Randomized tree metrics

Probability

Gaussian random variables

Gaussian random variables

$$X \sim N(\mu, \sigma^2)$$

mean, variance

Def: the nicest, most pleasant, convenient random variable ever

Fact let $X \sim N(\mu, \sigma^2)$, $a \in \mathbb{R}$.
then $X + a \sim N(\mu + a, \sigma^2)$
$aX \sim N(a\mu, a^2\sigma^2)$

Fact let $X_1 \sim N(\mu_1, \sigma_1^2)$, $X_2 \sim N(\mu_2, \sigma_2^2)$
then $X_1 + X_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$

Fact let $X \sim N(0,1)$, $t < 1/2$
then $E[e^{tX^2}] = \frac{1}{\sqrt{1-2t}}$

Algorithms

Dimensionality reduction

Bag-of-words

literally any sentence is fine

encodes text in high dimensions

Dimensionality Reduction – high level

$O(\log n)$ dimensions suffice
where $n = \#$ points

$$\mathbb{R}^{\text{very large}} \xrightarrow{f(x)} \mathbb{R}^{O(\log n)}$$

One can map n points into $\mathbb{R}^{O(\log n/\varepsilon^2)}$ while preserving all distances up to $(1 \pm \varepsilon)$-APX

Lemma 8.8. *Let $A \sim \mathcal{N}^{k \times d}$ be a random matrix where each coordinate A_{ij} is an independently drawn sample from $\mathcal{N}$. Let $\epsilon > 0$ be sufficiently small. For any vector x,*

$$\mathbf{P}\left[(1-\epsilon)\|x\|^2 \leq \frac{1}{k}\|Ax\|^2 \leq (1+\epsilon)\|x\|^2\right] \geq 1 - 2e^{-k/8}.$$

$$E[\|Ax\|^2] = \sum_{i=1}^{k} E[(Ax)_i^2] = k\|x\|^2$$

$$E[(\Sigma A_{ij}x_j)^2] = \|x\|^2$$

$$A_{ij}x_j \sim N(0, x_j^2)$$

$$\sum_{j=1}^{n} A_{ij}x_j \sim N(0, \|x\|^2)$$

$$(Ax) \sim N^k(0, \|x\|^2)$$

$$Pr[\|Ax\|_2^2 \geq k\|x\|^2 \cdot (1+\epsilon)] \leq e^{-\epsilon^2 k/8}$$

Probability (and techniques)

LP duality

L_1-metric embeddings

Algorithms

Sparsest Cut

Probability (and techniques)

Algorithms

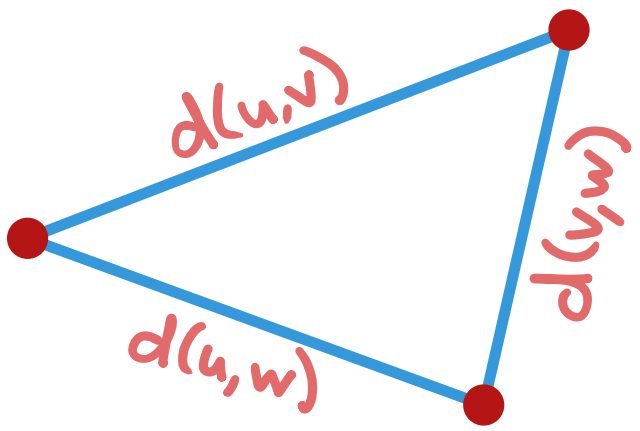
Randomized

Tree metrics

Metrics $d: V \times V \to \mathbb{R}_{\geq 0}$ s.t.

(a) Symmetry:

$$d(u,v) = d(v,u)$$



(b) Positivity:

$d(u,v) \geq 0$,

$d(u,v) = 0 \iff u = v$

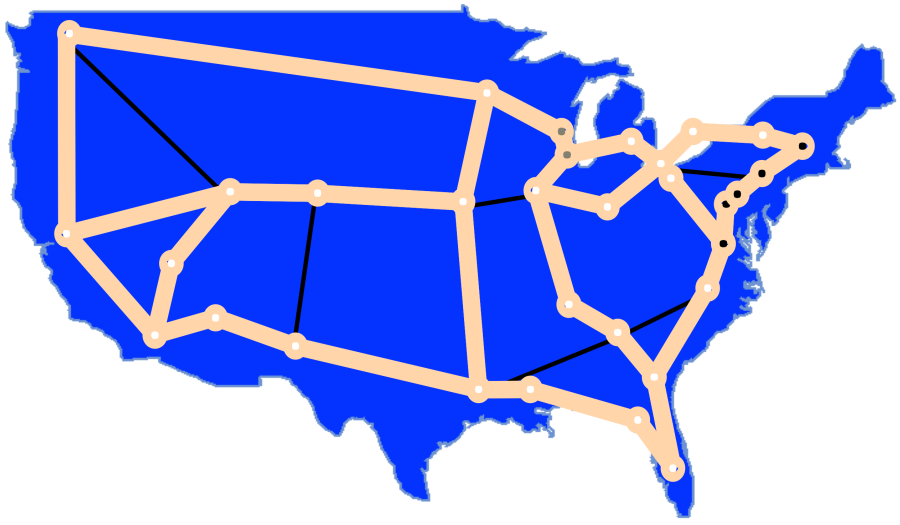
(c) Triangle inequality:

$$d(u,v) \leq d(v,w) + d(w,v) \quad \forall u,v,w$$

e.g. Euclidean metrics, shortest path metrics, cut metrics

Problems about metrics

Buy-at-bulk network design



Input:
- undirected graph $G=(V,E)$
- edge lengths $\ell: E \to \mathbb{R}_{>0}$
- demands $b(s,t) \geq 0 \quad \forall s,t \in V$
- for each edge $e \in E$,

  subadditive cost function $f_e(x) > 0$

  $f_e(x+y) \leq f_e(x) + f_e(y)$

  (per unit length of unit capacity)

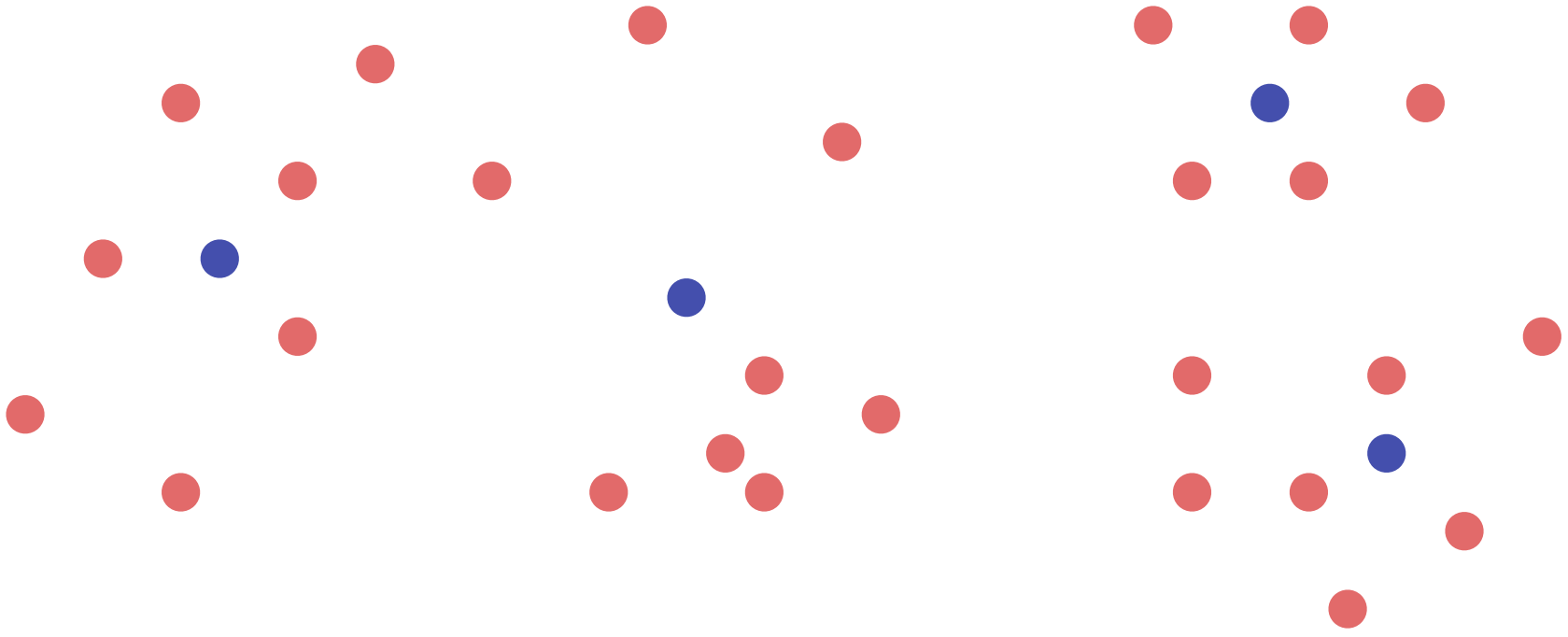
Output: (s,t)-path $P_{st} \quad \forall s,t \in V$

Goal: minimize $\sum_{e \in E} \ell(e) f_e(u_e)$ w/ $u_e = \sum_{s,t:\, e \in P_{st}} b(s,t)$

k-median clustering

Input: n points V, metric $d: V \times V \to \mathbb{R}_{\geq 0}$

Output: k "centers" $c_1, \ldots, c_k$



Goal: minimize $\sum_{v \in V} \min_{i=1,\ldots,k} d(v, c_i)$

Metric labeling

Input: graph $G=(V,E)$, edge weights $w: E \to \mathbb{R}_{>0}$,
k labels $\mathcal{L}$, label costs $c: L \times L \to \mathbb{R}_{>0}$, metric $d: V \times V \to \mathbb{R}_{\geq 0}$

Output: labeling $f: V \to L$

Goal: minimize $\sum_{v \in V} c(v, f(v)) + \sum_{\{u,v\} \in E} w(u,v)\, d(f(u), f(v))$

Other metric-based problems

(sometimes via correspondance between cuts ↔ metrics)

