Parallel Algorithms for Geometric Graph Problems

Grigory Yaroslavtsev

http://grigory.us

Appeared in STOC 2014, joint work with Alexandr Andoni, Krzysztof Onak and Aleksandar Nikolov.

"The Big Data Theory"

What should TCS say about big data?

- This talk:
  - **Running time**: (almost) linear, sublinear, …
  - **Space**: linear, sublinear, …
  - **Approximation**: $(1+\epsilon)$, best possible, …
  - **Randomness**: as little as possible, …

- Special focus today: **round** complexity

Round Complexity

Information-theoretic measure of performance

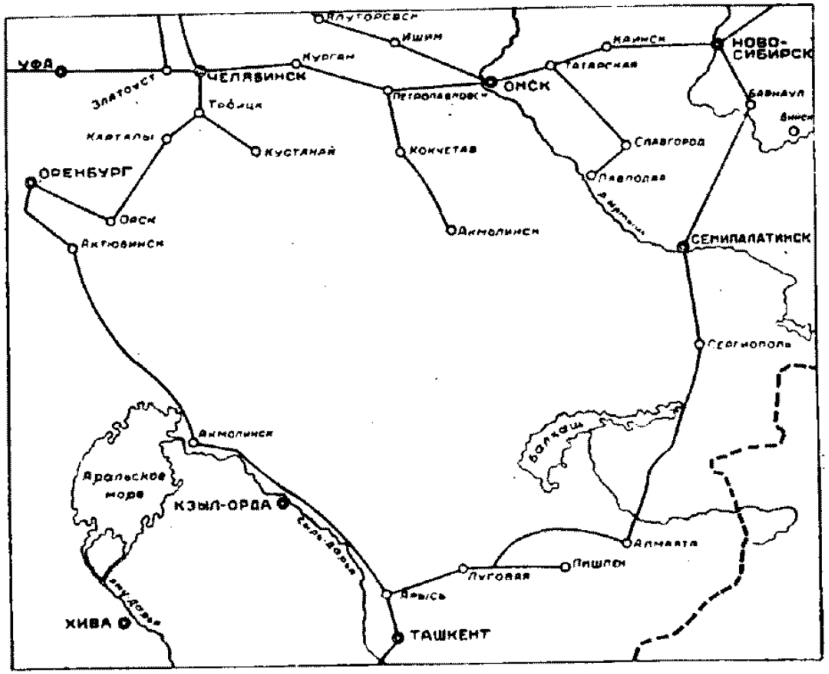
- Tools from information theory (Shannon'48)
- Unconditional results (lower bounds)

Example:

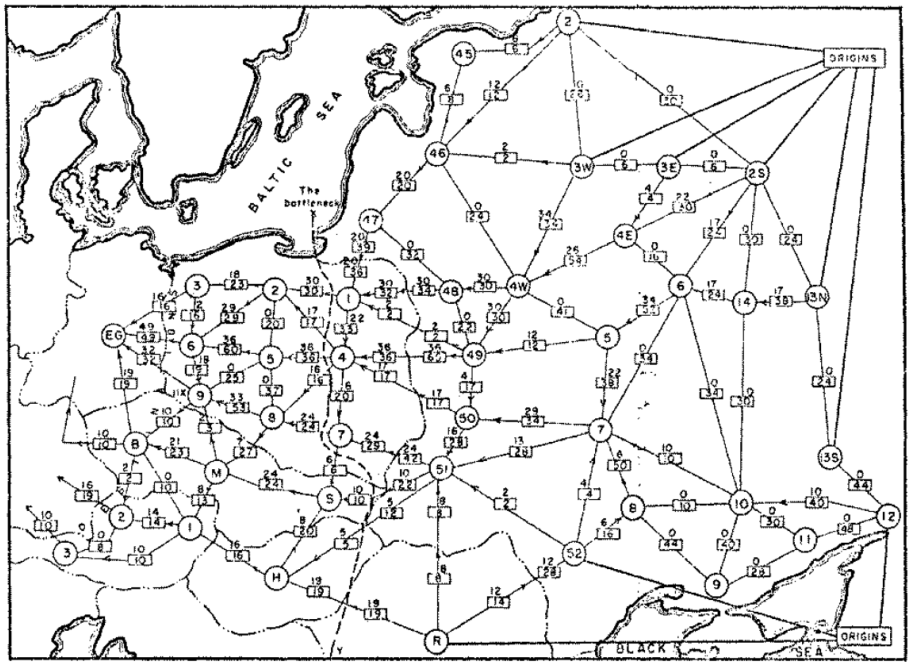
- Approximating Geometric Graph Problems

Approximation in Graphs

1930-50s: Given a graph and an optimization problem…



Transportation Problem:
Tolstoi [1930]



Minimum Cut (RAND):
Harris and Ross [1955] (declassified, 1999)

