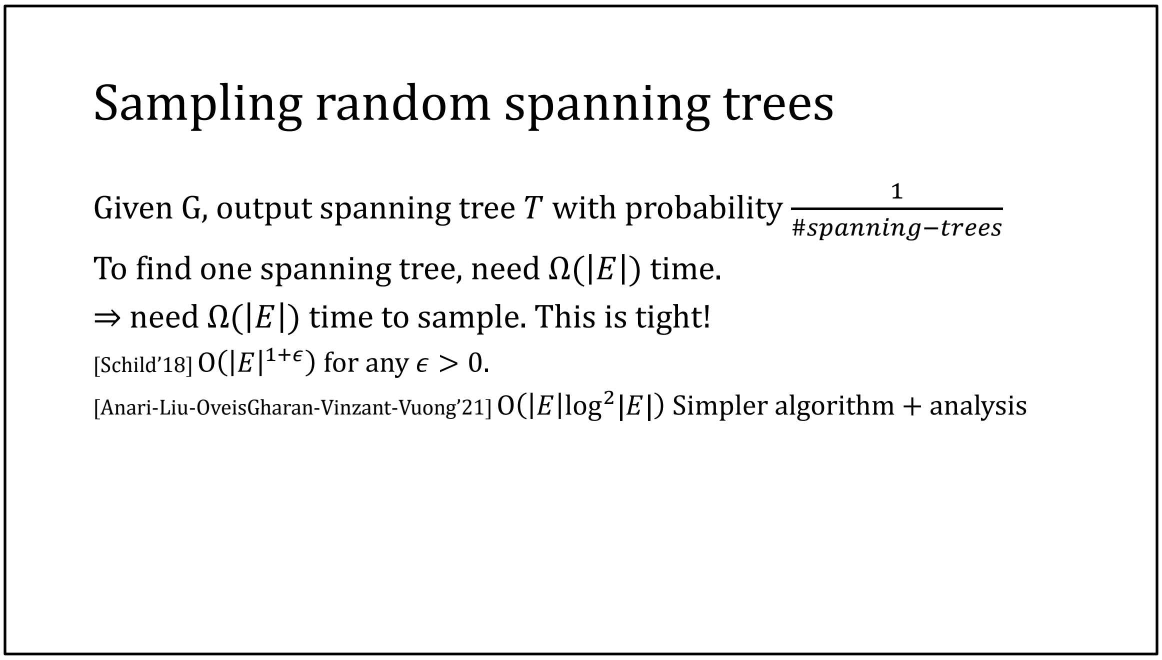
$\Rightarrow$ need $\Omega(|E|)$ time to sample



Bridge: edge whose removal disconnect the graph

Every spanning tree need to contain the red bridge edge

Okay, so now we establish that sampling random spanning tree is an important problem.
Now, what is the time complexity of this problem? Well, I claim that we need at least linear time in the number of edges to even find one spanning tree.
For example, suppose the graph contains a bridge edge, meaning an edge whose removal will disconnect the graph, then every spanning tree need to contain this bridge. But you need to read the entire graph to find this bridge.
If you can sample spanning tree, then you must also be able to find one spanning tree. So sampling is always a harder problem then search. Now since search takes at least linear time in the number of edges, so must sampling.



Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning-trees}$

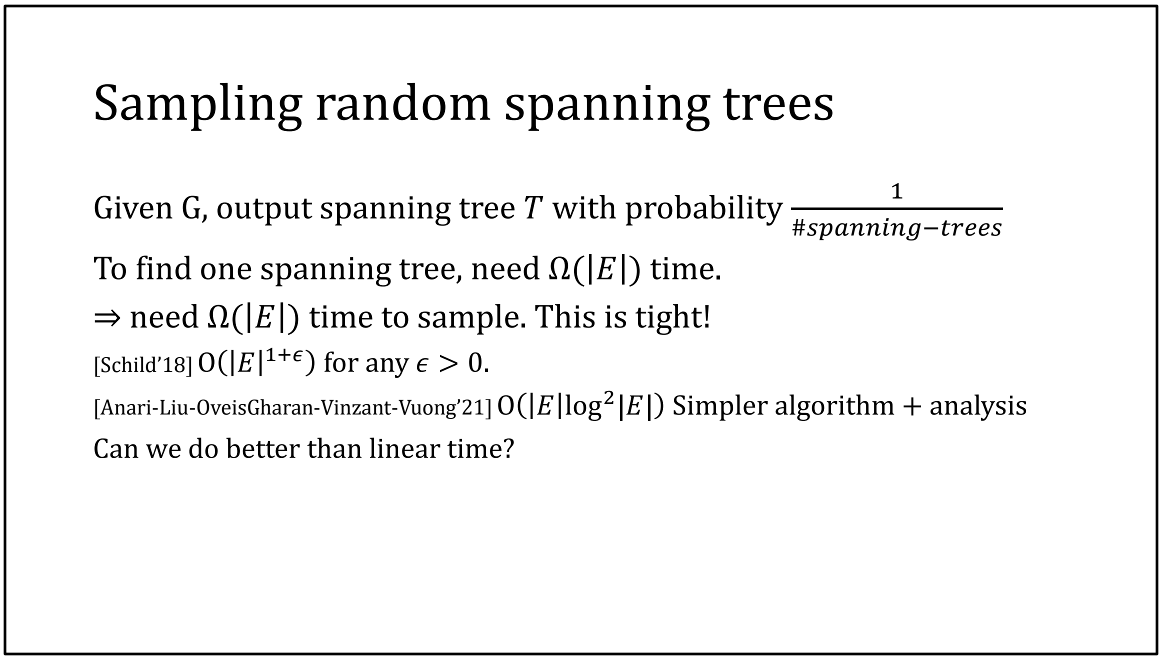
To find one spanning tree, need $\Omega(|E|)$ time.

$\Rightarrow$ need $\Omega(|E|)$ time to sample. This is tight!

[Schild'18] $O(|E|^{1+\epsilon})$ for any $\epsilon > 0$.

[Anari-Liu-OveisGharan-Vinzant-Vuong'21] $O(|E|\log^2|E|)$ Simpler algorithm + analysis

Surprisingly, this is tight, meaning that you can sample a random spanning tree in time roughly linear in the number of edges. There are 2 algo that achieves this runtime, one by Aaron schild that is the result of a long line of works start from the 90, and one standalone 20-pages paper by Nima, Kuikui, Shayan, Cynthia and myself that use a very simple algorithm and analysis and even get a slightly faster runtime.



Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning-trees}$

To find one spanning tree, need $\Omega(|E|)$ time.

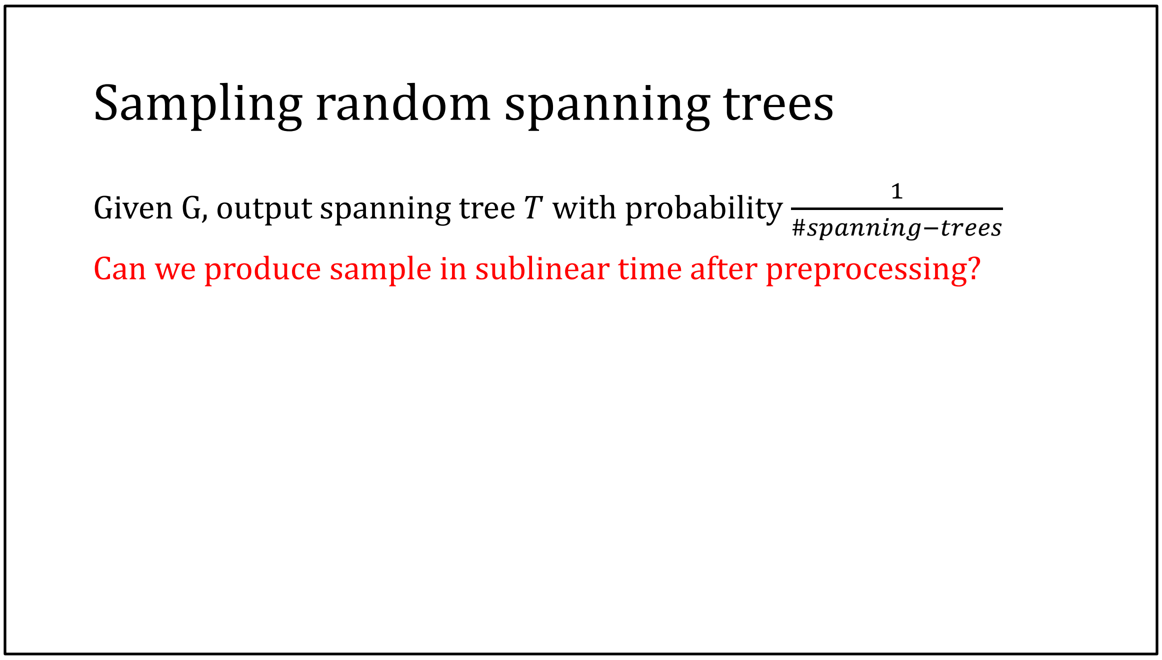
$\Rightarrow$ need $\Omega(|E|)$ time to sample. This is tight!

[Schild'18] $O(|E|^{1+\epsilon})$ for any $\epsilon > 0$.

[Anari-Liu-OveisGharan-Vinzant-Vuong'21] $O(|E| \log^2 |E|)$ Simpler algorithm + analysis

Can we do better than linear time?

So that's nice.
Okay here's a kicker: can we sample in sublinear time?