Buy-at-bulk network design

k-median clustering

metric labeling

group Steiner tree

metrical task systems

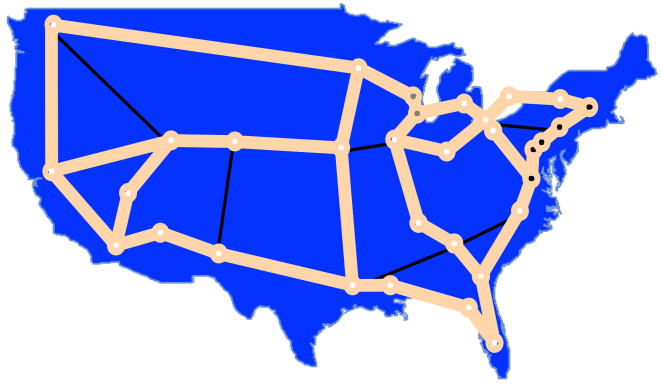
vehicle routing

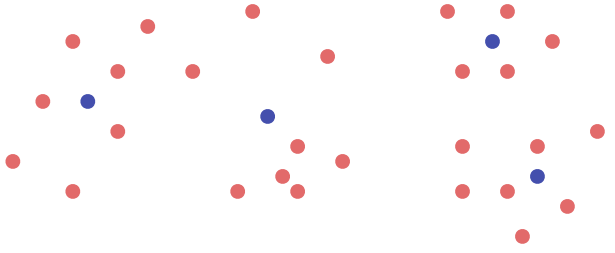
hierarchical placement

distributed k-server

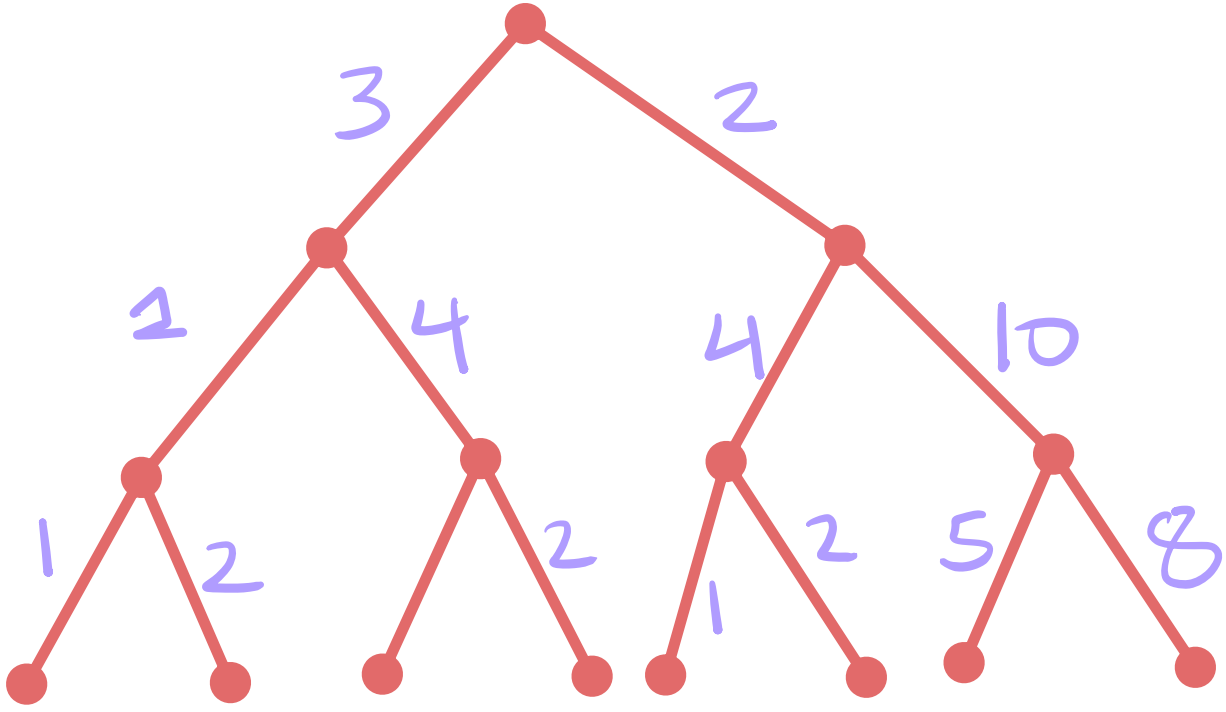
covering Steiner tree

&





Tree metrics: shortest path metric on a tree



(or sum/min over multiple such trees)

Shortest paths

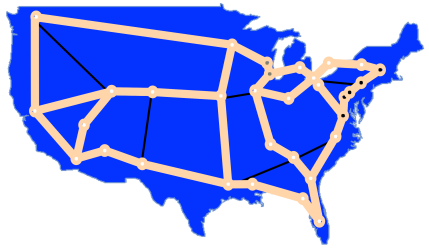
- $\tilde{O}(mn)$ time in general graphs
- super easy on trees!
  - LCA queries
  - path-to-root queries

} log(n) time w/ link-cut trees, etc

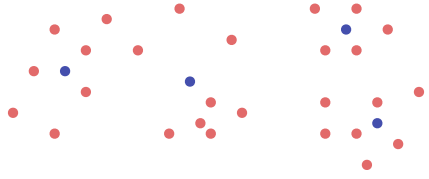
Other metric-based problems

(sometimes via correspondance between cuts ↔ metrics)

Buy-at-bulk network design



k-median clustering



metric labeling

group Steiner tree

metrical task systems

vehicle routing

hierarchical placement

distributed k-server

covering Steiner tree

&

these problems are <u>easy</u> for tree metrics

Approximation by tree metrics

Input: undirected $G=(V,E)$ w/ edge weights $w: E \to \mathbb{R}_{>0}$

(High-level)
Goal: compute tree T (containing V as nodes)
so that shortest paths on T are close to T

$$(\text{stretch } e) = \frac{d_T(u,v)}{d_G(u,v)} \quad \text{for } e=\{u,v\}$$

($d_T(u,v)$: distance in T; $d_G(u,v)$: distance in G)

ideally $(\text{stretch } e) = \text{polylog}(n) \quad \forall e$

Approximation by tree metrics

Input: undirected $G=(V,E)$ w/ edge weights $w: E \to \mathbb{R}_{>0}$

(High-level)
Goal: compute tree T (containing V as nodes)
so that shortest paths on T are close to T

---

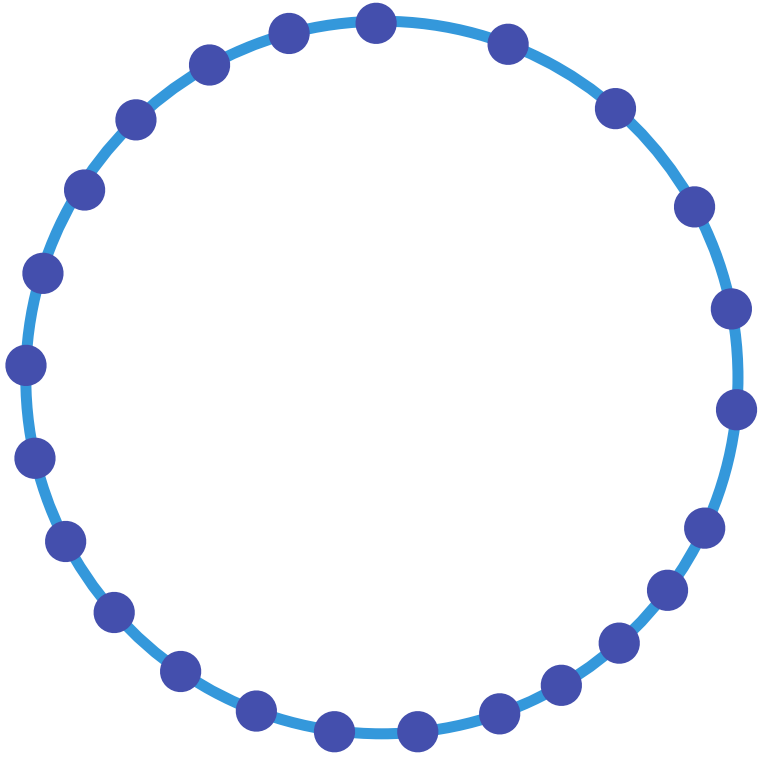
for $e=\{u,v\}$

$$(\text{stretch } e) = \frac{d_T(u,v)}{d_G(u,v)}$$

($d_T(u,v)$: distance in T; $d_G(u,v)$: distance in G)

ideally

$(\text{stretch } e) = \text{polylog}(n) \quad \forall e$



Approximation by tree metrics

Input: undirected $G=(V,E)$ w/ edge weights $w: E \to \mathbb{R}_{>0}$

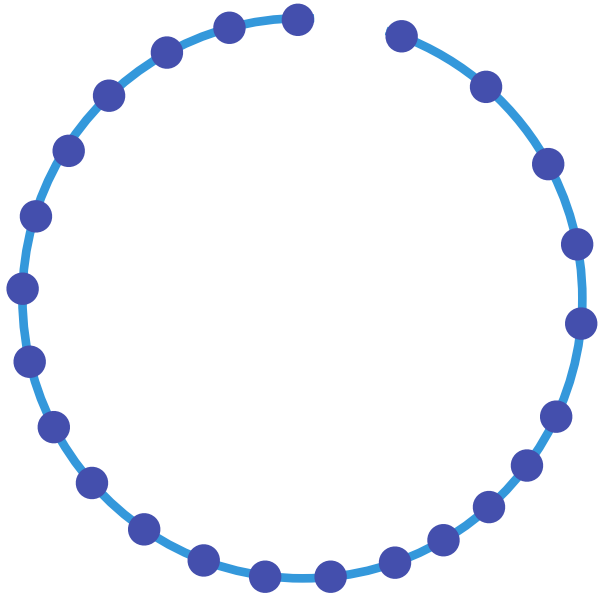
(High-level)
Goal: compute tree T (containing V as nodes)
so that shortest paths on T are close to T

---

① deterministic low-stretch spanning trees [AKPW95]

spanning tree T s.t.

$$\frac{1}{|E|} \sum_{e \in E} (\text{stretch } e) \leq n^{o(1)}$$



Approximation by tree metrics

Input: undirected $G=(V,E)$ w/ edge weights $w: E \to \mathbb{R}_{>0}$

(High-level)
Goal: compute tree T (containing V as nodes)
so that shortest paths on T are close to T

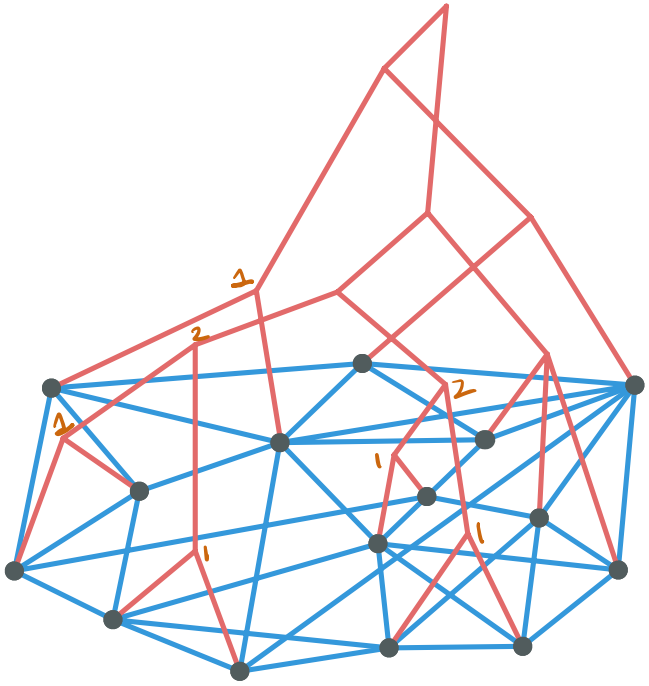
---

① deterministic low-stretch spanning trees [AKPW95]

---

② Randomized dominating tree metric [Bartal 96, 98; FRT04]