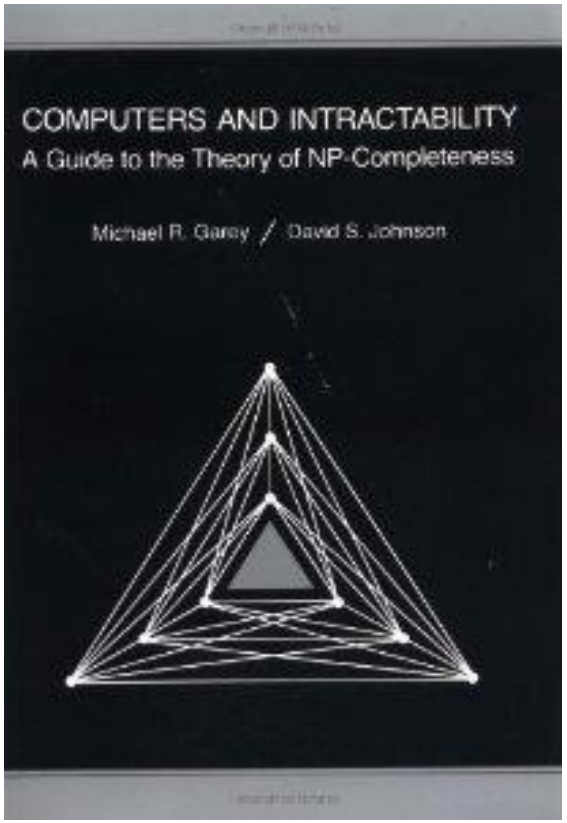
Approximation in Graphs

1960s: Single processor, main memory (IBM 360)

Approximation in Graphs

1970s: NP-complete problem – hard to solve exactly in time polynomial in the input size

“Black Book”



Approximation in Graphs

Approximate with multiplicative error $\boldsymbol{\alpha}$ on the **worst-case** graph G:

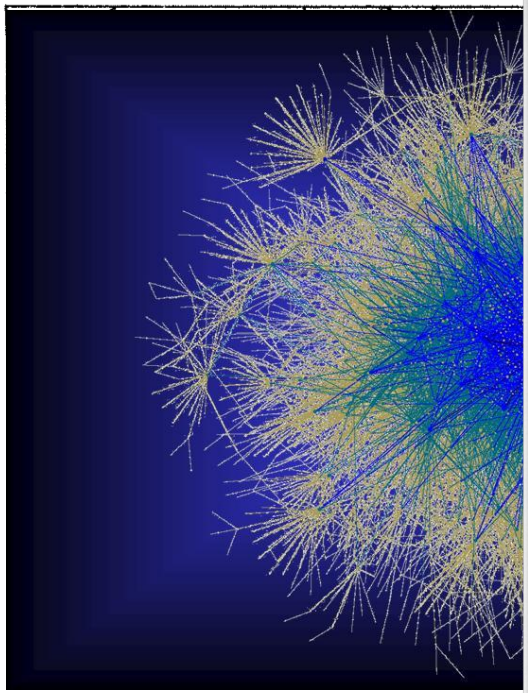
$$max_G \frac{Algorithm(G)}{Optimum(G)} \leq \boldsymbol{\alpha}$$

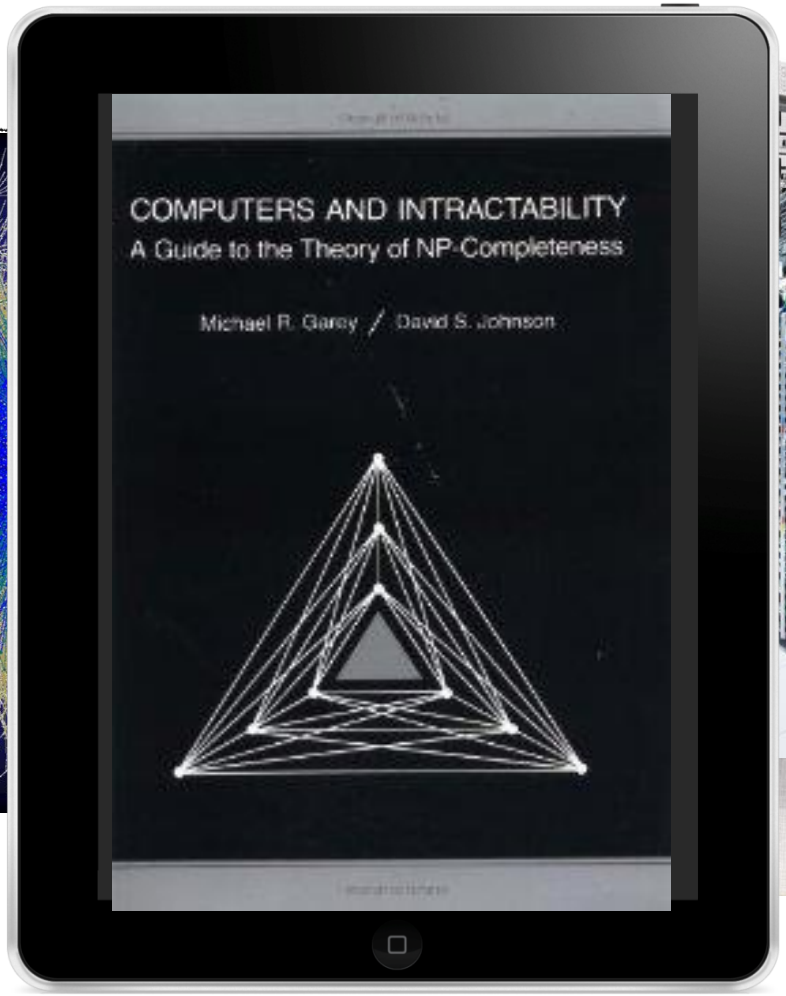
Generic methods:

- Linear programming
- Semidefinite programming
- Hierarchies of linear and semidefinite programs
- Sum-of-squares hierarchies
- …