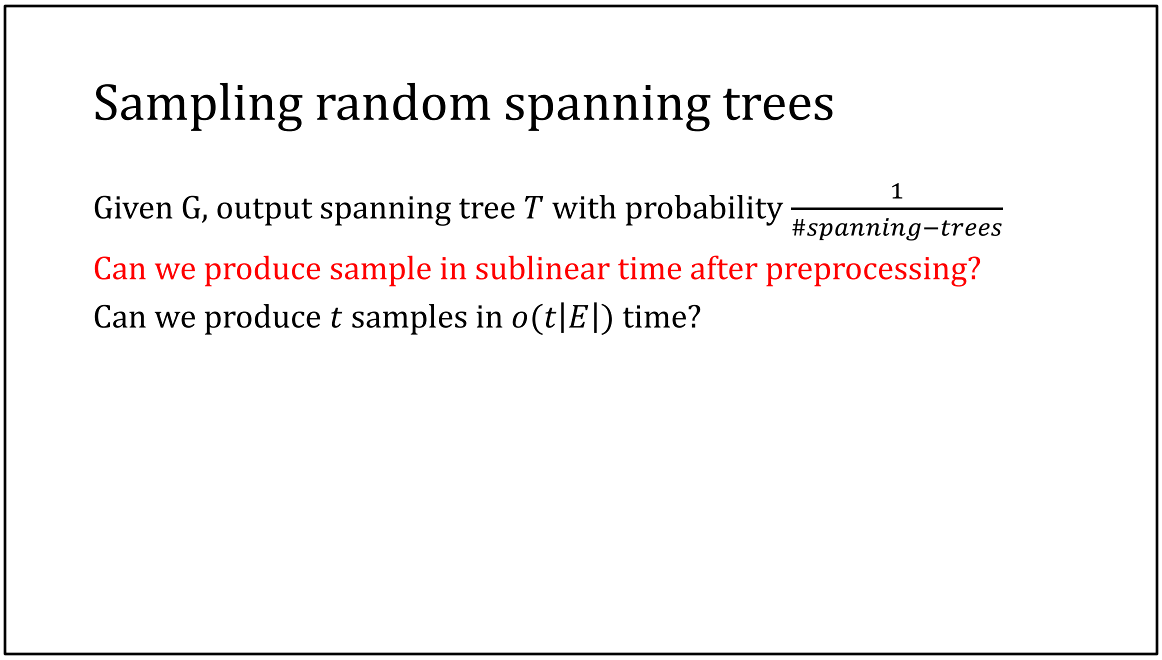


Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning-trees}$

Can we produce sample in sublinear time after preprocessing?

Or more precisely, can we sample in sublinear time after some preprocessing



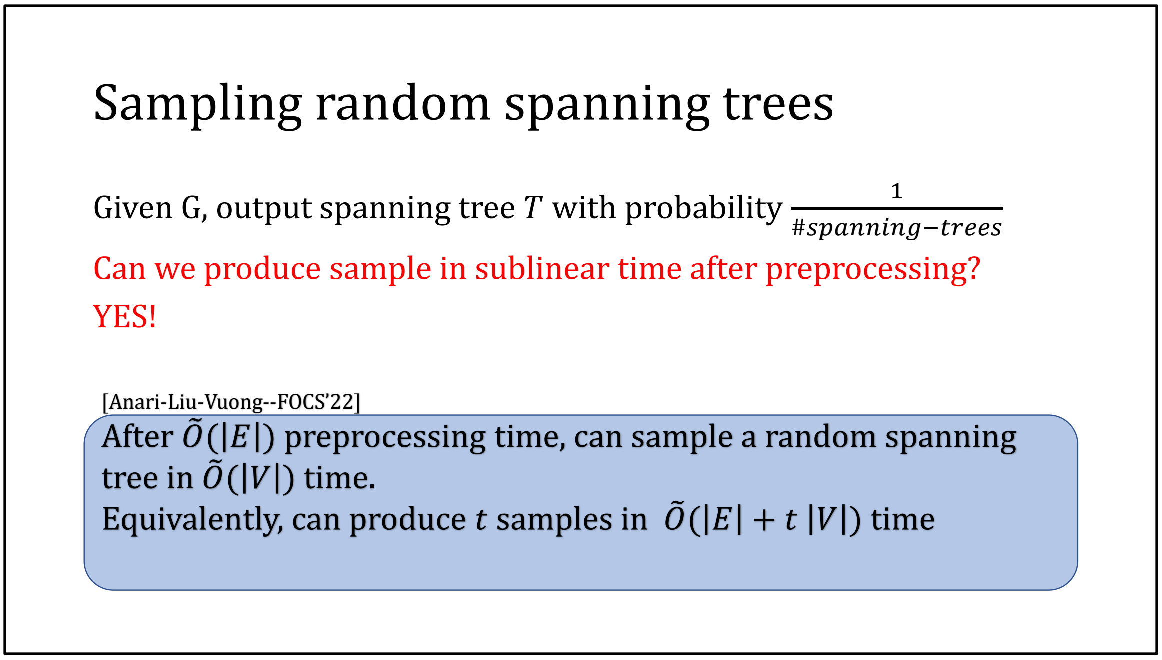
Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning{-}trees}$

Can we produce sample in sublinear time after preprocessing?

Can we produce t samples in $o(t|E|)$ time?

Another way to view this question is, can we achieve some economy of scale? So if we want to produce t random spanning tree, do our algorithm needs to run in time k times the number of edges, or can we do better?



Sampling random spanning trees

Given G, output spanning tree T with probability $\frac{1}{\#spanning{-}trees}$

Can we produce sample in sublinear time after preprocessing?

YES!

[Anari-Liu-Vuong--FOCS'22]

After $\tilde{O}(|E|)$ preprocessing time, can sample a random spanning tree in $\tilde{O}(|V|)$ time.
Equivalently, can produce t samples in $\tilde{O}(|E| + t\ |V|)$ time

Heres our main result. After once time processing of quasi-linear cost in the number of edges, we can produce subsequent sample of random spanning tree in time linear in the number of vertices.
Equivalently, we can produce t samples in $\tilde{O}(|E| + t\ |V|)$ time

Sampling random spanning trees

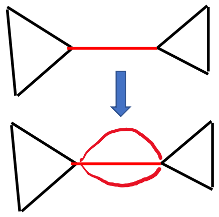
Given G, output spanning tree T with probability $\frac{1}{\#spanning-trees}$

Can we produce sample in sublinear time after preprocessing?

YES!

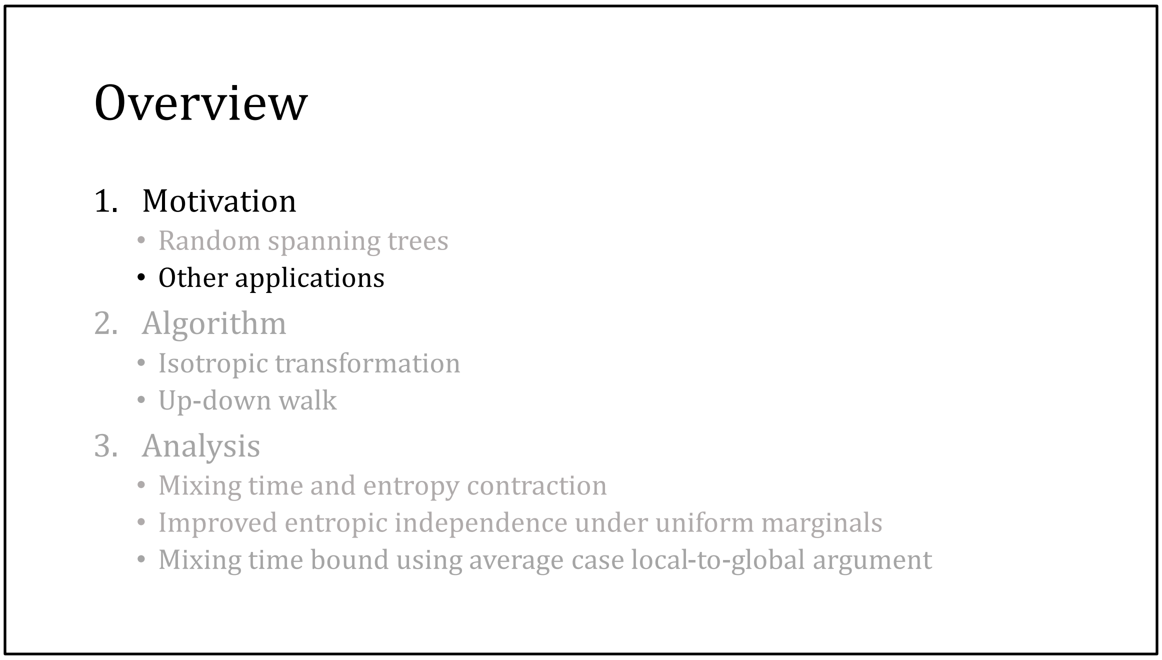
Idea:

1. Make all edges equal (having the same marginal under the uniform spanning tree distributions) via subdividing edges
1. Apply sampling algorithm from [Anari-Liu-OveisGharan-Vinzant-**V.**--STOC21] (up-down walk)



SO here's the idea for the algorithm. First step is the processing step, which we only need to do once, and cost time linear in the number of edges. In this step, we make all the edges equal in the sense that every edge is equally likely to be contained in a random spanning tree, and we achieve this by replacing an edges with multiple parallel copies of that edge.
After this preprocessing step, we will again apply the algorithm to sample spanning tree from the previous paper of Nima, Kuikui, Shayan, Cynthia and myself in STOC last year. In this paper, we will show that after the preprocessing step, this sampling algorithm takes time linear in the number of vertices instead in the number of edges.

Any question?



