Massively Parallel Algorithms and Hardness of Single-Linkage Clustering

Joint work with Adithya Vadapalli (Indiana University)

Grigory Yaroslavtsev

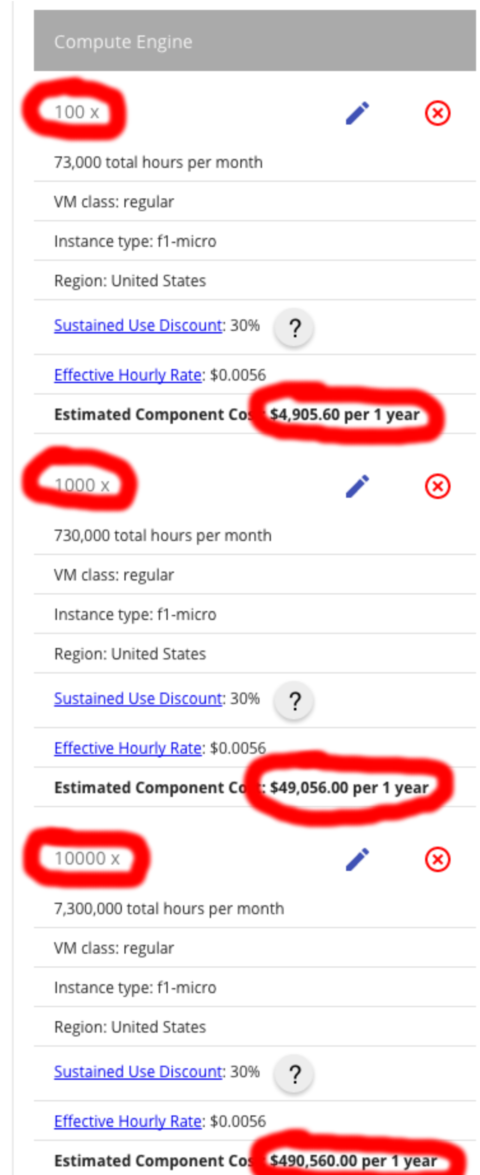
http://grigory.us

Algorithms for Big Data

- **User's perspective**: paradigm shift brought by cloud services
  - Outsourcing computation and data storage is great for both businesses and **researchers**
  - **Cloud service providers**: Amazon EC2, Google Compute Engine, Microsoft Azure, …
  - **Open source stacks/frameworks**: MapReduce/Hadoop, Apache Spark, etc.

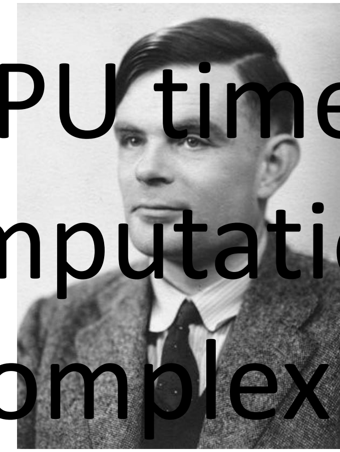
Business perspective

- Pricings:
  - https://cloud.google.com/pricing/
  - https://aws.amazon.com/pricing/
- ~Linear with **space** and **time** usage
  - 100 machines: 5K $/year
  - 10000 machines: 0.5M $/year
- You pay **a lot more** for using provided algorithms
  - https://aws.amazon.com/machine-learning/pricing/