The New: Approximating Geometric Problems in Parallel Models

1930-70s to 2014

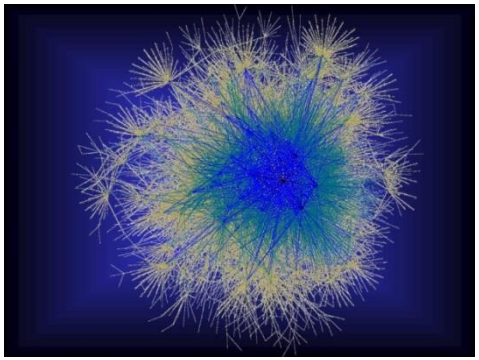




The New: Approximating Geometric Problems in Parallel Models

Geometric graph (implicit):

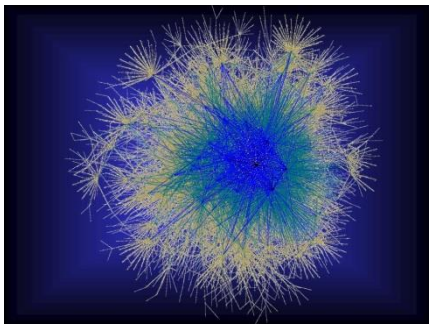
Euclidean distances between **n** points in $\mathbb{R}^{d}$



Already have solutions for old NP-hard problems (Traveling Salesman, Steiner Tree, etc.)

- Minimum Spanning Tree (clustering, vision)
- Minimum Cost Bichromatic Matching (vision)

Geometric Graph Problems

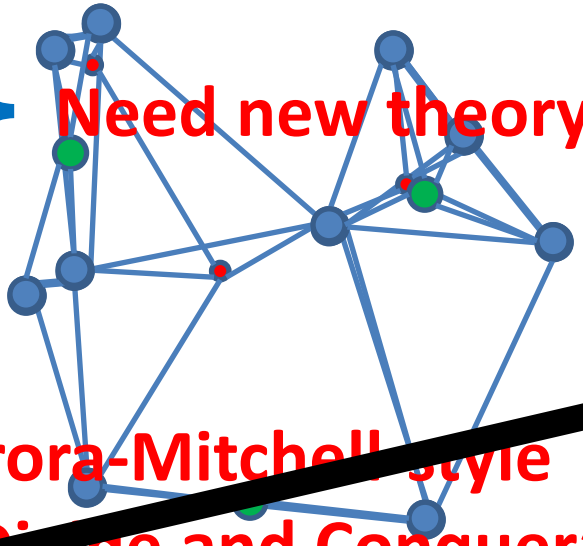


Combinatorial problems on graphs in $\mathbb{R}^d$

Polynomial time (easy)

- Minimum Spanning Tree
- Earth-Mover Distance =

Min Weight Bi-chromatic Matching



Need new theory!

NP-hard (hard)

- Steiner Tree
- Traveling Salesman
- Clustering (k-medians, facility location, etc.)

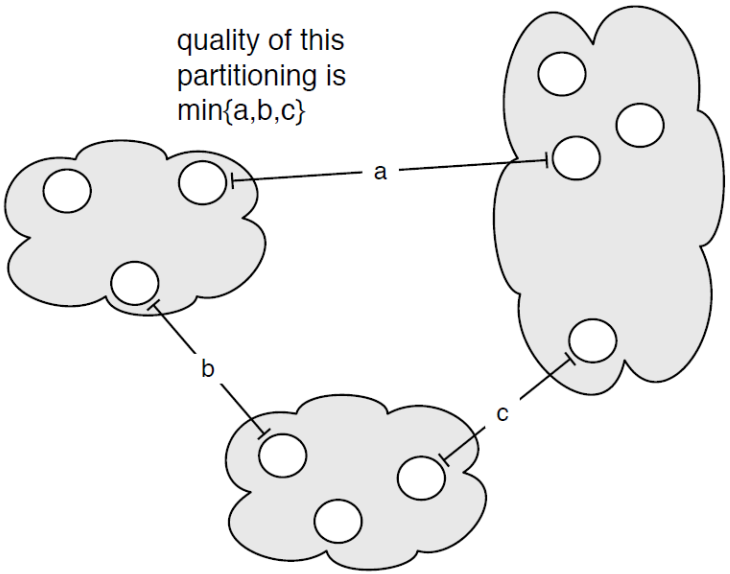
Arora-Mitchell style "Divide and Conquer", easy to implement in Massively Parallel Computational Models

MST: Single Linkage Clustering

- [Zahn'71] **Clustering** via MST (Single-linkage):

k clusters: remove $\boldsymbol{k-1}$ longest edges from MST

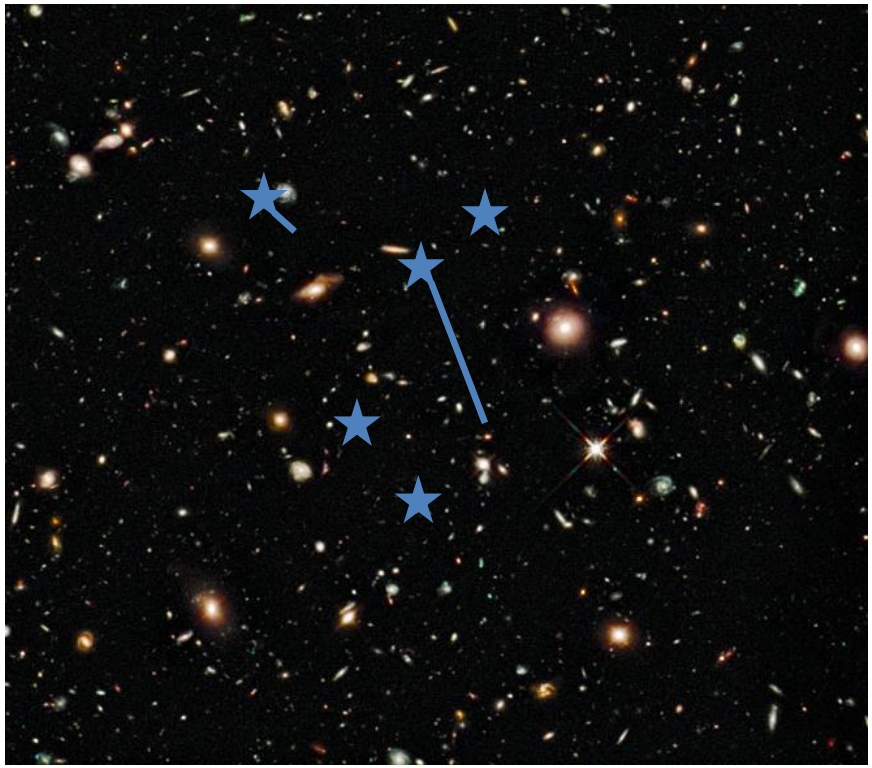
- Maximizes **minimum** intercluster distance