Overview

1. Motivation
   - Random spanning trees
   - Other applications
2. Algorithm
   - Isotropic transformation
   - Up-down walk
3. Analysis
   - Mixing time and entropy contraction
   - Improved entropic independence under uniform marginals
   - Mixing time bound using average case local-to-global argument

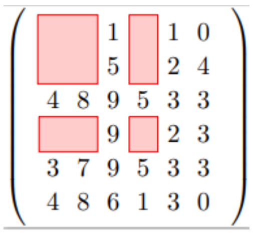
Our framework also applies to determinantal point processes, and generally all strongly Rayleigh distributions.

Other applications

Determinantal point processes (DPP):

(PSD) matrix $L \in \mathbb{R}^{n\times n}$: $\mu_L(S) = \det(L_{S,S})$ for $|S| = k$

Applications: randomized linear algebra, machine learning (recommender system)



S={1,2,4}

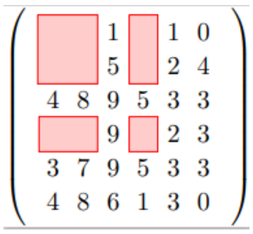
A DPP is defined by a symmetric positive semi-definite matrix L, and the probability of a set S under the DPP is proportional to the minor of L indexed by S. So here's an example. Lets say n = 6, so the matrix L is a 6 by 6 matrix, and k = 3. Lets say S = {1,2,4}, then the principle submatrix indexed by S is the matrix colored in red here.

Other applications

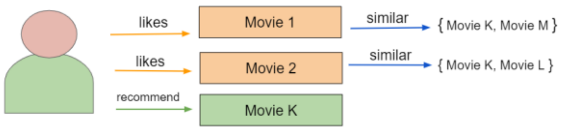
Determinantal point processes (DPP):

(PSD) matrix $L \in \mathbb{R}^{n \times n}$: $\mu_L(S) = \det(L_{S,S})$ for $|S| = k$

Applications: randomized linear algebra, machine learning (recommender system)



S={1,2,4}



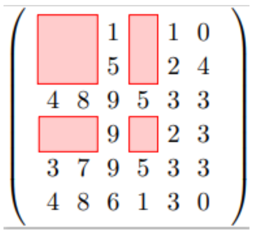
Determinantal point processes has application in randomized linear algebra, and generally in machine learning, for example, in recommendation system, say for recommending movies. The rough idea is to build the matrix L according to preference data of the user and attributes of the movies, then recommend a set of movies for the user

Other applications

Determinantal point processes (DPP):

(PSD) matrix $L \in \mathbb{R}^{n\times n}$: $\mu_L(S) = \det(L_{S,S})$ for $|S| = k$

Applications: randomized linear algebra, machine learning (recommender system)



[Anari-Liu-Vuong--FOCS'22]

After $\tilde{O}(nk^{\omega-1})$ preprocessing time, can sample from DPP in $\tilde{O}(k^{\omega})$ time.

Best time for first sample + (likely) optimal for subsequent samples b/c need $\theta(k^{\omega})$ to compute $\det(L_{S,S})$

For DPP, we show that after n k^{w-1} preprocessing time, can produce subsequent samples in time k^omega where omega is the matrix multiplication constant.
Note that the time it takes to even compute the determinant of a k by k matrix is already k^omega, so our runtime is likely optimal.
Even if we only want to produce 1 sample, then this runtime of n k^{omega -1} is already the best runtime currently.

Other applications

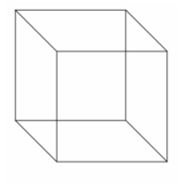
General strongly Rayleigh distributions:

$\mu: \binom{[n]}{k} \rightarrow \mathbb{R}_{\geq 0}$ is strongly Rayleigh if $f_\mu \neq 0$ when $Im(z_i) \geq 0$

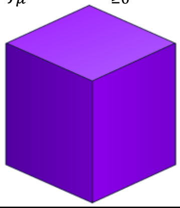
$$f_\mu(z_1, \dots, z_n) := \sum \mu(S) \prod_{i \in S} z_i$$

Results apply for general strongly Rayleigh distributions, including random spanning tree and DPPs

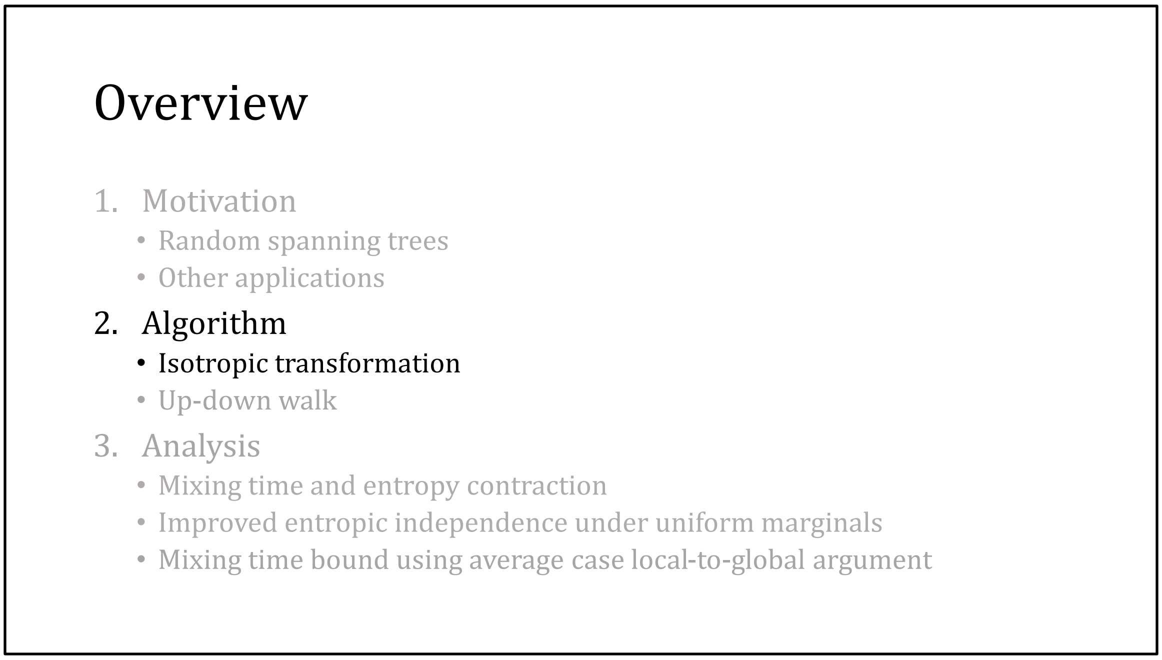
$\mu: \{0,1\}^n \rightarrow \mathbb{R}_{\geq 0}$



$f_\mu: \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$



Both the uniform distribution over spanning tree and determinantal point processes fall under the broader class of strongly Rayleigh distributions.
Each dist. Mu is associated with a multivariate polynomial called the generating polynomial of mu, defined as the sum of mu(S) multiply by the product of z_i for I in S.
You can think of f_{mu} as a way to extend the function mu defined on the vertices of the hypercube to the full hypercube
mu is called strongly Rayleigh if its generating poly doesn’t have root in the upper half of the complex plane.



Overview

1. Motivation
   - Random spanning trees
   - Other applications
2. Algorithm
   - Isotropic transformation
   - Up-down walk
3. Analysis
   - Mixing time and entropy contraction
   - Improved entropic independence under uniform marginals
   - Mixing time bound using average case local-to-global argument

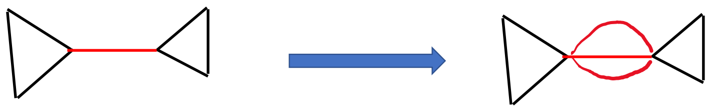
Okay so now we can describe the algorithm in more details

Isotropic transformation

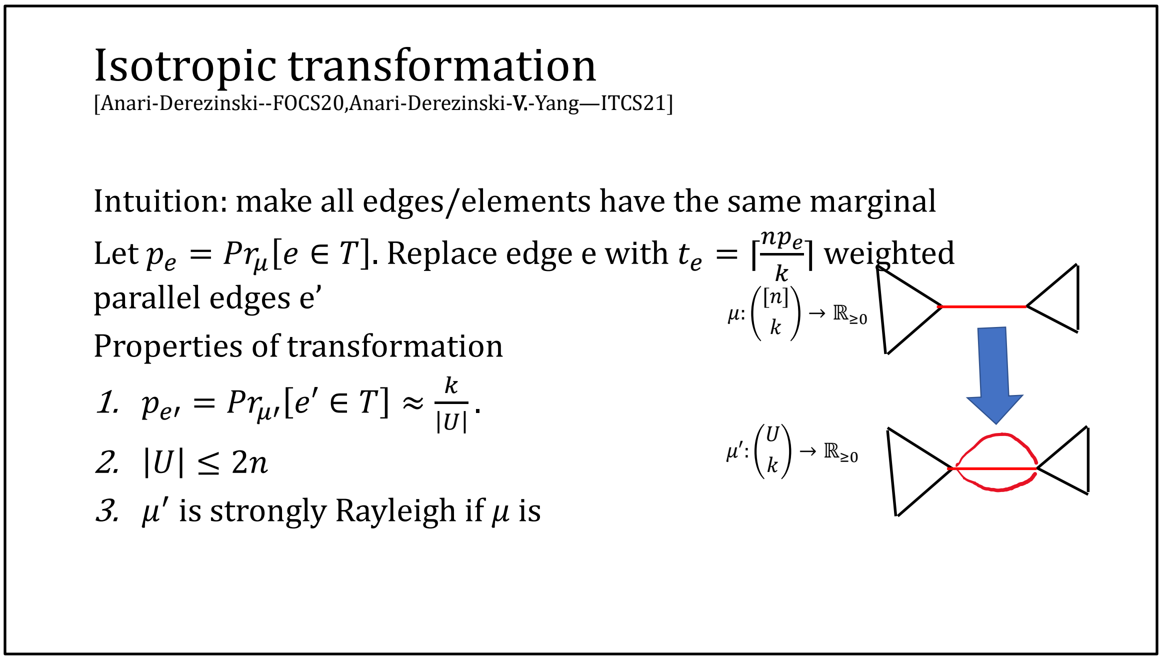
[Anari-Derezinski--FOCS20,Anari-Derezinski-**V.**-Yang—ITCS21]

Intuition: make all edges/elements have the same marginal

Let $p_e = Pr_\mu[e \in T]$. Replace edge e with $t_e = \lceil \frac{np_e}{k} \rceil$ parallel edges e' of weight $1/\ t_e$ each.