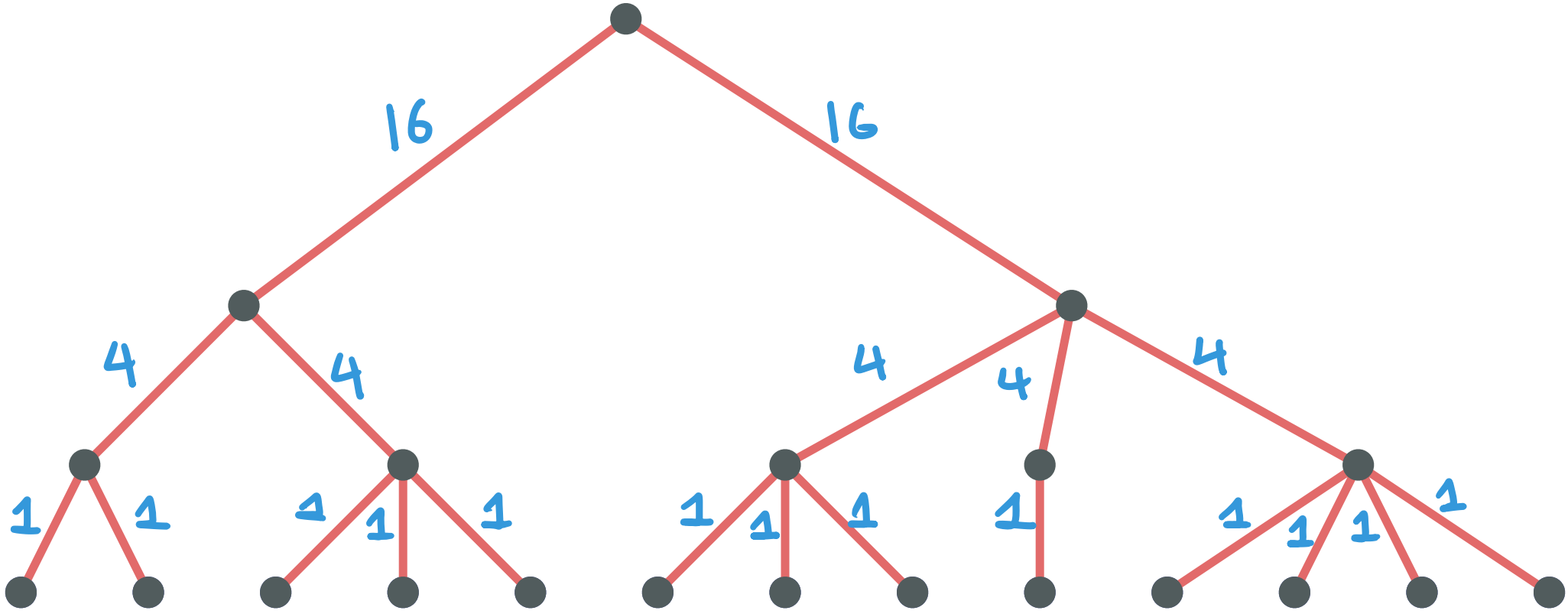
Auxiliary tree T w/ V as leaves s.t.



("dominating")
- $d_T(u,v) \geq d_G(u,v) \quad \forall u,v$

- $E[\text{stretch } e]$ small $\forall e \in E$

Hierarchical Tree Metric



edge from height $i+1$ to i has weight d^i for fixed d. (we will have $d=4$)

On Approximating Arbitrary Metrics by Tree Metrics

Yair Bartal *

Abstract

We improve the result of [Bart96] on probabilistic approximation of metric spaces by "hierarchically well-separated tree" metric spaces. We obtain an approximation factor of $O(\log n \log\log n)$ getting the gap to the lower bound within lower order factors. We also give a deterministic version of the result which gives a tree with low average distortion of distances.

The results have applications in a variety of areas including approximation algorithms, on-line algorithms and distributed computation and hence we obtain new approximation bounds for these applications.

1 Introduction

1.1 Probabilistic Metric Approximations and HSTs

We study the problem of approximating arbitrary metric spaces by distances in more simple metric spaces. This allows reducing the solution of problems to the more simple spaces.

Approximations of metric spaces by more simple metric spaces has been intensively studied in various areas of mathematics and computer science. We mention but a few. Johnson and Lindenstrauss [JL84] and Bourgain [Bour85] study embeddings in Hilbert spaces (from a perspective of functional analysis). Graham and Winkler [GW85] study embeddings in $\mathbb{Z}^d$ (from a graph theoretic motivation). Algorithmic applications in distributed computation and graph algorithms, have led to the notion of graph spanners, introduced by Peleg and Ullman [PU89] and later studied in many papers including [PS89, ADDJS90, CDNS92], and to low-distortion embeddings in low-dimensional real normed spaces by Linial, London and Rabinovich [LLR94].

In [Bart96] we define *probabilistic approximation* of metric spaces by a set of "simpler" metric spaces $\mathcal{S}$. Given a metric space M over a finite set V, we consider sets $\mathcal{S}$ of "simple" metric spaces over V such that distances in M are dominated by the corresponding distances in metric spaces in $\mathcal{S}$. M is said to be *α-probabilistically-approximated* by $\mathcal{S}$, if there exists a probability distribution over $\mathcal{S}$ such that for *every* pair of nodes in V, the expected ratio between the distance in a metric space in $\mathcal{S}$ chosen from the probability distribution, and the distance in M is at most α.

Regarding a randomized algorithm as a distribution over deterministic algorithms, its performance ratio is the expected performance ratio of these algorithms over its own coin tosses. The probabilistic metric approximation approach provides a *novel technique* for bounding the performance ratio of *randomized* algorithms for optimization problems on metric spaces where a solution can be expressed as a linear combination of distances in the metric space. If every metric space is α-probabilistically-approximated by $\mathcal{S}$, and the performance ratio of randomized algorithms for $\mathcal{S}$ is at most β, then the performance ratio of randomized algorithms for any metric space is at most $\alpha\beta$.

The advantage of using probabilistic metric space approximation is that certain classes of simple metric spaces which cannot provide a good deterministic approximation of arbitrary metric spaces may provide a good probabilistic approximation.

A natural candidate for a class of "simple" metric spaces is the class of tree metric spaces. Indeed, intensive recent research has been concerned with approximating metric spaces by tree metrics, including work on *numerical taxonomy* [BG92, ABFNPT96] and *tree spanners* [Cai92, CC95].

However, there exist simple metric spaces for which every specific tree dominating the distances of the metric space has distortion $\Omega(n)$ (trivially matched by the

*Bell-Labs, Lucent Technologies, 600 Mountain Avenue, Murray Hill, NJ 07974. *E-mail:* yair@research.bell-labs.com

Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Journal of Computer and System Sciences 69 (2004) 485–497
JOURNAL OF COMPUTER AND SYSTEM SCIENCES
http://www.elsevier.com/locate/jcss

A tight bound on approximating arbitrary metrics by tree metrics

Jittat Fakcharoenphol,[a,1] Satish Rao,[b,2] and Kunal Talwar[c,*,3]

[a] *Kasetsart University, Bangkok, Thailand*
[b] *Computer Science Division, University of California, Berkeley, CA 94720, USA*
[c] *Computer Science Division, University of California, Berkeley, CA 94720, USA*

Received 3 September 2003; revised 12 April 2004

Available online 24 June 2004

Abstract

In this paper, we show that any n point metric space can be embedded into a distribution over dominating tree metrics such that the expected stretch of any edge is $O(\log n)$. This improves upon the result of Bartal who gave a bound of $O(\log n \log\log n)$. Moreover, our result is existentially tight; there exist metric spaces where any tree embedding must have distortion $\Omega(\log n)$-distortion. This problem lies at the heart of numerous approximation and online algorithms including ones for group Steiner tree, metric labeling, buy-at-bulk network design and metrical task system. Our result improves the performance guarantees for all of these problems.

Keywords: Metrics; Embeddings; Tree Metrics

1. Introduction

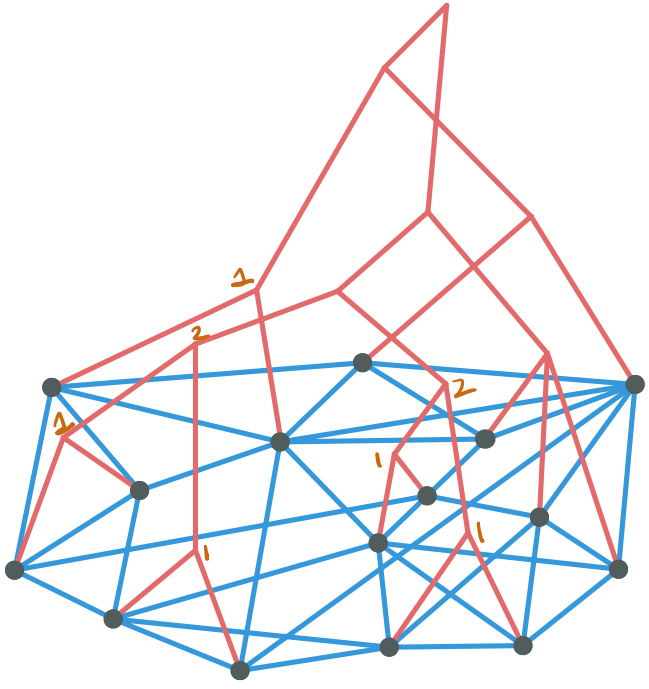
1.1. Metric approximations

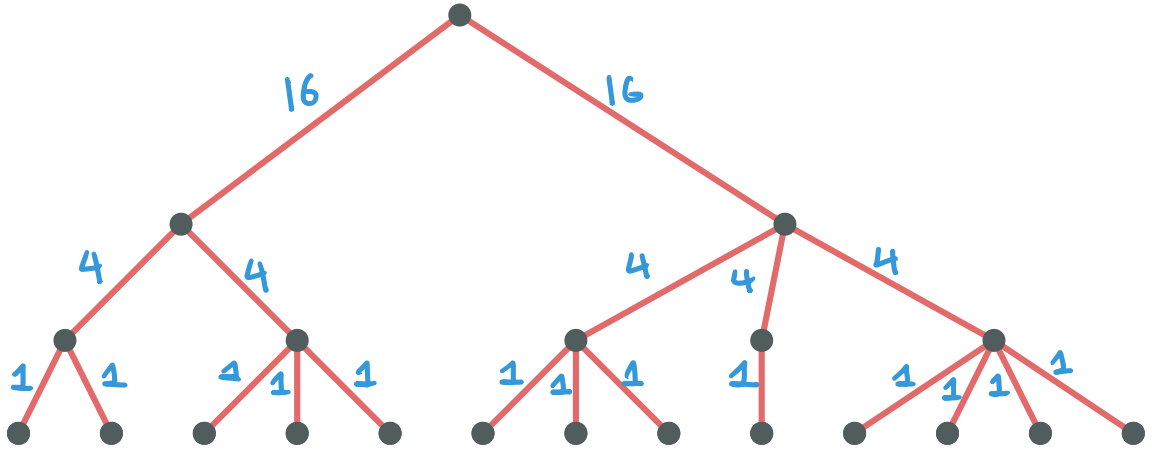
The problem of approximating a given graph metric by a "simpler" metric has been a subject of extensive research, motivated from several different perspectives. A particularly simple metric of

*Corresponding author.
E-mail address: kunal@cs.berkeley.edu (K. Talwar).
[1] Supported in part by a DPST scholarship and NSF Grant CCR-0105533. Work done while the author was at University of California, Berkeley.
[2] Supported by NSF Grant CCR-0105533.
[3] Supported by NSF Grants CCR-0105533 and CCR-9820897.