[Kleinberg, Tardos]

Earth-Mover Distance

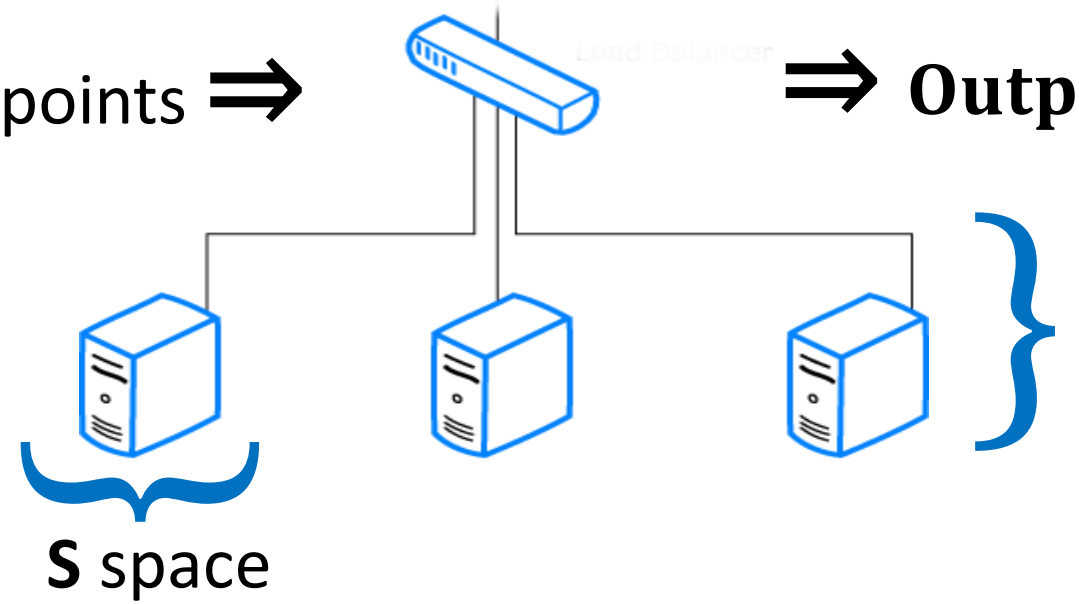
- Computer vision: compare two pictures of moving objects (stars, MRI scans)



Computational Model

- **Input**: **n** points in a **d**-dimensional space (**d** constant)
- $\boldsymbol{M}$ machines, space $\boldsymbol{S}$ on each ($\boldsymbol{S} = \boldsymbol{n}^{\alpha}$, $0 < \alpha < 1$)
  - Constant overhead in total space: $\boldsymbol{M} \cdot \boldsymbol{S} = O(\boldsymbol{n})$
- **Output**: solution to a geometric problem (size O($\boldsymbol{n}$))
  - Doesn't fit on a single machine ($\boldsymbol{S} \ll \boldsymbol{n}$)