The first step is the preprocessing step, where we make the marginal of every edge equal under the spanning tree distribution. We call this step isotropic transformation, iso is a fancy word for being equal.
So recall that the marginal of an edge is the probability that it is included in random spanning tree. The marginal is also known as effective resistant or leverage score in the literature.
We can compute the marginals of all the edges in time linear in the number of edges via prior results. After we got the marginals, we can replace an edge e with t_e weighted parallel copy of e. The idea is to make each new parallel edge having the same probability of being included in a random spanning tree, and we want this probability to be k/n where k is the number of edges in a spanning tree and n is the number of edges in the graph.



Isotropic transformation

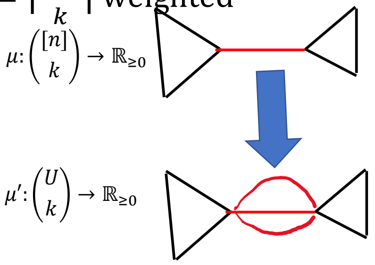
[Anari-Derezinski--FOCS20,Anari-Derezinski-**V.**-Yang—ITCS21]

Intuition: make all edges/elements have the same marginal

Let $p_e = Pr_\mu[e \in T]$. Replace edge e with $t_e = \lceil \frac{np_e}{k} \rceil$ weighted parallel edges e'

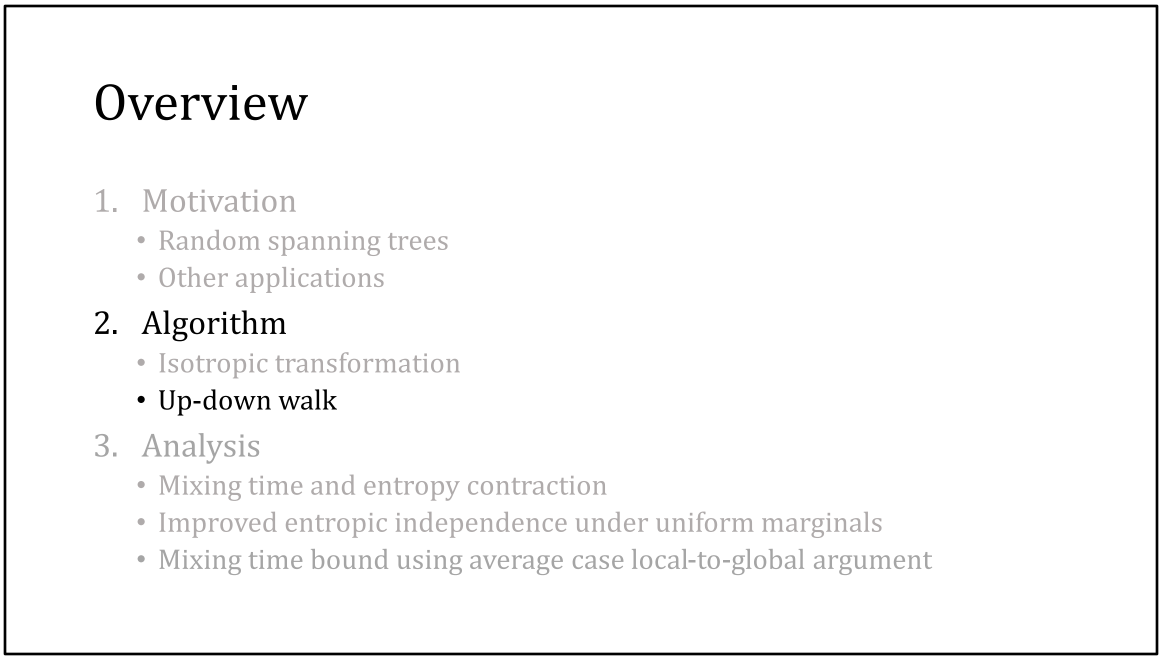
Properties of transformation

1. $p_{e'} = Pr_{\mu'}[e' \in T] \approx \frac{k}{|U|}$.
2. $|U| \leq 2n$
3. μ' is strongly Rayleigh if μ is


$\mu: \binom{[n]}{k} \to \mathbb{R}_{\geq 0}$
$\mu': \binom{U}{k} \to \mathbb{R}_{\geq 0}$

This transformation will turn the distribution mu into mu' that is a weighted spanning tree dist, where the probability of sampling a tree is proportional to the product of the weights of the edges in the tree. The first nice properties is that marginals of edges in mu' are all equal
Secondly, the number of new edges is only blow up by a factor of 2.
Thirdly, our alg to sample spanning tree from the prev paper can also handle weighted spanning tree distribution too, so now we only need to worry about sampling from mu'. More generally, mu' will be strongly Rayleigh as long as mu is, so this transformation works for any strongly Rayleigh distribution.

As an aside, I want to mention that this isotropic transformation has been used in previous papers by Nima and Michal derezinski + another followup that involves Nima, Michal, Elizabeth Yang and myself. This paper improves the result in those papers.
Similar method has also been used in graph sparsification algo and alg to count number of spanning tree by Kyng, Durfee, Rao and others.



Overview

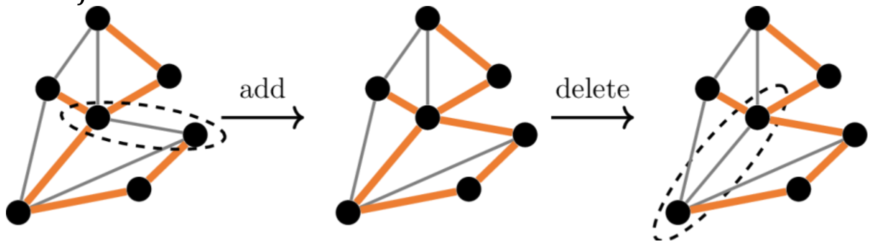
1. Motivation
   - Random spanning trees
   - Other applications
2. Algorithm
   - Isotropic transformation
   - Up-down walk
3. Analysis
   - Mixing time and entropy contraction
   - Improved entropic independence under uniform marginals
   - Mixing time bound using average case local-to-global argument

So now we need to sample from this weighted spanning tree distribution mu'. I will describe the alg from our paper in STOC last year.

Up-down walk

Repeat for sufficiently many times. Take tree T

1. Add an edge e
2. Remove an edge f from the unique cycle in $T + e$. Update $T = T + e - f$



It’s a Markov chain alg that we named the updown walk. We start with a tree T, we add an edge e, then we remove an edge f from the fundamental cycle of e wrt T, i.e. the unique cycle in T union with e. We update T to be T+e –f then repeat the step. The probability of adding e and removing f is according to the weight of the edges in the graph, but I wont go into details there. If the weights are all the same, then the edge e is uniformly random edge from the complement of T, and f is a uniformly random edge from the fundamental cycle

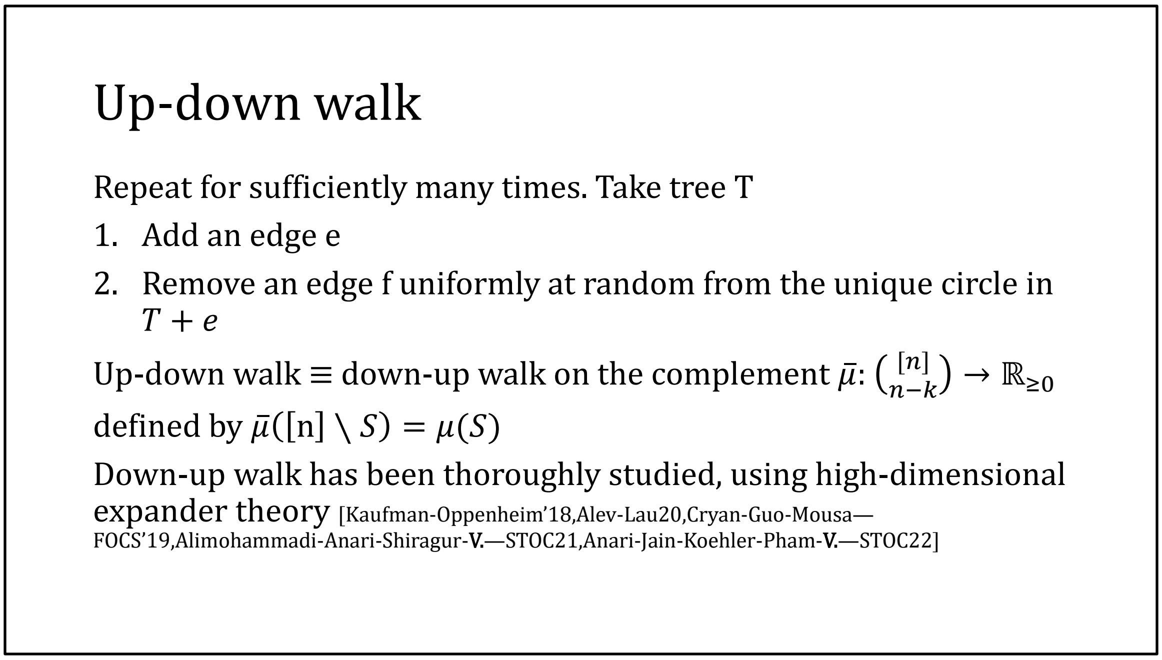
Up-down walk

Repeat for sufficiently many times. Take tree T

1. Add an edge e
2. Remove an edge f from the unique cycle in $T + e$

Up-down walk ≡ down-up walk on the complement $\bar{\mu}: \binom{[n]}{n-k} \to \mathbb{R}_{\geq 0}$ defined by $\bar{\mu}([\text{n}] \setminus S) = \mu(S)$

We can view this up-down walk as the down-up walk on the complement distribution bar{mu} which is the distribution over complement of spanning trees defined by mu.