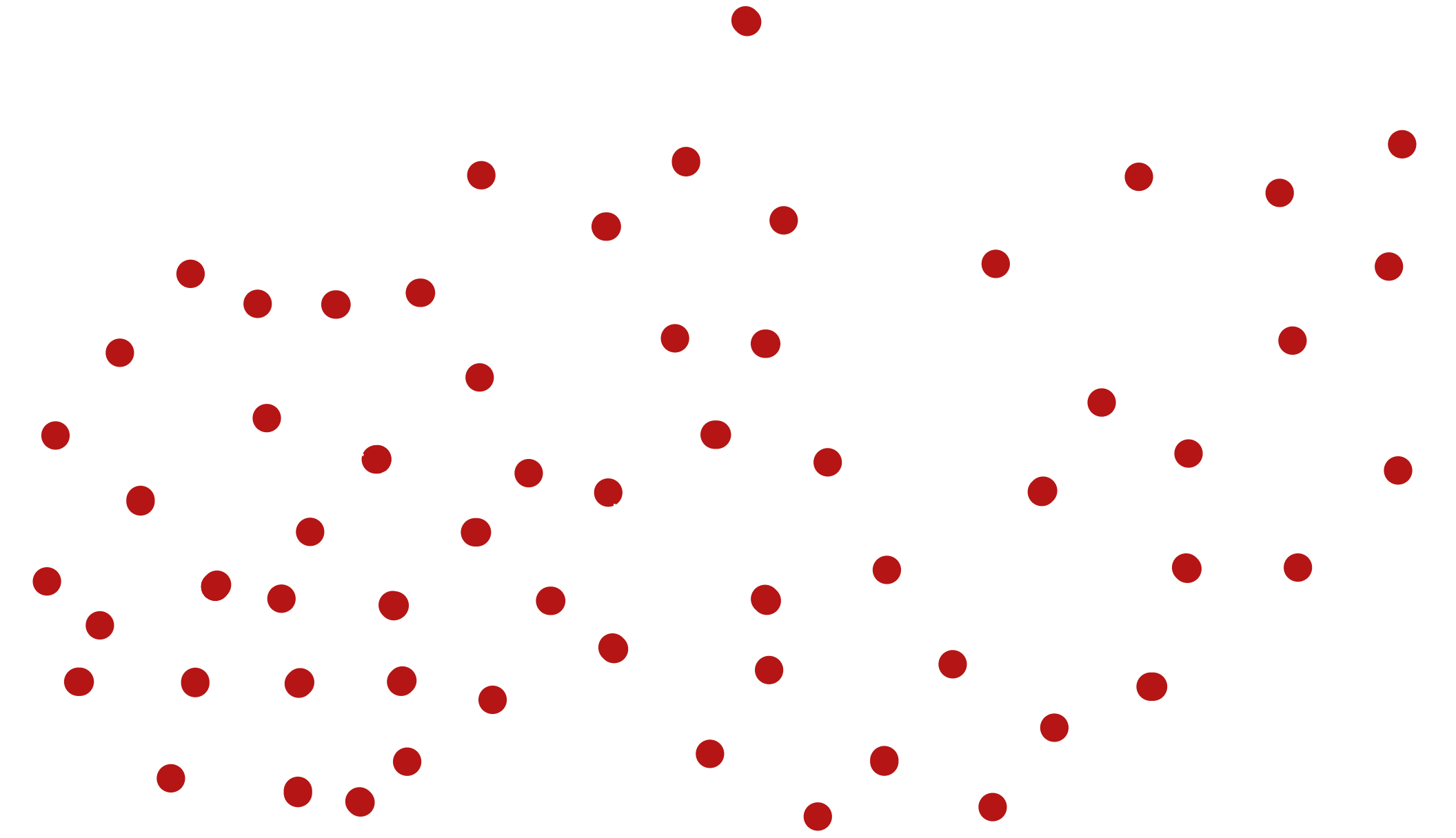
Theorem In randomized polynomial time, randomized hierarchical dominating tree metric w/

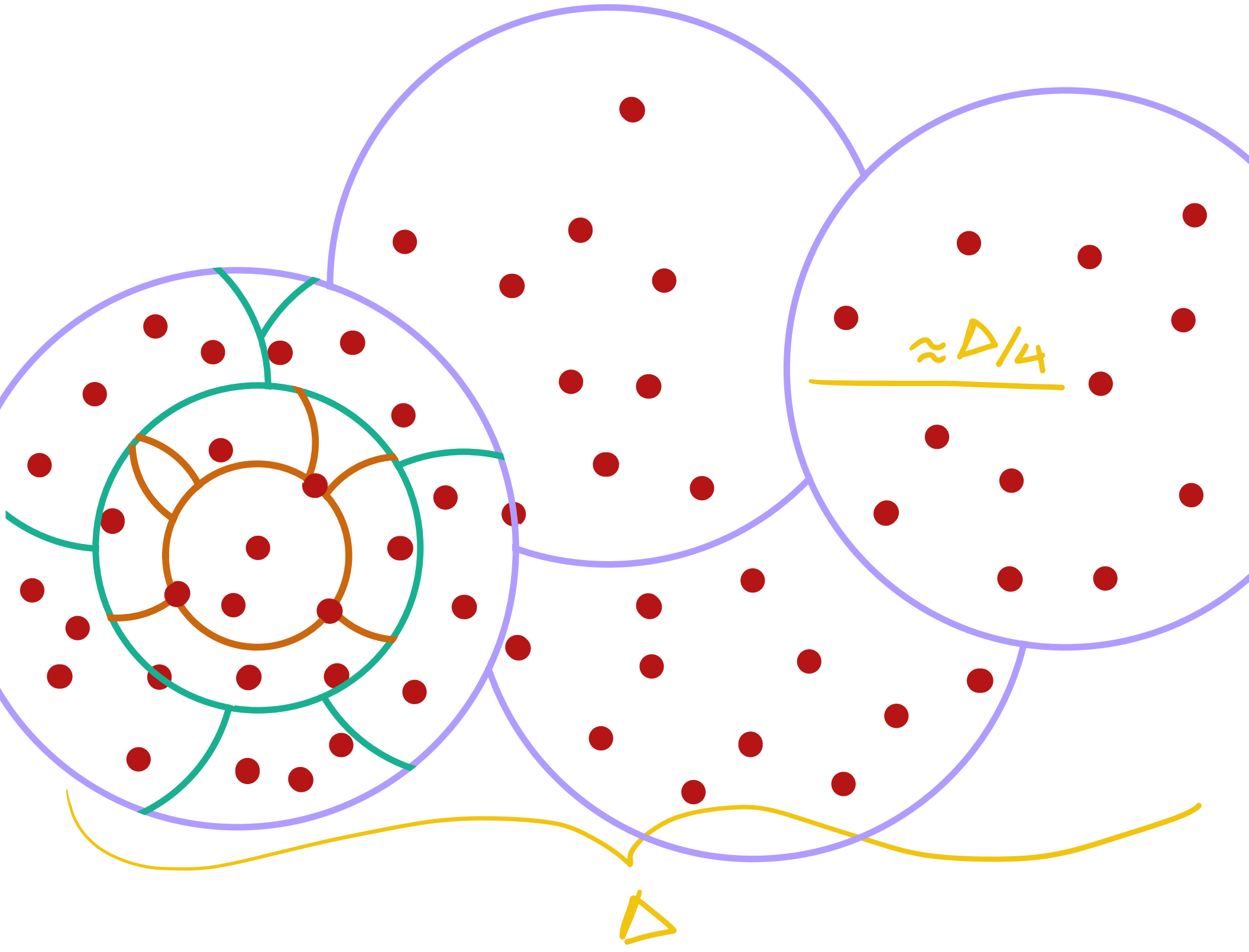
$$E[\text{stretch } e] \leq O(\log n) \quad \forall e \in E$$