Input: $\boldsymbol{n}$ points ⇒



⇒ **Output**: *size* $\boldsymbol{O(n)}$

$\boldsymbol{M}$ machines

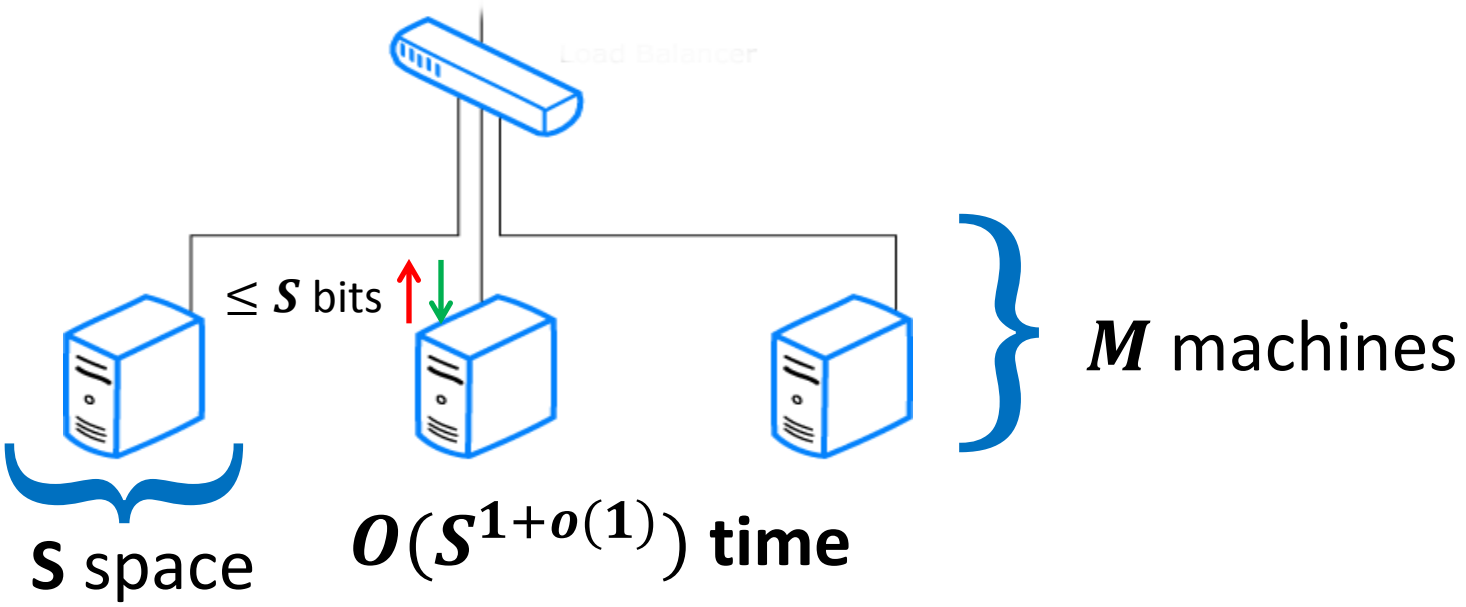
S space

Computational Model

- Computation/Communication in $\boldsymbol{R}$ rounds:
  - Every machine performs a **near-linear time** computation => Total running time $O(n^{1+o(1)}\boldsymbol{R})$
  - Every machine **sends/receives at most $\boldsymbol{S}$ bits** of information => Total communication $O(\boldsymbol{nR})$.

Goal: Minimize $\boldsymbol{R}$.

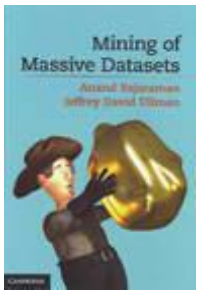
Our work: $\boldsymbol{R}$ = constant.



MapReduce-style computations

What I won't discuss today

- PRAMs (**shared memory**, multiple processors) (see e.g. [Karloff, Suri, Vassilvitskii'10])
  - Computing XOR requires $\widetilde{\Omega}(\log n)$ rounds in CRCW PRAM
  - Can be done in $O(\log_s n)$ rounds of MapReduce
- Pregel-style systems, Distributed Hash Tables (see e.g. Ashish Goel's class notes and papers)
- Lower-level implementation details (see e.g. Rajaraman-Leskovec-Ullman book)



Models of parallel computation

- **Bulk-Synchronous Parallel Model** (BSP) **[Valiant,90]**

  Pro: Most general, generalizes all other models

  Con: Many parameters, hard to design algorithms

- **Massive Parallel Computation** [Feldman-Muthukrishnan-Sidiropoulos-Stein-Svitkina'07, Karloff-Suri-Vassilvitskii'10, **Goodrich-Sitchinava-Zhang'11, ..., Beame, Koutris, Suciu'13**]

  Pros:

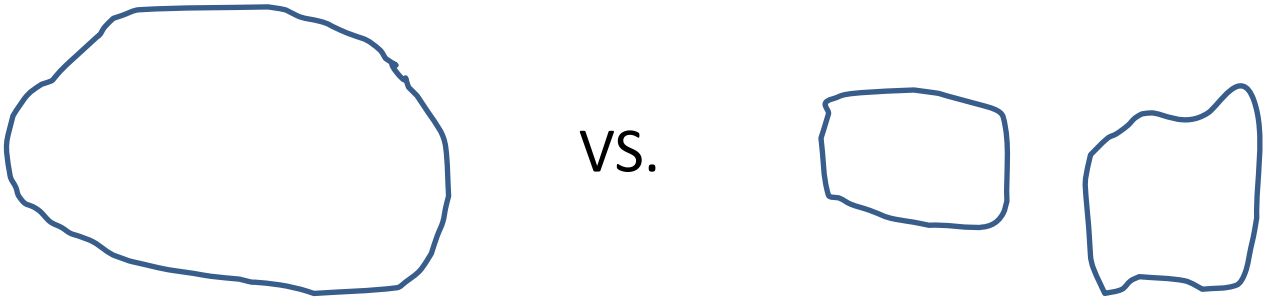
  - Inspired by **modern** systems (Hadoop, MapReduce, Dryad, Pregel, …)
  - Few parameters, **simple** to design algorithms
  - **New algorithmic ideas**, robust to the exact model specification
  - **# Rounds** is an information-theoretic measure => can prove unconditional lower bounds
  - Between **linear sketching** and **streaming with sorting**

Previous work

- **Dense graphs** vs. sparse graphs
  - **Dense**: $S \gg n$ (or $S \gg$ solution size)

    "Filtering" (Output fits on a single machine) [Karloff, Suri Vassilvitskii, SODA'10; Ene, Im, Moseley, KDD'11; Lattanzi, Moseley, Suri, Vassilvitskii, SPAA'11; Suri, Vassilvitskii, WWW'11]
  - Sparse: $S \ll n$ (or $S \ll$ solution size)

    Sparse graph problems appear hard (**Big open question**: (s,t)-connectivity in $o(\log n)$ rounds?)