Up-down walk

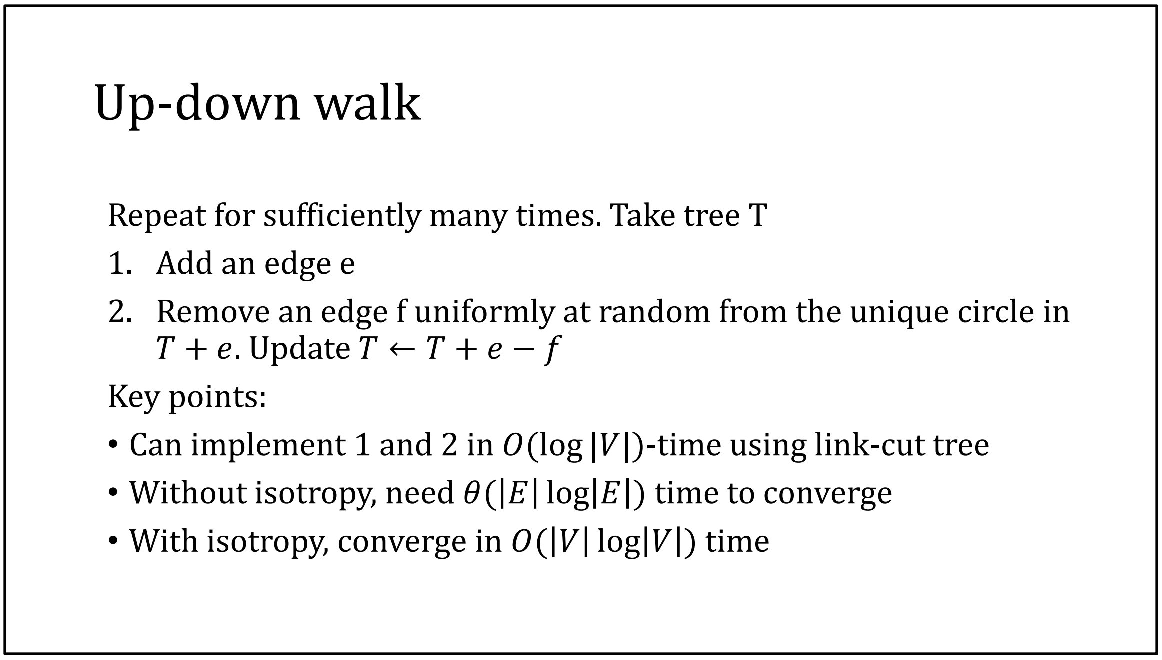
Repeat for sufficiently many times. Take tree T

1. Add an edge e
2. Remove an edge f uniformly at random from the unique circle in $T+e$

Up-down walk ≡ down-up walk on the complement $\bar{\mu}: \binom{[n]}{n-k} \to \mathbb{R}_{\geq 0}$ defined by $\bar{\mu}([\mathrm{n}] \setminus S) = \mu(S)$

Down-up walk has been thoroughly studied, using high-dimensional expander theory [Kaufman-Oppenheim’18,Alev-Lau20,Cryan-Guo-Mousa—FOCS’19,Alimohammadi-Anari-Shiragur-**V.**—STOC21,Anari-Jain-Koehler-Pham-**V.**—STOC22]

And down-up walk has been thoroughly studied, using theory of high-dimensional expander. So that’s why it makes sense to cast this walk as a down-up walk



Up-down walk

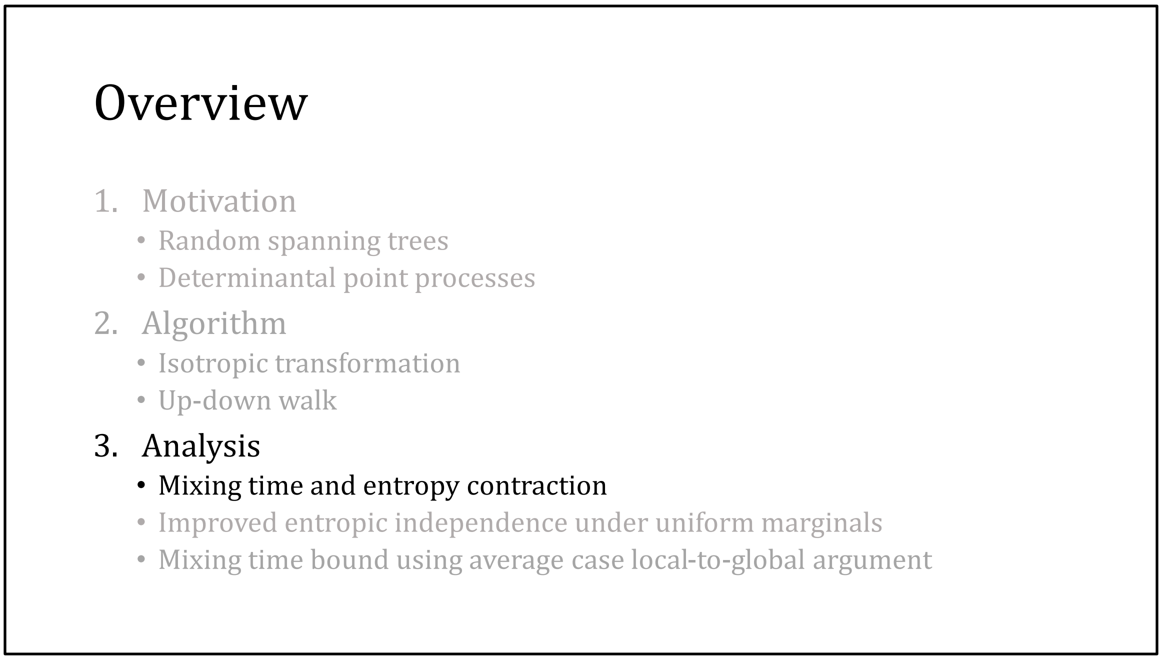
Repeat for sufficiently many times. Take tree T

1. Add an edge e
2. Remove an edge f uniformly at random from the unique circle in $T + e$. Update $T \leftarrow T + e - f$

Key points:

- Can implement 1 and 2 in $O(\log |V|)$-time using link-cut tree
- Without isotropy, need $\theta(|E| \log |E|)$ time to converge
- With isotropy, converge in $O(|V| \log |V|)$ time

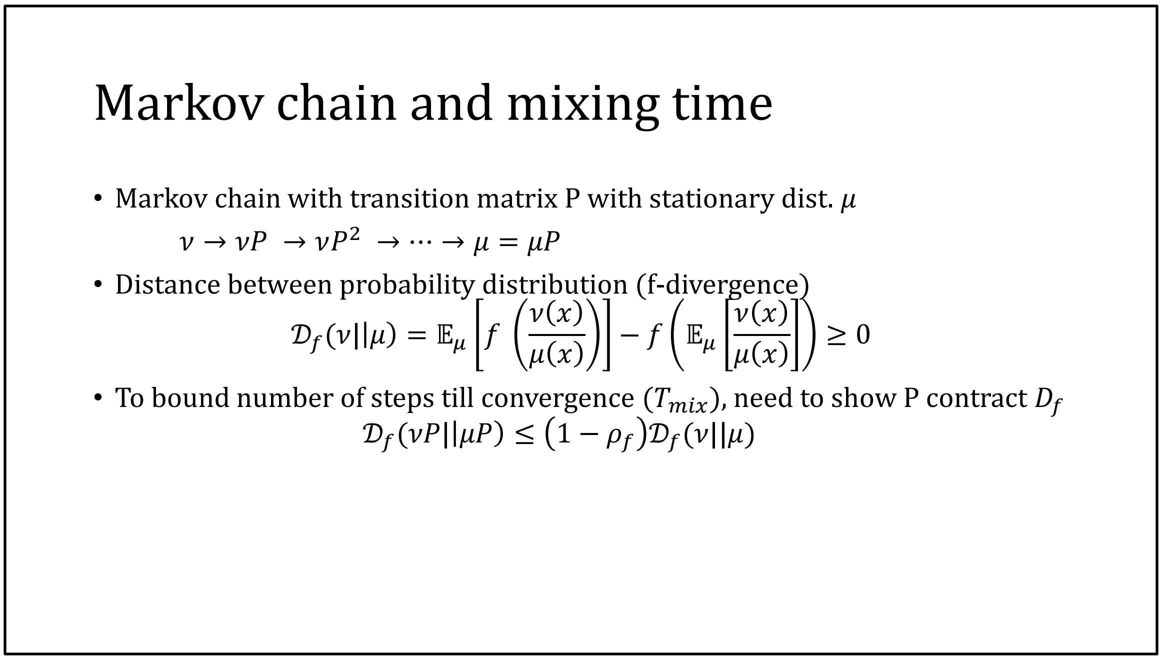
We summary the key points of this random walk algo.
Each step can be implemented in log time using the link-cut tree data structure
Without isotropy, meaning the assumption that all edges have equal prob. Of being included in a spanning tree, this algo takes linear time in the number of edges to converge to the target weighted spanning tree dist.
But, with the isotropy assumption, this walk will converge in quasilinear time in the number of vertices.