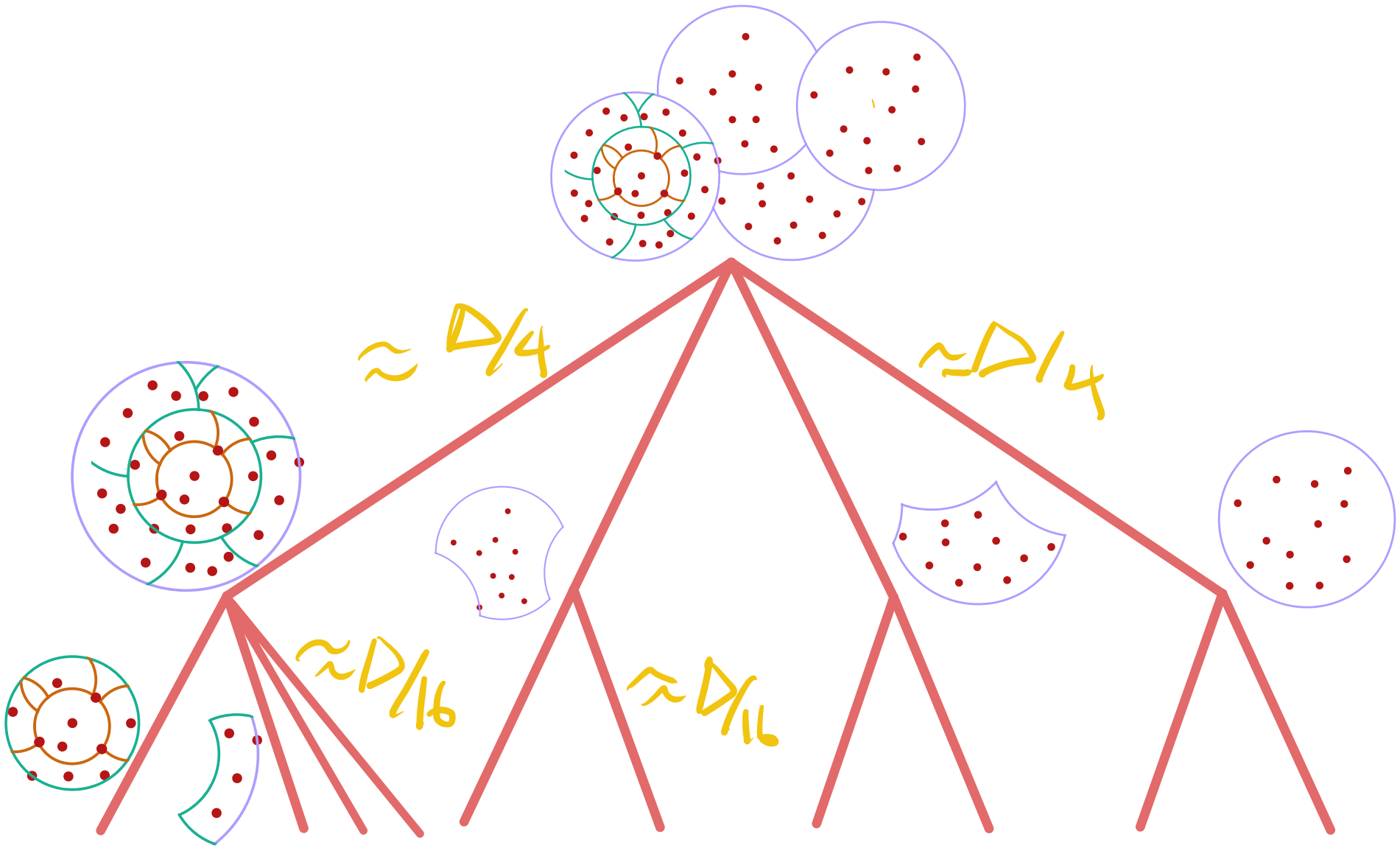




Scoopedy-whoop-de-scoop

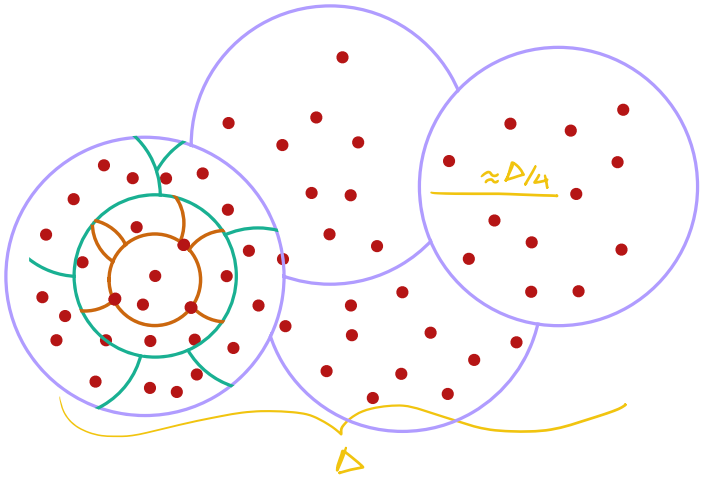


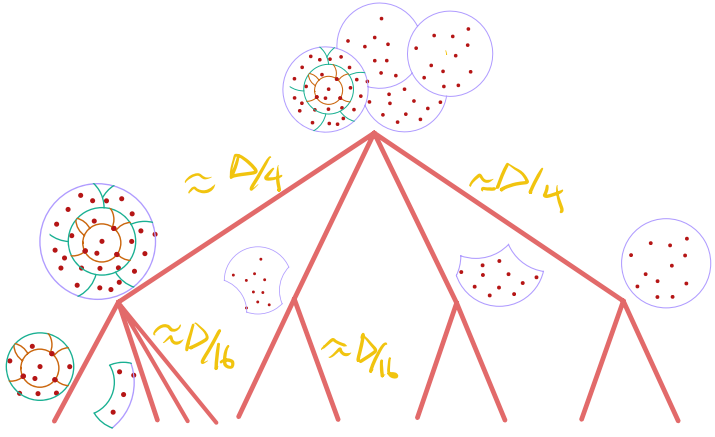

≈ D/4
D


≈ D/4
≈D/4
≈D/16
≈D/16

Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $\alpha \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_i, \alpha 4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.





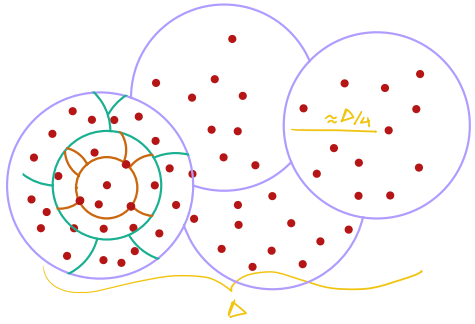
(each node associated w/ subset of V, starting w/ root $= V$
node w/ set S at height i has children correspond.
to (nonempty) sets of the form $S \cap C_{i,j}$)

Analysis

Fix $u, v \in V$.

need to show

$$E[d_T(u,v)] \leq O(\log n)\, d(u,v)$$



Let D = diameter, $L = \lceil \log_4 D \rceil$

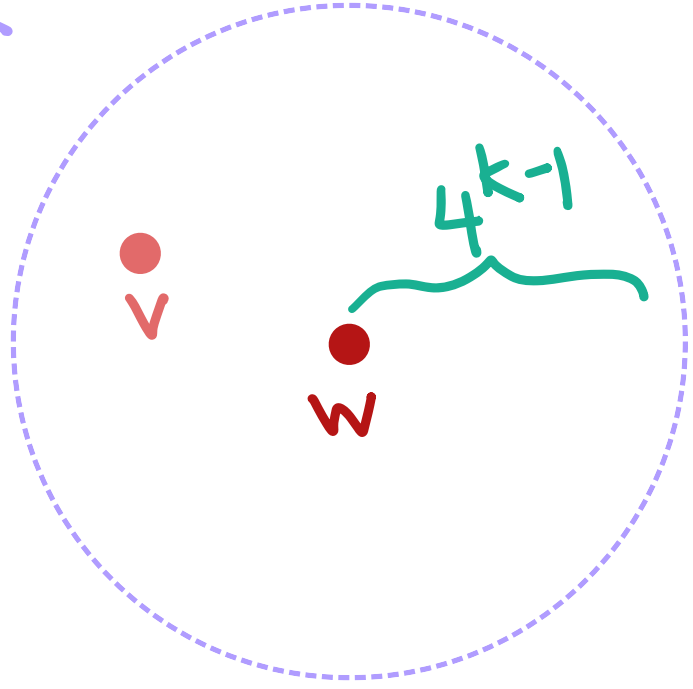
1. let $v_1, \sim, v_n$ order V randomly
   $\alpha \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, \alpha 4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

need to show

$$E[d_T(u,v)] \le O(\log n)\, d(u,v)$$

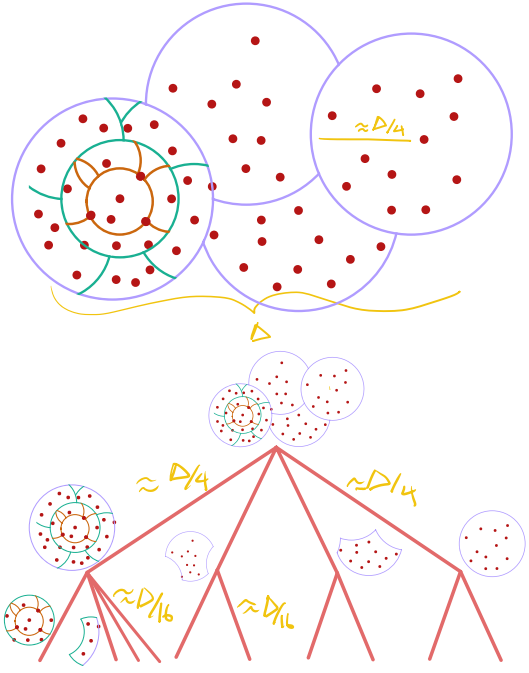
let k = height where u and v are split

then $d_T(u,v) \approx 4^k$



u, v split by cluster centered at some $w \in V$

then we say w "contributes" 4^k to $d_T(u,v)$



Let D = diameter, $L = \lceil \log_4 D \rceil$

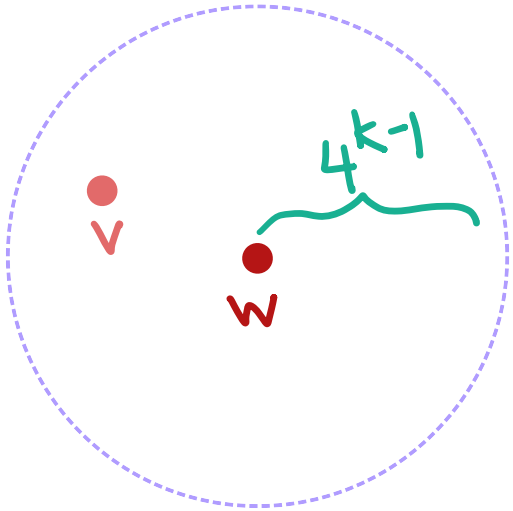
1. let $v_1, \sim, v_n$ order V randomly
   $\alpha \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, \alpha 4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

need to show

$$E[d_T(u,v)] \le O(\log n)\, d(u,v)$$

let k = height where u and v are split

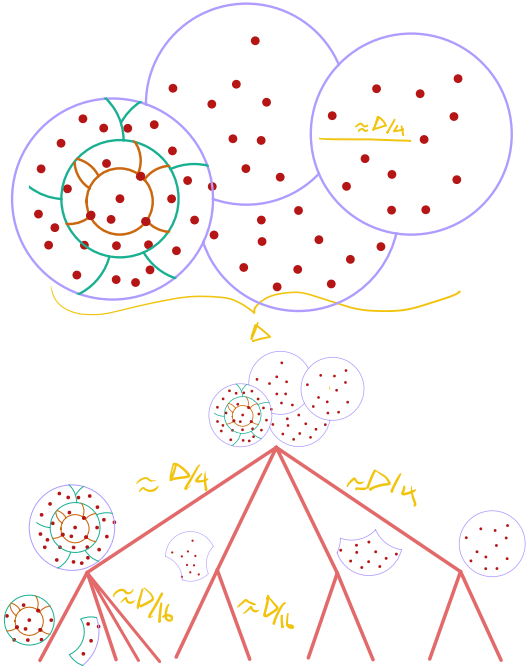
then $d_T(u,v) \approx 4^k$



u, v split by cluster centered at some $w \in V$

w "contributes" 4^k to $d_T(u,v)$

$$d_T(u,v) = O\left(\sum_w (\text{contrib. of } w \text{ to } d_T(u,v))\right)$$



Let D = diameter, $L = \lceil \log_4 D \rceil$

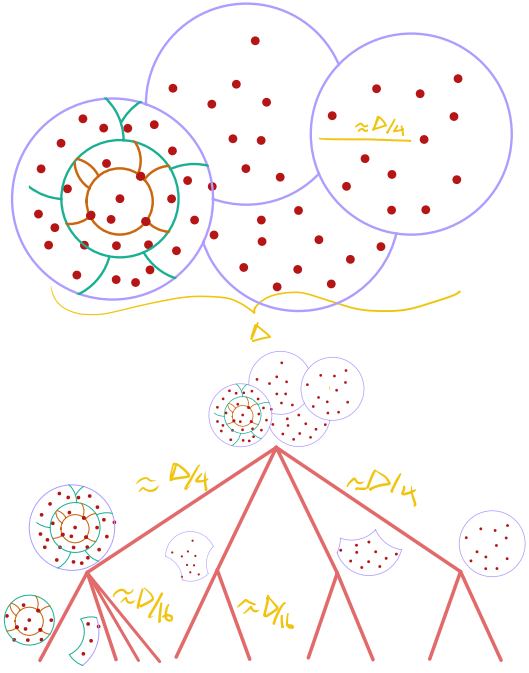
1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

need to show $E[d_T(u,v)] \le O(\log n)\, d(u,v)$

$d_T(u,v) = O(\sum_w (\text{contrib. of } w \text{ to } d_T(u,v)))$

Claim: let w be ℓth closest vertex to u or v. then

$$E[\text{contrib. from } w] \le \frac{4\, d(u,v)}{\ell}$$



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

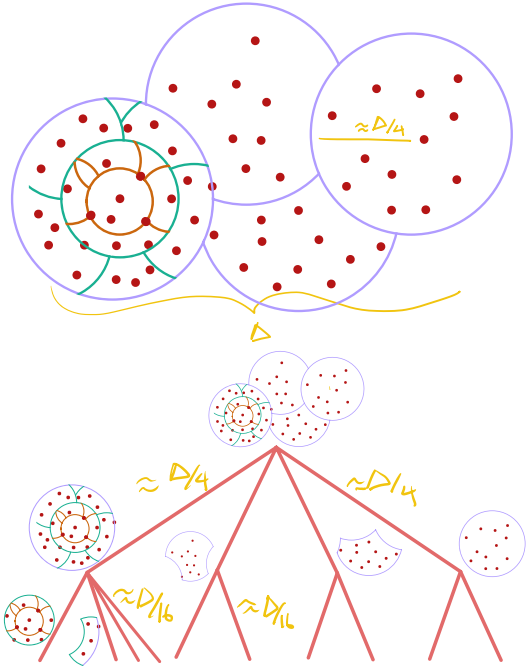
need to show $E[d_T(u,v)] \le O(\log n)\, d(u,v)$