Large geometric graphs

- Graph algorithms: **Dense graphs** vs. sparse graphs
  - **Dense**: $S \gg n$.
  - Sparse: $S \ll n$.

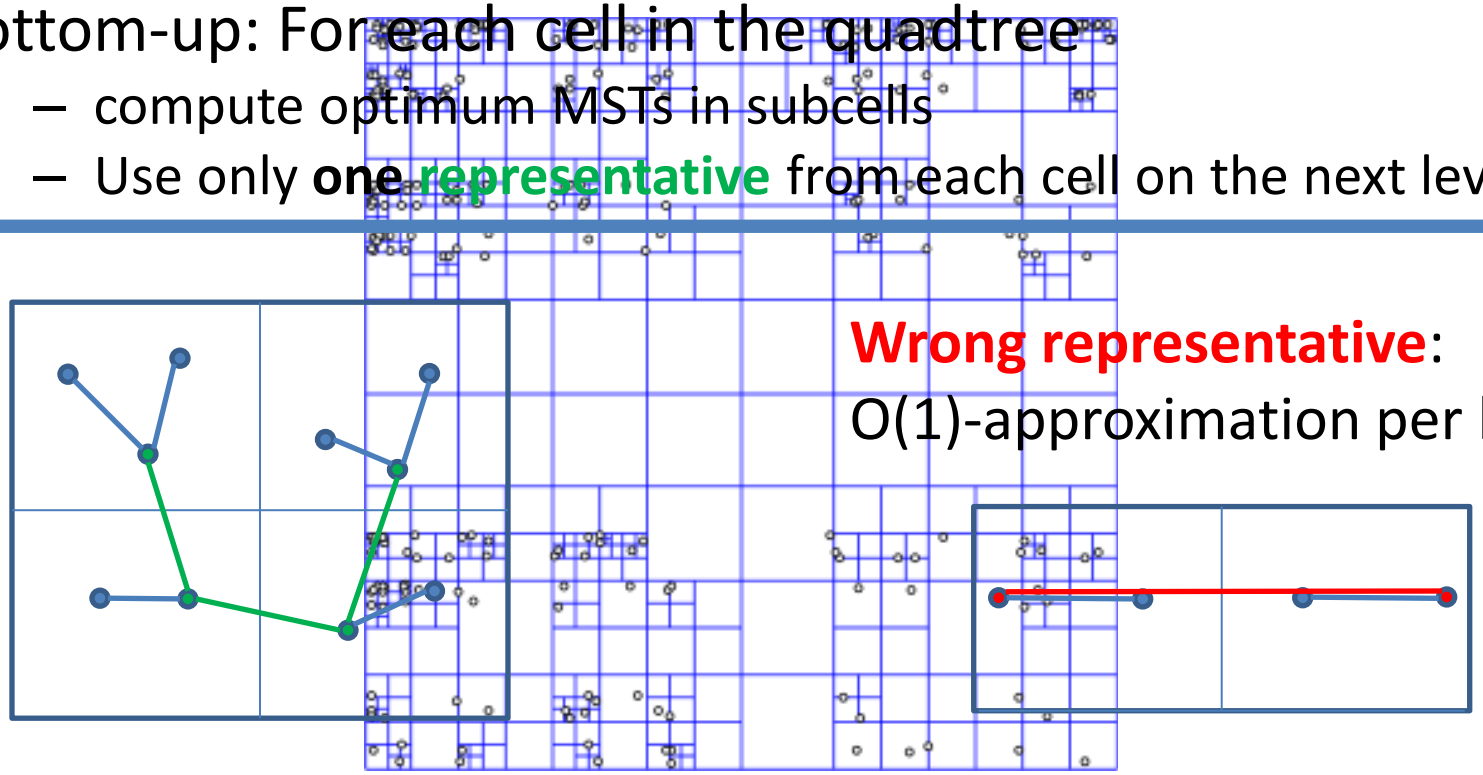
- **Our setting:**
  - Dense graphs, sparsely represented: O(n) space
  - Output doesn't fit on one machine ($S \ll n$)
- **Today:** $(1+\epsilon)$-approximate MST
  - $d = 2$ (easy to generalize)
  - $R = \log_S n = O(1)$ rounds ($S = n^{\Omega(1)}$)

$O(\log n)$-MST in $R = O(\log n)$ rounds

- Assume points have integer coordinates $[0, \dots, \Delta]$, where $\Delta = O(n^2)$.

Impose an $O(\log n)$-depth quadtree

Bottom-up: For each cell in the quadtree
- compute optimum MSTs in subcells
- Use only **one representative** from each cell on the next level



Wrong representative:
O(1)-approximation per level

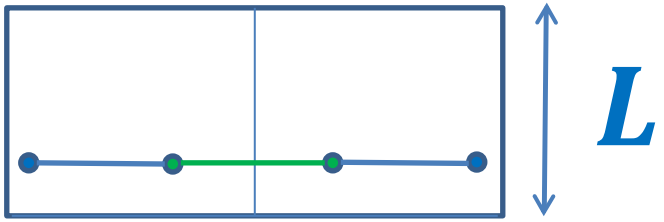
ϵL-nets

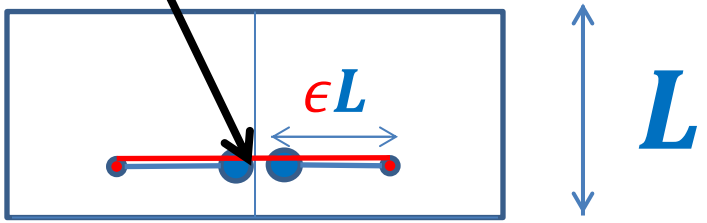
- ϵL-net for a cell C with side length L:
  Collection **S** of vertices in C, every vertex is at distance <= ϵL from some vertex in **S**. (Fact: Can efficiently compute ϵ-net of size $O\left(\frac{1}{\epsilon^2}\right)$)