Overview

1. Motivation
   - Random spanning trees
   - Determinantal point processes
2. Algorithm
   - Isotropic transformation
   - Up-down walk
3. Analysis
   - Mixing time and entropy contraction
   - Improved entropic independence under uniform marginals
   - Mixing time bound using average case local-to-global argument

Now we move on to the analysis. First I will introduce some tool to bound the mixing time, which is the time a random walk take to converge to the stationary distribution. For this part lets assume the weight are all equal, so you are sampling from the uniform random spanning tree distribution, for simplicity. But there's way to make the analysis works for weighted spanning tree distribution.



Markov chain and mixing time

- Markov chain with transition matrix P with stationary dist. μ

$$\nu \to \nu P \to \nu P^2 \to \cdots \to \mu = \mu P$$

- Distance between probability distribution (f-divergence)

$$\mathcal{D}_f(\nu || \mu) = \mathbb{E}_\mu \left[f\left(\frac{\nu(x)}{\mu(x)}\right)\right] - f\left(\mathbb{E}_\mu\left[\frac{\nu(x)}{\mu(x)}\right]\right) \geq 0$$

- To bound number of steps till convergence (T_{mix}), need to show P contract D_f

$$\mathcal{D}_f(\nu P || \mu P) \leq (1 - \rho_f)\mathcal{D}_f(\nu || \mu)$$

For a Markov chain with trans matrix P, we say P has stationary distribution mu iff after applying P multiple time to a starting distribution nu, we obtain mu. Note that this means mu is equial to mu times P , or mu is the left eigenvalue of P. So recall that we think about probability distribution as vectors, then we can define a distance function on this vector space.
For a convex function f over the real domain, let the f-divergence of two distribution nu and mu be the difference between expectation of f of nu divides by mu and f of ecpectation of nu divides by mu, where the expectation is taken over mu. Note that this divergence is always nonnegtive by convexity of f. If nu and mu is equal then nu over mu is always 1, and the divergence is 0.

In order to bound the number of steps till divergence, we can prove that P contract the f-divergence, meaning that after applying P to both nu and mu the f divergence decreases by a factor of 1-rho

Entropy contraction vs. mixing time

f -divergence contraction for $f(t) = t\ log\ t$

- $\mathcal{D}_{x \log x} = \mathcal{D}_{KL}; \rho_{KL} \equiv$ modified log-sobolev constant (MLSI)
- $T_{mix} \leq \rho_{KL}^{-1} \log\log(1/\min \mu(x))$

For f equal x log x, the divergence is the entropy of nu over mu, so we call rho of x log x the entropy contraction contraction factor. Here I write KL bc the divergence is also the KL divergence between nu and mu.
Mixing time is upper bounded by the inverse of entropy contraction factor time log log of min mu(x) inverse

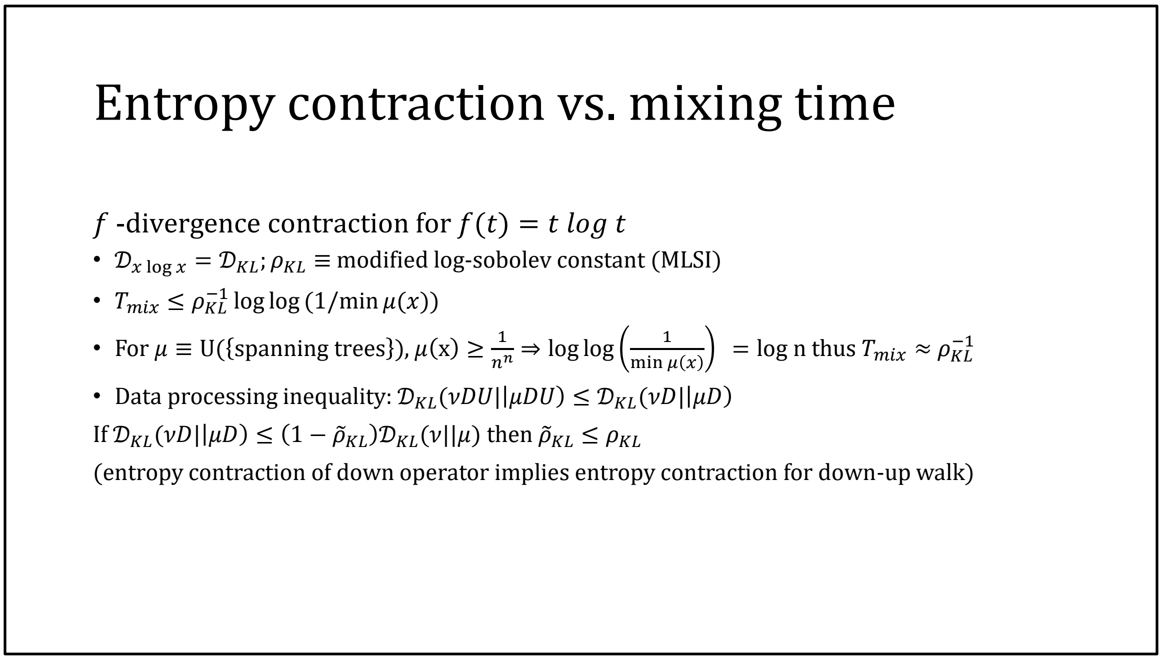
Entropy contraction vs. mixing time

f -divergence contraction for $f(t) = t \, log \, t$

- $\mathcal{D}_{x \log x} = \mathcal{D}_{KL}; \rho_{KL} \equiv$ modified log-sobolev constant (MLSI)
- $T_{mix} \leq \rho_{KL}^{-1} \log\log(1/\min \mu(x))$
- For $\mu \equiv$ U({spanning trees}), $\mu(\text{x}) \geq \frac{1}{n^n} \Rightarrow \log\log\left(\frac{1}{\min \mu(x)}\right) = \log$ n

thus $T_{mix} \approx \rho_{KL}^{-1}$

For the uniform spanning tree dist, since the number of spanning tree is atmost n^n, so min mu(x) is at least 1/n^n. so log log of inverse min of mu is around log n, thus the mixing time is roughly the inverse of the MLSI constant.



Entropy contraction vs. mixing time

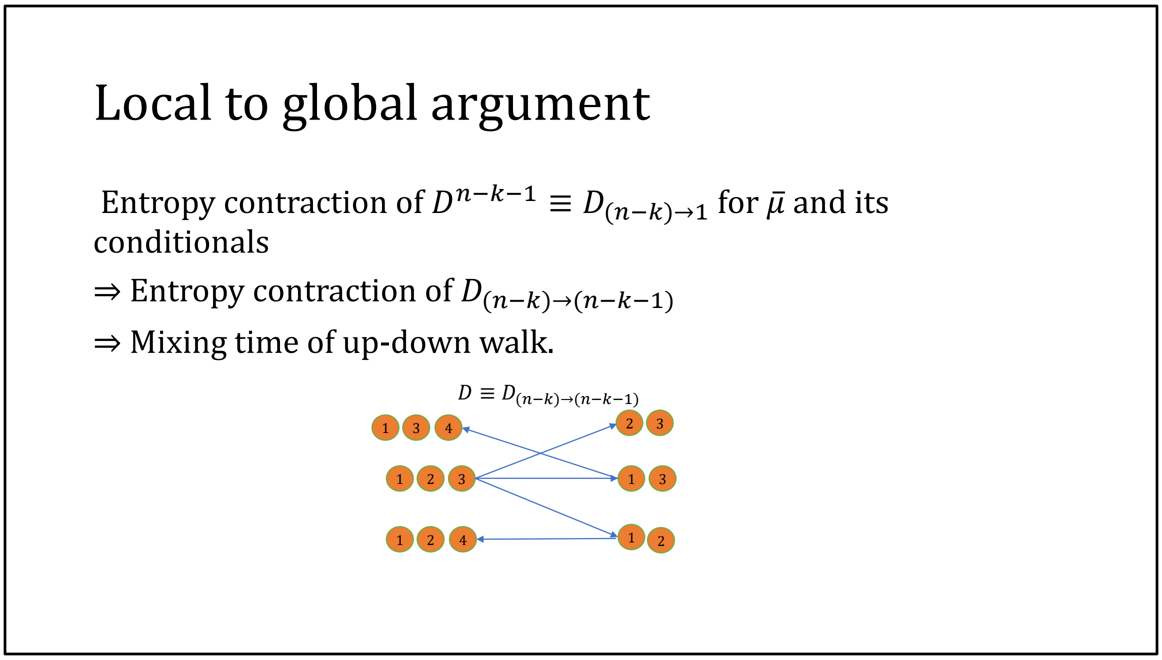
f -divergence contraction for $f(t) = t\ log\ t$

- $\mathcal{D}_{x \log x} = \mathcal{D}_{KL}$; $\rho_{KL} \equiv$ modified log-sobolev constant (MLSI)
- $T_{mix} \leq \rho_{KL}^{-1} \log\log(1/\min \mu(x))$
- For $\mu \equiv$ U({spanning trees}), $\mu(\mathrm{x}) \geq \frac{1}{n^n} \Rightarrow \log\log\left(\frac{1}{\min \mu(x)}\right) = \log \mathrm{n}$ thus $T_{mix} \approx \rho_{KL}^{-1}$
- Data processing inequality: $\mathcal{D}_{KL}(\nu DU || \mu DU) \leq \mathcal{D}_{KL}(\nu D || \mu D)$

If $\mathcal{D}_{KL}(\nu D || \mu D) \leq (1 - \tilde{\rho}_{KL})\mathcal{D}_{KL}(\nu || \mu)$ then $\tilde{\rho}_{KL} \leq \rho_{KL}$

(entropy contraction of down operator implies entropy contraction for down-up walk)

By data processing inequality, when the Markov chain is the down up walk, you can bound the entropy contraction factor of P by the entropy contraction factor of just the down operator D (this is the step when you add a uniformly random edge to the spanning tree).