$d_T(u,v) = O(\sum_w (\text{contrib. of } w \text{ to } d_T(u,v)))$

Claim: let w be ℓth closest vertex to u or v. then

$$E[\text{contrib. from } w] \le \frac{4\, d(u,v)}{\ell}$$

if so:



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $\alpha \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, \alpha 4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

Claim: let w be ℓth closest vertex to u or v. then $E[\text{contrib. from } w] \le \frac{4\,d(u,v)}{\ell}$

Proof wlog $d(w,u) \le d(w,v)$

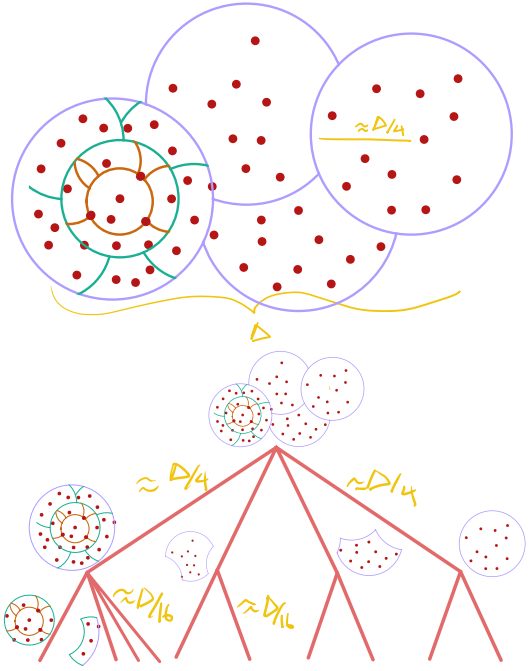
define two necessary events

E_1: $d(w,u) \le d4^k \le d(w,v)$

E_2: w ordered before any of the $\ell-1$ vertices closer to u or v

E_1 and E_2 are independent (!)

$Pr[E_2] = 1/\ell$



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

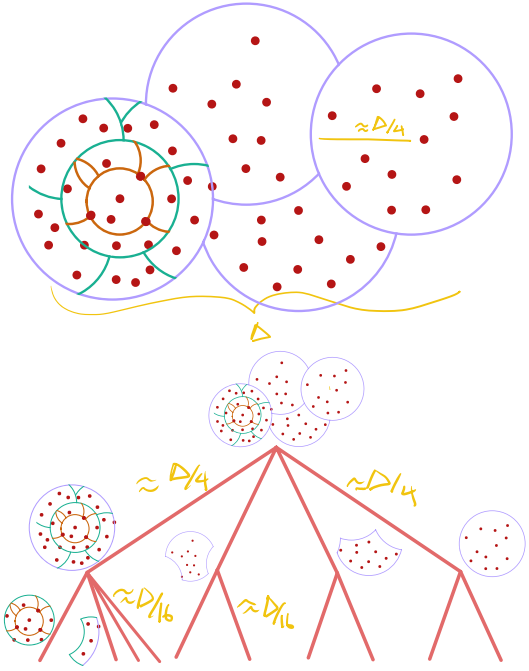
E_1: $d(w,u) \le d4^k \le d(w,v)$

2 cases.

case 1: $d(u,v) \ge 4d(w,u)$

case 2: $d(u,v) \le d(w,u)$

($d(w,u)$ might be much bigger than $d(u,v)$, which is risky)



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

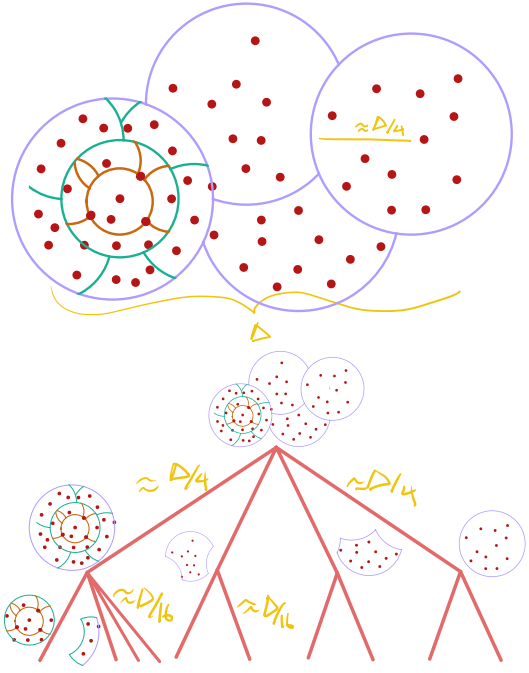
k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

E_1: $d(w,u) \le \alpha 4^k \le d(w,v)$

case 1: $d(u,v) \ge 4d(w,u)$



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $\alpha \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, \alpha 4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

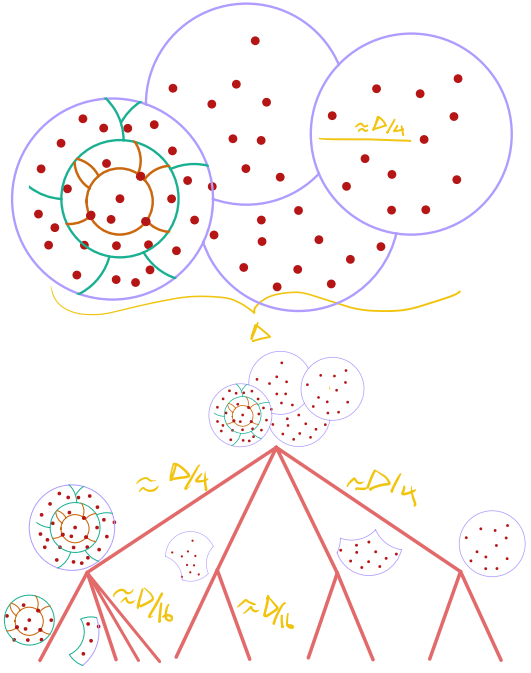
then w "contributes" 4^k to $d_T(u,v)$

E_1: $d(w,u) \le d4^k \le d(w,v)$

case 2: $d(u,v) \le d(w,v)$

($d(w,u)$ might be much bigger than $d(u,v)$, which is risky)

claim: choice of k is unique



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

E_1: $d(w,u) \le \alpha 4^k \le d(w,v)$

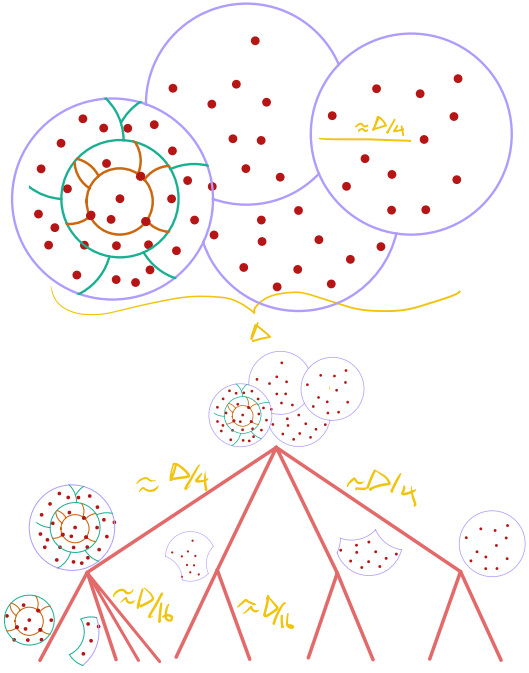
case 2: $d(u,v) \le d(w,v)$

($d(w,u)$ might be much bigger than $d(u,v)$, which is risky)

claim: choice of k is unique

let $\alpha \in [1,2]$, k satisfy E_1

for $k+1$:



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $\alpha \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, \alpha 4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

E_1: $d(w,u) \le d4^k \le d(w,v)$

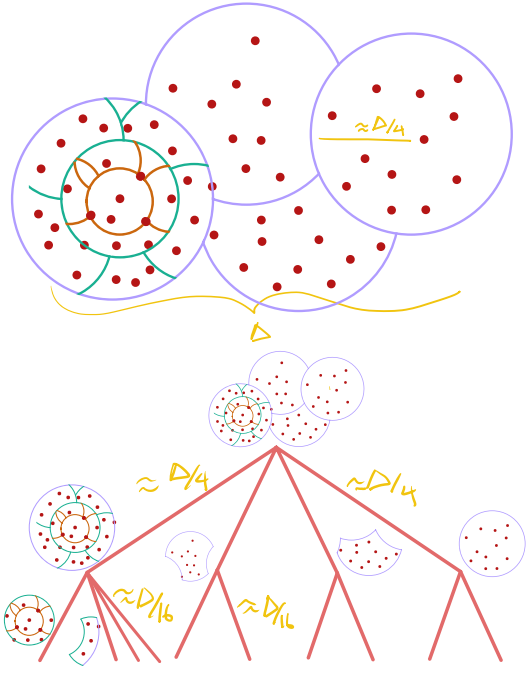
case 2: $d(u,v) \le d(w,v)$

($d(w,u)$ might be much bigger than $d(u,v)$, which is risky)

claim: choice of k is unique

let $d \in [1,2]$, k satisfy E_1

for $k-1$:



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

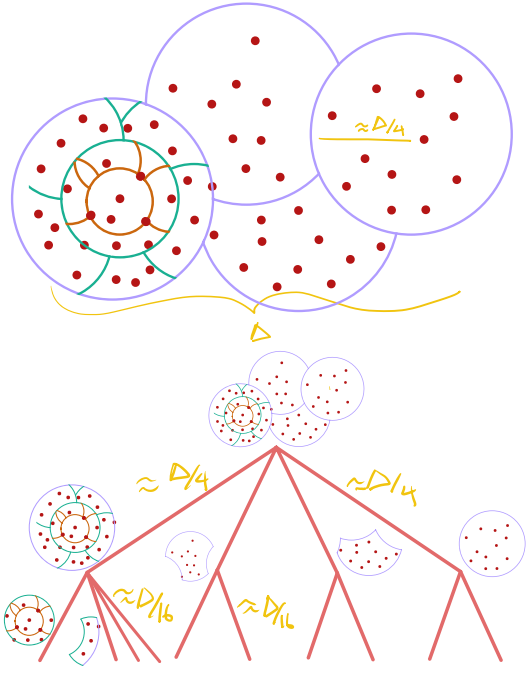
E_1: $d(w,u) \le d4^k \le d(w,v)$

case 2: $d(u,v) \le d(w,v)$

($d(w,u)$ might be much bigger than $d(u,v)$, which is risky)