Bottom-up: For each cell in the quadtree
– Compute optimum MSTs in subcells
– Use ϵL-net from each cell on the next level

- **Idea**: Pay only $O(\epsilon L)$ for an **edge** cut by cell with side L
- Randomly shift the quadtree:
  $\Pr[cut\ edge\ of\ length\ \ell\ by\ L] \leq \ell / L$ – charge errors
  O(1)-approximation per level

Wrong representative





Randomly shifted quadtree

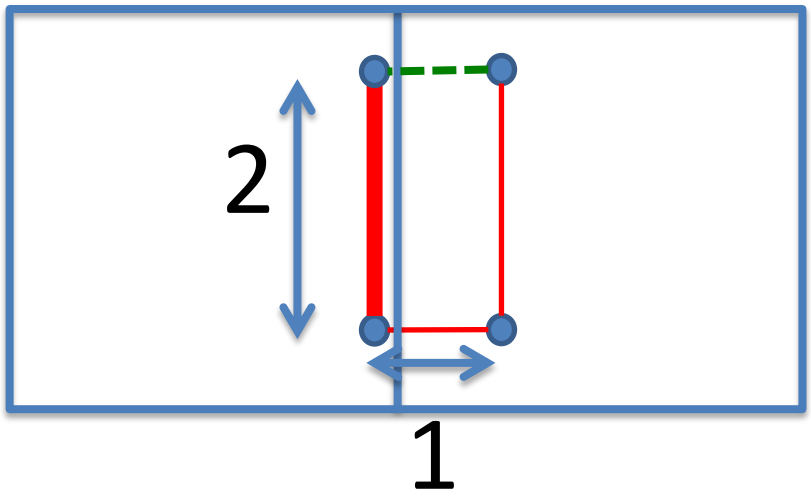
- Top cell shifted by a random vector in $[0, \boldsymbol{L}]^2$

Impose a **randomly shifted** quadtree (top cell length $\boldsymbol{2\Delta}$)

Bottom-up: For each cell in the quadtree

– Compute optimum MSTs in subcells

– Use $\boldsymbol{\epsilon L}$-net from each cell on the next level



Pay **5** instead of **4**

Bad Cut

Pr[Bad Cut] = Ω(1)

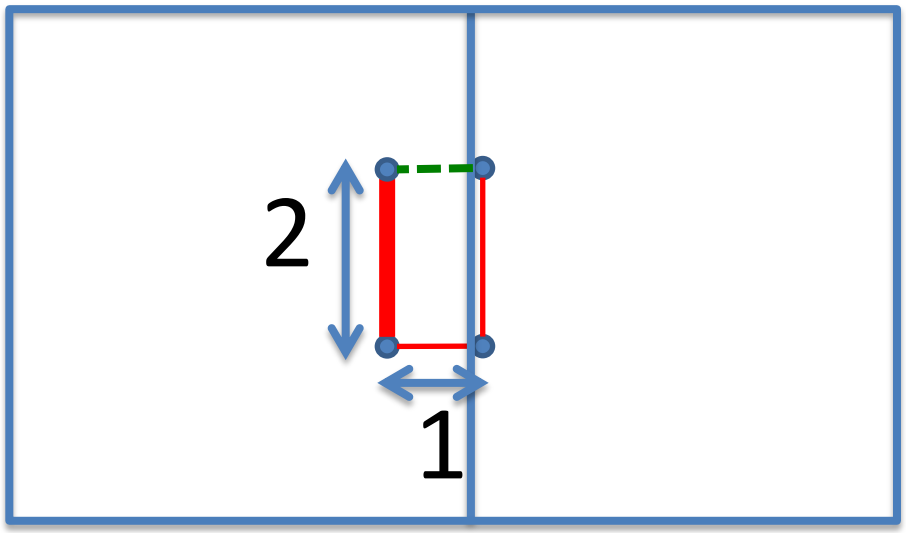
$(1+\epsilon)$-MST in $\mathbf{R}$ $= O(\log\ n)$ rounds

- **Idea:** Only use short edges inside the cells

Impose a **randomly shifted** quadtree (top cell length $\frac{2\Delta}{\epsilon}$)

Bottom-up: For each node (cell) in the quadtree

- compute optimum Minimum Spanning **Forests** in subcells, **using edges of length $\leq \epsilon L$**
- Use only $\epsilon^2 L$-net from each cell on the next level