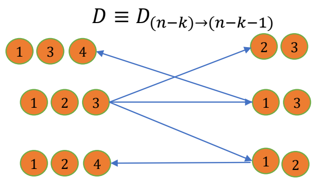


Local to global argument

Entropy contraction of $D^{n-k-1} \equiv D_{(n-k)\to 1}$ for $\bar{\mu}$ and its conditionals

$\Rightarrow$ Entropy contraction of $D_{(n-k)\to(n-k-1)}$

$\Rightarrow$ Mixing time of up-down walk.



Recall that the 1-step down operator is the operator that takes a set of size h and remove 1 element uniformly at random from that set. Similarly, we can define the ell step down operator as the 1-step down operator applies ell times. So here ell is n-k, because we are applying the down operator to the complement of the spanning tree, and since each tree has k edges, and there are n edges in total, the complement has n-k edges.
We can understand the entropy contraction of the 1-step down-operator, which control the mixing time of the random walk that we care about, by understanding the entropy contraction of this multi-step operator D from (n-k) to 1

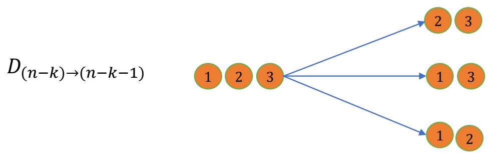
Local to global argument

Entropy contraction of $D_{(n-k)\to 1}$ for $\bar{\mu}$ and its conditionals

$\Rightarrow$ Entropy contraction of $D_{(n-k)\to(n-k-1)}$ $\equiv$ lower bound of $\tilde{\rho}_{KL}$

$\Rightarrow$ Mixing time of up-down walk.

Conditional of ν at S: $\nu_S(T \setminus S) = \nu(T|S) = \nu\,(T)$ for T $\supseteq S$



A technical details is that we need to also look at the conditional of \bar{mu}. For a distr nu, for a set S of elements, the conditional of mu at S is the distribution induced on subset of the support of nu that contain S.

Entropic independence [Anari-Koehler-Jain-Pham-V.-STOC22]

$D_{(n-k)\to 1}(S)$: sample $i \in S$ uniformly

$\bar{\mu}$ *is* $\frac{1}{\alpha}$-entropic independence ⇔ ∀ν:

$$\mathcal{D}_{KL}(\bar{\nu}||\bar{\mu}) \geq \alpha(n-k)\mathcal{D}_{KL}(\bar{\nu}D_{(n-k)\to 1}||\bar{\mu}D_{(n-k)\to 1})$$

Entropic independence [Anari-Koehler-Jain-Pham-V.-STOC22]

$D_{(n-k)\to 1}(S)$: sample $i \in S$ uniformly

$\bar{\mu}\ is\ \frac{1}{\alpha}$-entropic independence ⇔ ∀ν:

$$\mathcal{D}_{KL}(\bar{\nu}||\bar{\mu}) \geq \alpha(n-k)\mathcal{D}_{KL}(\bar{\nu}D_{(n-k)\to 1}||\bar{\mu}D_{(n-k)\to 1})$$

Strongly Rayleigh ⇒ 1-entropic independence

$$\mathcal{D}_{KL}(\bar{\nu}||\bar{\mu}) \geq (n-k)\mathcal{D}_{KL}(\bar{\nu}D_{(n-k)\to 1}||\bar{\mu}D_{(n-k)\to 1})$$

For strongly Rayleigh distribution mu, the complement bar{mu} is also strongly Rayleigh, and strongly Rayleigh implies 1-ent indp so alpha =1, and you got this entropy contraction ineq of the D (n-k)->1 operator

Improved entropic independence under uniform marginals

$\mu: \binom{[n]}{k} \to \mathbb{R}_{\geq 0}$ strongly Rayleigh. When $p_e \leq \tilde{O}\left(\frac{k}{n}\right) \forall e \in [n]$

$$\mathcal{D}_{KL}(\bar{\nu} || \bar{\mu}) \geq (n-k)\log(n/k)\,\mathcal{D}_{KL}\left(\nu D_{(n-k)\to 1} \middle|\middle| \mu D_{(n-k)\to 1}\right)$$

Add picture

Improved entropic independence under uniform marginals

$\mu: \binom{[n]}{k} \to \mathbb{R}_{\geq 0}$ strongly Rayleigh. When $p_e \leq \tilde{O}\left(\frac{k}{n}\right) \forall e \in [n]$

$$\mathcal{D}_{KL}(\bar{\nu}||\bar{\mu}) \geq (n-k)\log\left(\frac{n}{k}\right)\mathcal{D}_{KL}(\bar{\nu}D_{(n-k)\to 1}||\bar{\mu}D_{(n-k)\to 1})$$

1. $\mathcal{D}_{KL}(\bar{\nu}||\bar{\mu}) = \mathcal{D}_{KL}(\nu||\mu) \geq k\mathcal{D}_{KL}(\nu D_{k\to 1}||\mu D_{k\to 1})$

When the marginal are uniform, we get improved marginals, so instead of (n-k) in the prev slides, we get a factor (n-k) log(n/k). And this extra log (n/k) factor saves us.

Improved entropic independence under uniform marginals

$\mu: \binom{[n]}{k} \to \mathbb{R}_{\geq 0}$ strongly Rayleigh. When $p_e \leq \tilde{O}\left(\frac{k}{n}\right) \forall e \in [n]$

$$\mathcal{D}_{KL}(\bar{\nu}||\bar{\mu}) \geq (n-k)\log\left(\frac{n}{k}\right)\mathcal{D}_{KL}(\bar{\nu}D_{(n-k)\to 1}||\bar{\mu}D_{(n-k)\to 1})$$

1. $\mathcal{D}_{KL}(\bar{\nu}||\bar{\mu}) = \mathcal{D}_{KL}(\nu||\mu) \geq k\mathcal{D}_{KL}(\nu D_{k\to 1}||\mu D_{k\to 1})$
2. $k\mathcal{D}_{KL}(\nu D_{k\to 1}||\mu D_{k\to 1}) \geq (n-k)\log\left(\frac{n}{k}\right)\mathcal{D}_{KL}(\bar{\nu}D_{(n-k)\to 1}||\bar{\mu}D_{(n-k)\to 1})$

Here we use the uniform marginal assumption.

Some ideas about proof for this improved entropic independence. First, we can use entropic indepdence of mu to relate kl divergence of nu vs mu to the kl divergence of the marginal of nu vs of mu. Next we use the uniform marginal assumption to compare the KL div of marginal of nu vs mu with the KL div of marginal of nu complement vs mu complement. Note that here we are basically assume mu D_{k to 1} is the uniform distr over the set 1 to n, and now its just an elementary inequality.

Improved entropy contraction ⇒ improved mixing time

Entropy contraction of $D_{(n-k)\to 1}$ for $\bar{\mu}$ and its conditionals

⇒ Entropy contraction of $D_{(n-k)\to(n-k-1)}$

⇒ Mixing time of up-down walk.

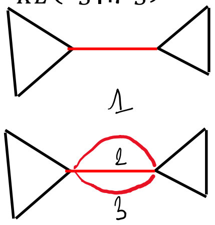
$(n-k)$ contraction ⇒ $n \log n$ mixing time ☹

$(n-k)\log(\frac{n}{k})$ contraction ⇒ k $\log n$ mixing time ☺

So this improved entropy contraction will lead to order k log n mixing time of the random walk.

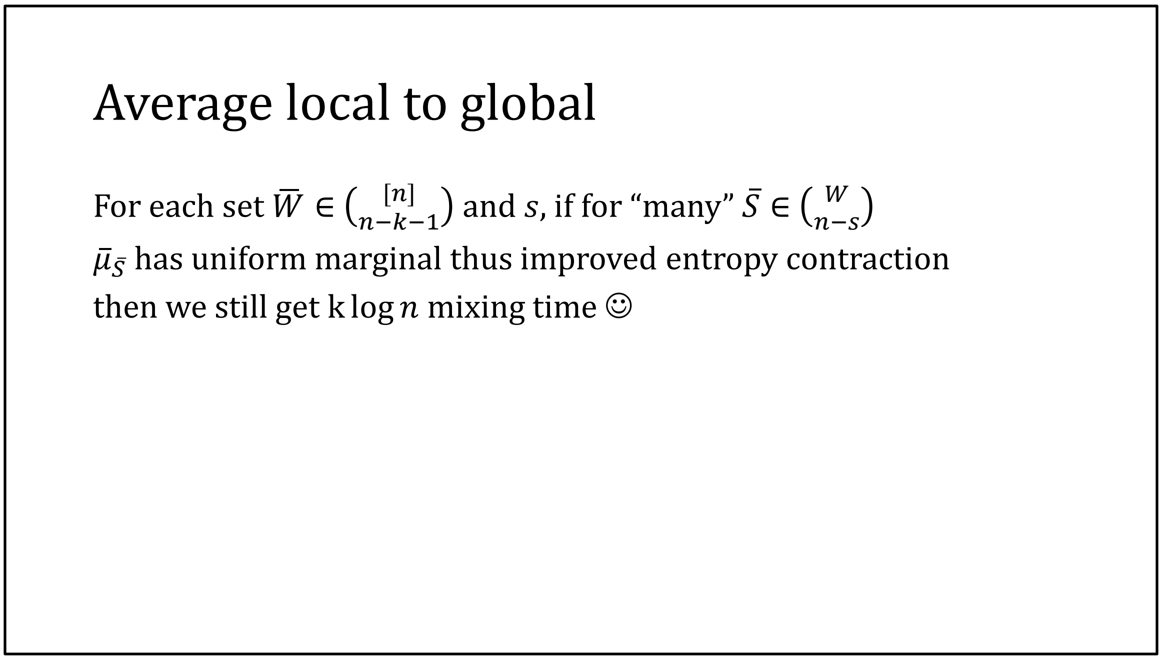
But, not all conditionals of $\bar{\mu}$ has improved entropy contraction ☹
i.e. exists $\bar{S}$ s.t.