✓ choice of k is unique

$P[E_1] \le$



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

k = height where u and v are split

u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

E_1: $d(w,u) \le d4^k \le d(w,v)$

case 2: $d(u,v) \le d(w,v)$

($d(w,u)$ might be much bigger than $d(u,v)$, which is risky)

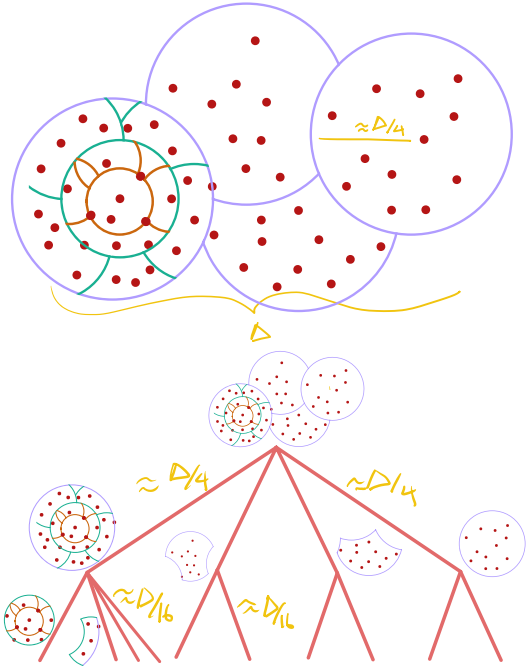
✓ choice of k is unique

✓ $P[E_1] \le \frac{d(u,v)}{4^k}$

$E[\text{contrib. from } w \sim]$

$\le 4^{k+1} \cdot P[E_1] \cdot P[E_2]$

$\le 4\, d(u,v)/\ell$



Let D = diameter, $L = \lceil \log_4 D \rceil$

1. let $v_1, \sim, v_n$ order V randomly
   $d \in [1,2]$ unif. at random
2. for $i = L, L-1, \sim, 0$
   a. for each v_j in order
      i. $C_{i,j} \leftarrow B(v_j, d4^{i-1}) \setminus \bigcup_{k<j} C_{i,k}$
3. Use $C_{i,j}$'s to partition V in a tree from top down.

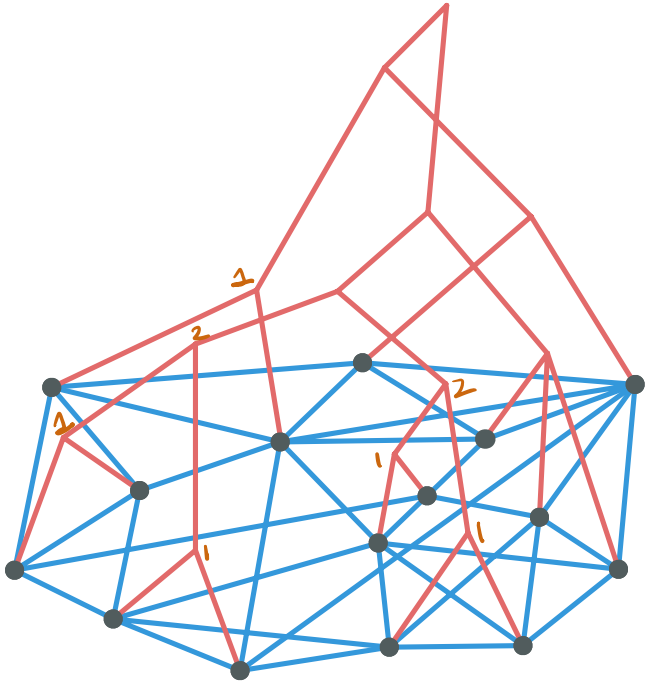
k = height where u and v are split

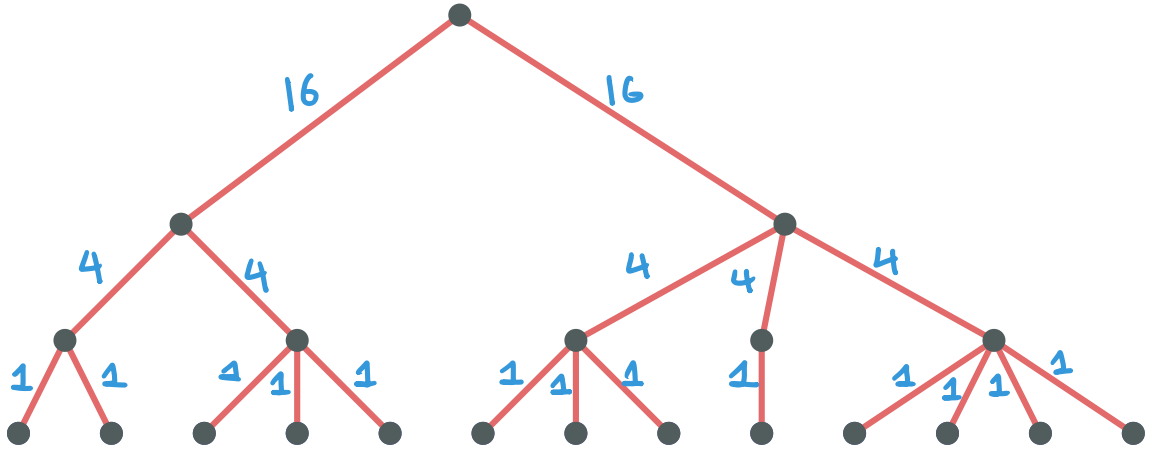
u,v split by cluster centered at some $w \in V$

then w "contributes" 4^k to $d_T(u,v)$

Theorem In randomized polynomial time, randomized hierarchical dominating tree metric w/

$$E[\text{stretch } e] \leq O(\log n) \quad \forall e \in E$$





Other metric-based problems

(sometimes via correspondance between cuts ↔ metrics)

Buy-at-bulk network design

k-median clustering

metric labeling

group Steiner tree

metrical task systems

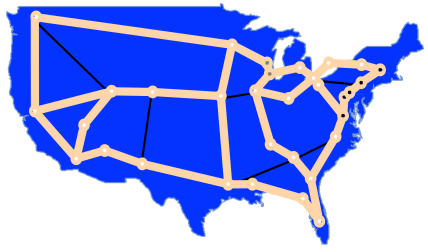
vehicle routing

hierarchical placement

distributed k-server

covering Steiner tree

&



these problems are <u>easy</u> for tree metrics

Chapter 11

Tree Metrics

11.1 Introduction

Many flow and cut problems are ultimately about paths, whether packing paths in flow, cutting paths in cuts, evaluating shortest paths in fractional cuts, etc. Obviously there are many paths between any two points in a graph and this makes all these problems nontrivial. An extremely simple setting, then, would be a graph where there is a unique path between any two vertices: by definition, a *tree*. Many graphs problems become trivial in a tree. Here we will study a bold approach to graph algorithms based on this idea: process the input graph G to produce a tree T that preserves its salient properties, solve the problem on T, and lift the solution back to G. Of course a tree T cannot preserve all of G, so we will only preserve specific properties and only approximately at that, in such a way that is appropriate to the problem at hand.

In this discussion, we will focus on preserving the shortest path metric of a graph. Let G be an undirected graph, and let d_G denote the shortest path metric in G. Let T be a spanning tree of T (with the same edge weights), and let d_T denote the shortest path metric in T. (Of course, the shortest path in T is also the only path.) Ideally, we want $d_T(u, v)$ to resemble $d_G(u, v)$ as much as possible for each edge $e = \{u, v\}$. In general, we have

$$d_G(u, v) \leq d_T(u, v) \text{ for all } u, v \in V$$

simply because T is a subgraph of G. For an edge $e = \{u, v\}$, we say that the *stretch* of e is defined as the ratio

$$(\text{stretch } e) \stackrel{\text{def}}{=} \frac{d_T(u, v)}{d_G(u, v)}.$$

We say that d_T has *uniform stretch* (at most) α, for $\alpha \geq 1$, if every edge has stretch at most α.

A natural goal is to obtain a spanning tree with small uniform stretch. However the n-vertex cycle C_n presents a lower bound of $n-1$. Indeed, any spanning tree T of C_n is obtained by dropping one edge e; this edge e is stretched around the cycle, so to speak, and has stretch $n-1$.

The n-vertex cycle C_n indicates that we cannot, in the worst-case, find spanning trees with uniform stretch better than $n-1$. (We leave it as exercise 11.1 to obtain a matching upper bound.) The rest of this chapter discusses two different approaches that obtain better bounds for relaxations of this problem. (In the lecture, we only discuss the second, randomized one in detail.)

Low-stretch spanning trees. Alon, Karp, Peleg, and West [AKPW95] showed how to compute a spanning tree T where the average stretch among all edges is $n^{o(1)}$, in the sense that

$$\frac{1}{|E|}\sum_{e\in E}(\text{stretch } e) \leq n^{o(1)}$$

Dominating tree metrics. Bartal [Bar96; Bar98] ignored the requirement that T is a spanning tree of G; more generally he sought auxiliary trees T where the vertex of G correspond to the leaves of G, while retaining the property T that $d_T \geq d_G$. He produced *randomized* trees where *for each edge* the average stretch was polylog(n):

$$\mathop{\mathbf{E}}_{T}[(\text{stretch } e)] \leq O(\text{polylog}(n)) \text{ for all } e \in E.$$

Note that this a different sense of "average stretch" then above. We will present an algorithm of [FRT04] building on [Bar98] to obtain (per-edge) average stretch of $O(\log n)$.

11.2 Low-Stretch Spanning Trees

We first present the low-stretch spanning trees of [AKPW95]. Here we recall that we want to compute a spanning tree with low stretch on average over all the edges.

Suppose our goal was to obtain average stretch (roughly) D for a parameter $D > 0$. Let us partition the graph in vertex-disjoint subgraphs each with radius at most $D/2$ from some center vertex, and compute a shortest path tree from each center. Then every edge *within* a neighborhood has stretch at most D; it remains to address the edges that are cut by the partition. We can try to address these edges recursively by contracting every subgraph/subtree into a single vertex, leaving a multigraph G' consisting of the cut edges, and recursing on this graph. This produces a tree T' on the contracted multigraph G'; expanding out the vertices of T' by the underlying

[AKPW95] algorithm for $m^{o(1)}$-average stretch spanning trees:

1. *Compute a low diameter decomposition:* repeatedly, until no vertices remain:
   A. Select a remaining vertex v and compute a shortest path tree of v in the remaining graph.
   B. Remove the set C_v all vertices (remaining) in V within some distance $R' \leq R$ such that

$$|\delta(C_v)| \leq O(\log(m))(1 + |E[C_v]|)$$

   where $E[C_v]$ denotes the set of all (remaining) edges incident to some vertex in v.
2. *Recurse:* Let G' be the multi-graph obtained from the input graph by contracting each C_v to a single vertex. Recurse on G' to obtain a spanning tree T', and return the tree T obtained by replacing each C_v with the corresponding shortest path tree.