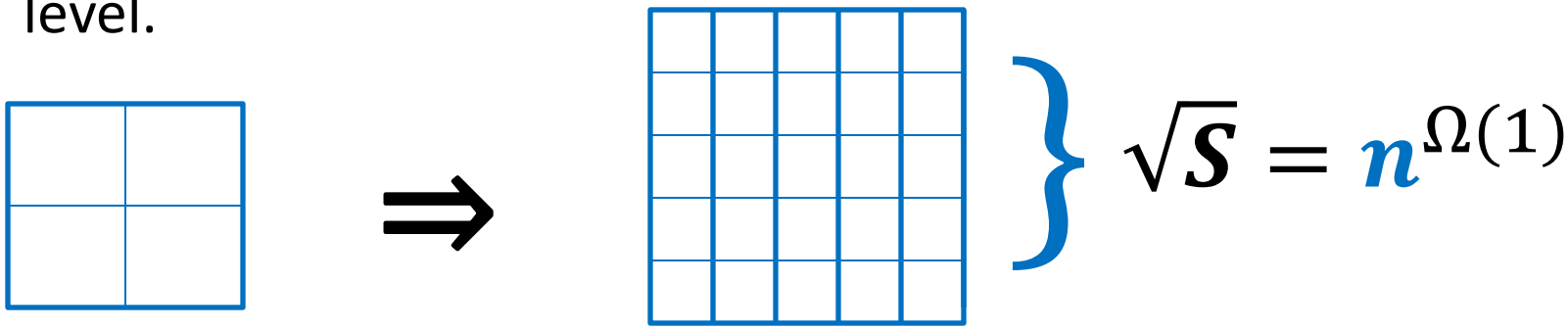


$$L = \Omega\left(\frac{1}{\epsilon}\right)$$

$$\mathbf{Pr[Bad\ Cut]} = O(\epsilon)$$

$(1+\epsilon)$-MST in $\mathbf{R} = O(1)$ rounds

- $O(\log n)$ rounds => O($\log_S n$) = O(1) rounds
  - Flatten the tree: ($\sqrt{S} \times \sqrt{S}$)-grids instead of (2x2) grids at each level.



Impose a **randomly shifted** ($\sqrt{S} \times \sqrt{S}$)-tree

Bottom-up: For each node (cell) in the tree

- compute optimum MSTs in subcells via edges of length $\leq \epsilon L$
- Use only $\epsilon^2 L$-net from each cell on the next level

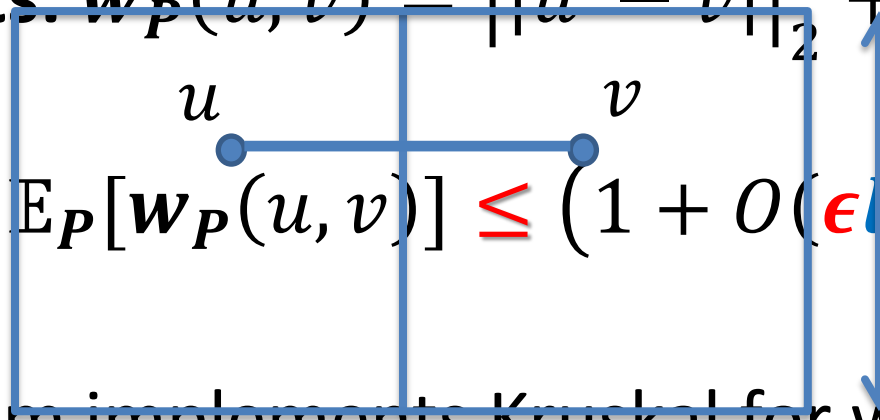
$(1+\epsilon)$-MST in $\mathbf{R} = O(1)$ rounds

Theorem: Let $l = \#$ levels in a random tree P

$$\mathbb{E}_P[\mathbf{ALG}] \leq \big(1 + O(\epsilon l d)\big)\mathbf{OPT}$$

Proof (sketch):

- $\Delta_P(u,v)$ = cell length, which first partitions (u,v)
- **New weights:** $w_P(u,v) = ||u-v||_2 + \epsilon\Delta_P(u,v)$



$$||u-v||_2 \leq \mathbb{E}_P[w_P(u,v)] \leq \big(1 + O(\epsilon l d)\big)||u-v||_2$$

- Our algorithm implements Kruskal for weights w_P

"Solve-And-Sketch" Framework

$(1+\epsilon)$-**MST**:

- "**Load balancing**": partition the tree into parts of the same size
- **Almost linear time**: Approximate Nearest Neighbor data structure [Indyk'99]
- Dependence on dimension **d** (size of ϵ-net is $O\left(\frac{d}{\epsilon}\right)^{d}$)
- Generalizes to bounded **doubling dimension**
- Basic version is teachable (Jelani Nelson's ``**Big Data**'' class at **Harvard**)

“Solve-And-Sketch” Framework

$(1+\epsilon)$-**Earth-Mover Distance, Transportation Cost**

- No simple “divide-and-conquer” Arora-Mitchell-style algorithm (unlike for general matching)
- Only recently sequential $(1+\epsilon)$-apprxoimation in $O_\epsilon\left(\boldsymbol{n}\log^{O(1)}\boldsymbol{n}\right)$time [Sharathkumar, Agarwal ‘12]

Our approach (convex sketching):

- Switch to the flow-based version
- In every cell, send the flow to the closest net-point until we can connect the net points

“Solve-And-Sketch” Framework

Convex sketching the cost function for $\boldsymbol{\tau}$ net points

- $F: \mathbb{R}^{\boldsymbol{\tau}-1} \rightarrow \mathbb{R}$ = the cost of routing fixed amounts of flow through the net points
- Function $F' = F\ +$ “normalization” is monotone, convex and Lipschitz, $(1+\boldsymbol{\epsilon})$-approximates F
- We can $(1+\boldsymbol{\epsilon})$-sketch it using a lower convex hull

Thank you! http://grigory.us

Open problems

- Exetension to high dimensions?
  - Probably no, reduce from connectivity => conditional lower bound : $\Omega(\log n)$ rounds for MST in ℓ_∞^n
  - The difficult setting is $d = \Omega(\log \boldsymbol{n})$ (can do JL)
- Streaming alg for **EMD** and **Transporation Cost**?
- Our work: first near-linear time algorithm for **Transportation Cost**
  - Is it possible to reconstruct the solution itself?