$$\mathcal{D}_{KL}(\bar{\nu}_{\bar{S}} || \bar{\mu}_{\bar{S}}) < (n-k)\log\binom{n}{k} \mathcal{D}_{KL}(\bar{\nu}_{\bar{S}} D_{(n-k)\to 1} || \bar{\mu}_{\bar{S}} D_{(n-k)\to 1})$$



$\bar{S} = \{\bar{1}, \bar{3}\}$

We are nearly done here, but still one last hurdle left. That is, not all conditionals of mu complement has improved entropy contraction, and we need improved ent. For example, lets get back the graph with the bridge again. Now we already subdivide the bridge edge into 3 parallel edges, but now if you condition on the edge 1 and 3 being in the complement, meaning that 1 and 3 is not contain in the spanning tree, then the edge 2 become a bridge edge once again, so it has marginal 1, which is much bigger than the marginals of the remaining edges.



Average local to global

For each set $\overline{W} \in \binom{[n]}{n-k-1}$ and s, if for "many" $\bar{S} \in \binom{W}{n-s}$
$\bar{\mu}_{\bar{S}}$ has uniform marginal thus improved entropy contraction
then we still get k $\log n$ mixing time ☺

Okay, so there's an obstacle, so the traditional local to global argument doesn't work. Fortunately, we can get past his hurdle by consider an average version of local to global. So instead of requiring that all conditionals of mu complement are good, meaning that it has uniform marginals and thus improved ent. Contraction, we only require "many" conditionals to have uniform marginals. And that's still enough to get the faster mixing res.

Average local to global

For each set $\bar{W} \in \binom{[n]}{n-k-1}$ and s, if for "many" $\bar{S} \in \binom{W}{n-s}$
$\bar{\mu}_{\bar{S}}$ has uniform marginal thus improved entropy contraction
then we still get $\mathrm{k} \log n$ mixing time ☺
"many" $=$ w/ prob. $1 - 1/n^{10}$ over uniformly chosen $\bar{S}$

So here, by many, I mean mu complement condition on a random set S will have uniform marginal with prob at least 1 – inverse polynomial probability over the choice of S.

Average local to global

For each set $\overline{W} \in \binom{[n]}{n-k-1}$ and s, if for "many" $\bar{S} \in \binom{W}{n-s}$

$\bar{\mu}_{\bar{S}}$ has uniform marginal thus improved entropy contraction

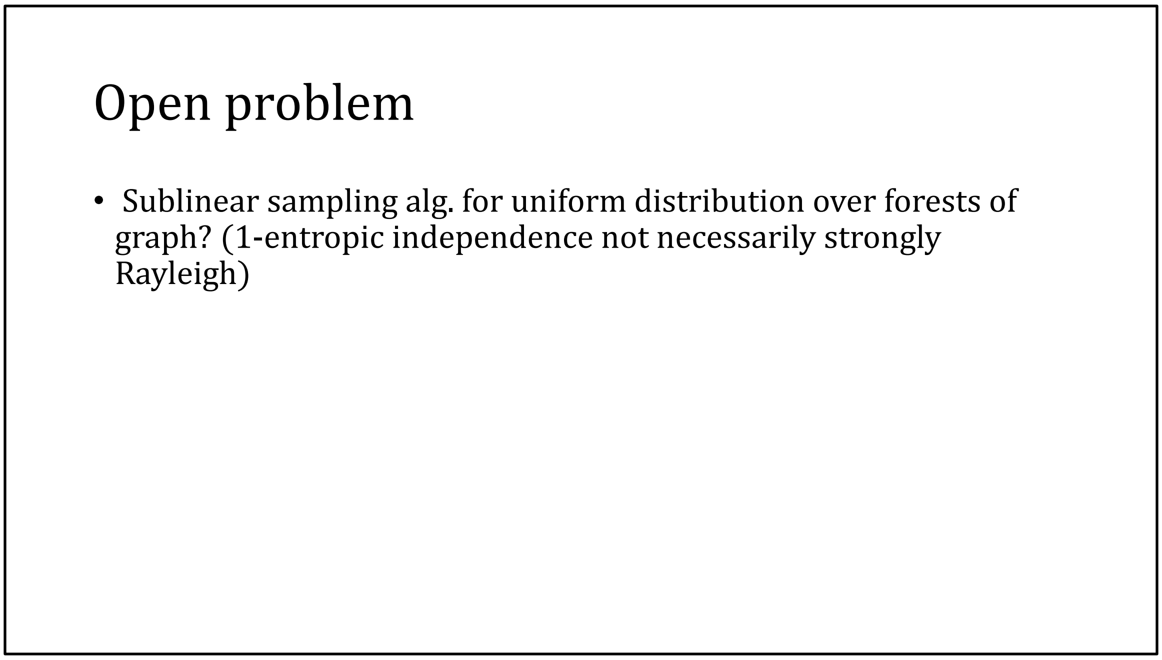
Proof sketch:

Compare marginals of $\bar{\mu}_{\bar{S}}$ and $\bar{\mu}_{\overline{S \cup \{s'\}}}$ for random s'

Since μ is strongly Rayleigh, marginal doesn't change much

Use martingale argument and Bernstein ineq.

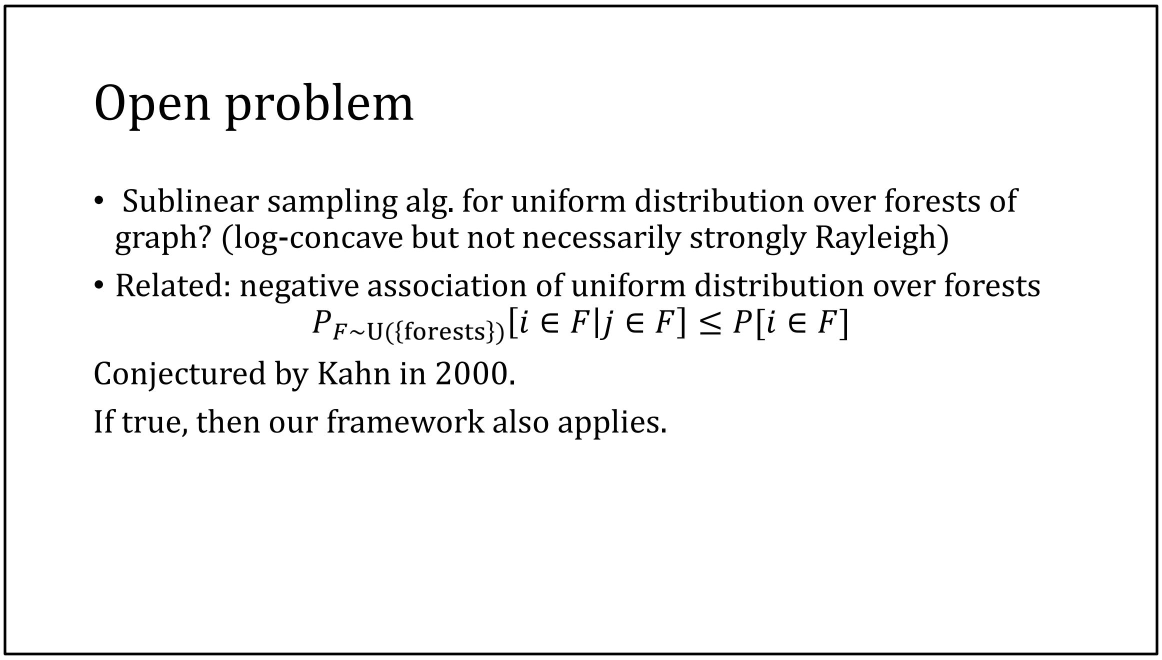
The proof is quite technical, using a martingale argument. The rough idea is, if you start from the empty set so bar{S} is empty, then you have uniform marginals by assumption on mu complement. Now if you add a random element to S complement, you are removing a random edge from the graph, and you are gonna arguing that the marginals of the edges doesn't change much after this step. And here we crucially use properties of strongly Rayleigh.
There's an easier argument that works only for spanning tree and DPP which we didn't include in the paper, but this argument works for all strongly Rayleigh.



Open problem

- Sublinear sampling alg. for uniform distribution over forests of graph? (1-entropic independence not necessarily strongly Rayleigh)

Here are the open problem. Can we go beyond strongly Rayleigh dist? For e.g. can we have similar sublinear sampling alg after preprocessing for uniform distribution over forests of graph. This dist is still 1-ent ind but not always strongly Rayleigh



Open problem

- Sublinear sampling alg. for uniform distribution over forests of graph? (log-concave but not necessarily strongly Rayleigh)
- Related: negative association of uniform distribution over forests

$$P_{F\sim \mathrm{U}(\{\mathrm{forests}\})}[i \in F | j \in F] \leq P[i \in F]$$

Conjectured by Kahn in 2000.

If true, then our framework also applies.

Related to this prev point, Jeff Kahn has this conjecture about negative association properties for edges under the random forest dist. Roughly it said, the marginal of i condition on j is less than the marginal of i.
If this conjectured is true, then our framework applies for forest.