---

shortest path trees gives a spanning tree T. Recursively, we might expect that every edge cut edge e has stretch $O(R)$ in T', but this expands out to stretch $O(R^2)$ with respect to T because passing through a vertex in T' actually corresponds to traversing a path of length $O(R)$ in the underlying tree.

So we have a problem where each step of the recursion induces an additional factor of R. Now, recall that we want to preserve *average* stretch. Let $f(m)$ be the stretch obtained by the recursive approach. We have

$$f(m) \leq D(\#\text{ internal edges}) + (D+1)f(\#\text{ external edges}).$$

Studying this recursion, we can see that if the number of external edges was extremely small, then we might hope that it can offset the extra factor of D. How can we minimize the number of exteneral, or cut, edges? Region growing! Low diameter decompositions!

If we partition the graph by region growing techniques, we can ensure that

$$(\#\text{ external edges}) \leq \frac{O(\log m)}{R} m.$$

This revises the recursive bound as

$$f(m) \leq Dm + (D+1)f(c\log(m)m/D)$$

for a constant $c > 0$. For $D = e^{\sqrt{\log(m)/\log\log m}}$ the recursion is bounded by $f(m) = e^{O\left(\sqrt{\log m \log\log m}\right)}$. (The calculations are given below.)

Theorem 11.1. *The AKPW algorithm returns a spanning tree with average stretch* $e^{O\left(\sqrt{\log m \log\log m}\right)}$.

Solving the recurrence. We have

$$f(m) \leq Dm + (D+1)f(c_0 \log(m)m/D)$$

where $\epsilon = c_0 \log(m)/D$ for a constant c_0. The height of the recursion is

$$h = O\Big(\log_{D/c_0 \log(m)} m\Big) = O(\log(m)/\log(D/c_0 \log(m))) = O\left(\frac{\log(m)}{\log(D)}\right)$$

assuming $D = \Omega(\log m)$. Unrolling the recursion gives

$$f(m) \leq O(Dm) \sum_{i=0}^{h} ((1 + 1/D)c_0 \log(m))^i \leq Dme^{O(h \log\log m)}.$$

To minimize the RHS, we can instead minimize the logarithm of the RHS, $\log(m) + \log(D) + O(h \log\log m)$. Choosing D to make the last two terms (roughly) equal, we have

$$\log(D) = \frac{\log(m)\log\log(m)}{\log(D)},$$

hence

$$\log(D) = \sqrt{\log(m)\log\log(m)}.$$

Then $D = e^{\sqrt{\log(m)\log\log(m)}}$ gives

$$f(m) \leq me^{O\left(\sqrt{\log(m)\log\log(m)}\right)},$$

as desired.

11.3 Hierarchical Tree Metrics

The [AKPW95] algorithm was able to obtain low stretch *in total*. This means that a few unlucky edges might have extremely high stretch. In this section we want every edge to have low stretch, in some sense. Unfortunately we already know that it is impossible to guarantee $o(n)$ stretch for every edge simultaneously. But a different possibility open up when we allow for randomization. Perhaps we can output a *randomized* tree T, such that for each edge e, the expected stretch of e is $o(n)$. Such a claim does *not* contradict the lower bound for the n-vertex cycle; in fact, one can get constant expected stretch for the cycle which we leave as exercise 11.2.

To build some intuition, let us start from the [AKPW95] algorithm for inspiration (even though ultimately we will not produce a spanning tree). A high level goal is to inject randomization so that every edge has a decent chance at having low stretch. To this end there are at least two natural ways to introduce randomization into [AKPW95].

1. We can make the radii of the clusters randomized, instead of a deterministic function of the total number of edges cut.

2. Second, the order of vertices that center the clusters can be randomized, which would seem most equitable.

Both of these ideas will be reflected at a high level in the following algorithm which we now present.

The algorithm we present will produced a randomized *hierarchical tree metric* over V. This means that the tree T will be rooted, with V at the leaves, and the edges between height i and height $i-1$ have length α^i for a fixed constant α. Here we choose $\alpha = 4$ to simplify calculations, though we note that $\alpha = 2$ is more common, and the analysis can be adjusted to accommodate any fixed constant. The convenience of a hierarchical tree T is that the tree distance $d_T(u, v)$ is entirely determined by the height of their least common ancestor, and within a constant factor of the biggest edges at the top of the corresponding subtree. This additional structure turns out to be useful for several other problems.

We now present [FRT04]'s randomized algorithm. We assume the input is an edge-weighted graph where the minimum edge length is normalized to 1. We let $D = \max_{u,v} d(u, v)$ denote the diameter of the graph. Below we describe the algorithm in detail and first we give a high level description. For every radius of the form $\alpha 4^i$, we are randomly scooping out balls of size $\alpha 4^i$, where $\alpha \in [1, 2]$ is drawn uniformly at random, and the vertices at the center of the balls are in random order. A key and subtle point is that we use the *same* (random) α and ordering for every i. the

intersections of these balls (across i's) induce a laminar family of sets over V which are arranged as a tree.

[FRT04]'s algorithm producing a randomized hierarchical tree metric.

1. Let $v_1, \dots, v_n$ be a uniformly random ordering of V. Let $L = \lceil \log_4 D \rceil$. Let $\alpha \in [1, 2]$ be drawn uniformly at random.
2. For i from L down to 0,
   (a) For each vertex v_j in order,
      i. Let $C_{i,j}$ be the set of vertices at distance at most $\alpha 4^{i-1}$ from v_j, excluding any vertex already included by the cluster of a previous $C_{i,j}$.
3. We use the $C_{i,j}$'s to arrange the vertices as leaves in a tree T hierarchically as follows. For each intermediate node x at height i, the leaves in the subtree rooted at x corresponds to a set of vertices with diameter at most 4^i. The root at height $L+1$ corresponds to V. The nodes at height L correspond to the clusters $C_{L,j}$. In general, for a node x at height i corresponding to a set $S \subseteq V$, its children correspond to the (nonempty) intersections of S with clusters $C_{\ell,j}$. (Here a cluster center v_j may not be in S.) Observe the leaves (at height 0) each correspond to a single vertex V because $C_{0,j} = \{v_j\}$ for all j. Each edge descending from height i is given weight 4^i.

So much for the algorithm. Here, then, is the key claim.

Theorem 11.2. *[FRT04]'s algorithm produces a randomized hierarchical tree T such that for each edge e, $\mathbf{E}[(\text{stretch } e)] \leq O(\log n)$.*

To prove the theorem, fix an edge $e = \{u, v\}$. Recall that $d(u, v)$ is decided, up to a constant factor, by the height k where u and v are first separated. (Then $d(u, v) = O\left(4^k\right)$.) When this occurs, u and v are separated in particular by a cluster of radius $\alpha 4^k$ centered at some vertex w; in this event, we say that w "contributes" 4^k to $d(u, v)$ (which upper bounds the diameter of the remaining vertices). By this terminology, we have

$$d_T(u, v) \leq \sum_w O(1)(\text{contrib. of } w \text{ to } d_T(u, v)). \tag{11.1}$$

Now, fix a vertex w. Suppose that w was the ℓth closest vertex to u or v (i.e., with respect to $\min\{d(w,u), d(w,v)\}$). The key lemma, which we analyze below, is that

$$\mathbf{E}[\text{contrib. from } w \text{ to } d_T(u,v)] \leq O(1/\ell)d(u,v).$$

Taking expectations of eq. (11.1) and applying the bound above to each w gives $O(\log n)$ stretch, as desired.

It remains to prove the key lemma, as follows.

Lemma 11.3. *Let w be the ℓth closest vertex to u or v. Then*

$$\mathbf{E}[\textit{contrib. from } w \textit{ to } d_T(u,v)] \leq 4d(u,v)/\ell.$$

Proof. We first observe that w contibutes to $d_T(u,v)$ only if the following two events both occur.

E_1. $d(w,u) \leq \alpha 4^k \leq d(w,v)$ for some k.

E_2. w is ordered before any of the $\ell - 1$ vertices that are closer to u or v.

Indeed, the necessity of the first condition is clear. The second is necessary because if any closer vertex would otherwise cluster either u or v (or both) before w.

We also observe that the above conditions are *independent*, since the first event depends (only) on α and the second event depends on the random ordering, which are independent. It is also clear that the second event E_2 occurs with probability $1/\ell$. It remains to analyze E_1. We have two cases.

Case 1: $d(u,v) \geq 4d(w,u)$. Then for any k satisfying the inequality in E_1, this inequality and the triangle inequality imply that

$$4^{k+1} \leq 2d(w,v) \leq 2(d(w,u) + d(u,v)) \leq (5/2)d(u,v).$$

Thus the contribution from w is at most $O(d(u,v))$. It follows that

$$\mathbf{E}[\text{contrib. from } w \text{ to } d_T(u,v)] \leq 2.5d(u,v)\,\mathbf{P}[E_2] = 2.5d(u,v)/\ell,$$

as desired.

Case 2: $d(u,v) \leq d(w,u)$. We first note that there may not be any k in item E_1 satisfying inequality E_1; if not, then the claim is immediate. Henceforth we assume such a k exists. We claim that the choice of k is unique. Indeed, suppose there exists some choice of k and $\alpha \in [1,2]$ such that the inequality in E_1 holds. We have

$$d(w,u) \overset{(a)}{\geq} d(w,v) - d(u,v) \overset{(b)}{>} 4^k - 4^k/2 > 2 \cdot 4^{k-1}$$

which rules out smaller values of k. Here (a) is by the the triangle inequality and (b) is by assumption on α. To rule out larger values of k, we have

$$d(w,v) \leq d(w,u) + d(u,v) < 4^{k+1}$$

by similar reasoning.

Thus the choice of k in E_1 is unique; fix k as such. Now we have

$$\mathbf{P}[E_1] \leq \frac{|[d(w,u), d(w,v)]|}{|[4^k, 2 \cdot 4^k]|} \overset{\text{(c)}}{\leq} \frac{d(u,v)}{4^k}$$

by (c) the triangle inequality. Thus

$$\mathbf{E}[\text{contrib. from } w \ ...] \leq 4^{k+1} \mathbf{P}[E_1] \mathbf{P}[E_2] \leq 4d(u,v)/\ell,$$

as desired. □

11.4 Exercises

Exercise 11.1. Design and analyze an algorithm that computes a spanning tree with uniform stretch $n-1$ (matching the lower bound induced by the cycle).

Exercise 11.2. For the n-vertex cycle C_n, describe a randomized tree metric where each edge has expected stretch $O(1)$.

Exercise 11.3. Recall that the low-stretch spanning tree of [AKPW95] obtained average stretch $n^{o(1)}$. We consider extensions to the weighted average. Let $w(e) \in \mathbb{R}_{>0}$ be a positive weight for every edge, and let $W = \sum_{e \in E} w(e)$ be the total weight. We assume for simplicity that the weights are between 1 and $\text{poly}(n)$. We still treat the edges as unit length edges. Design and analyze and algorithm to compute a spanning tree T such that

$$\frac{1}{W} \sum_{e \in E} w(e)(\text{stretch } e) \leq n^{o(1)}.$$

Exercise 11.4. Show how to use the randomized tree metric to randomly round the sparsest cut LP and obtain $O(\log n)$-approximation for sparsest cut.

Exercise 11.5. Prove that the $O(\log n)$ bound is tight for tree metrics (up to constants).