“Big Data Theory” = Turing meets Shannon

=



CPU time / Computational Complexity

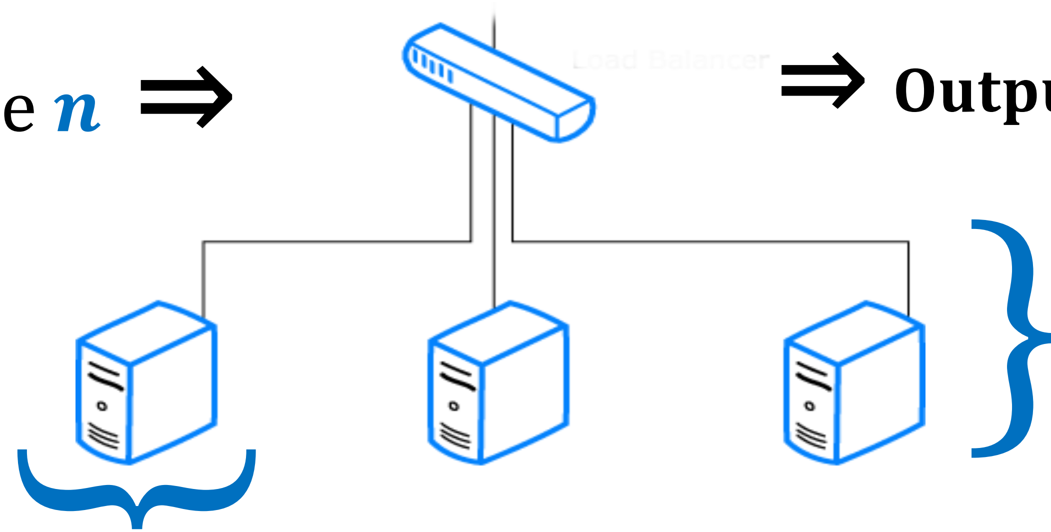
+

Network Time / Information and Communication Complexity

Computational Model

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

Input: size $\boldsymbol{n}$ $\Rightarrow$



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

$\boldsymbol{M}$ machines

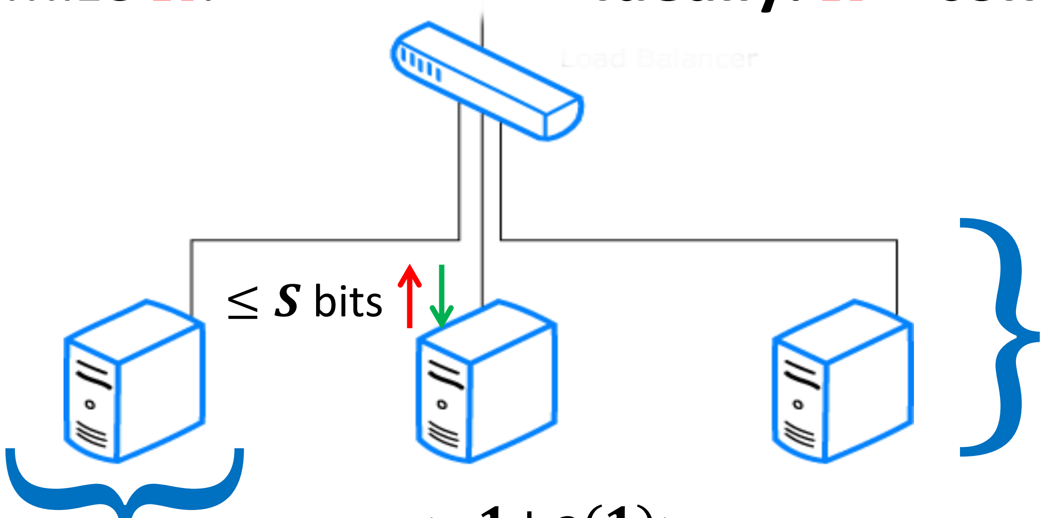
S space

Computational Model

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

Goal: Minimize $\boldsymbol{R}$.

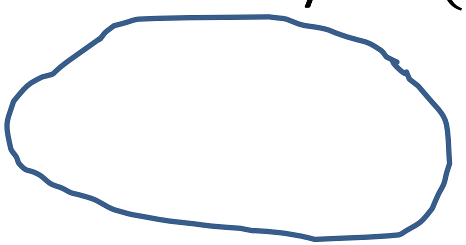
Ideally: $\boldsymbol{R}$ = constant.



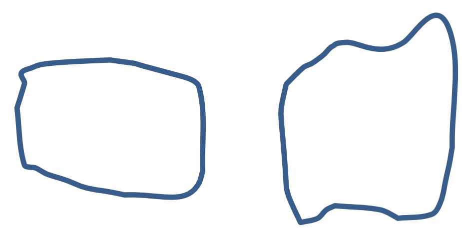
Algorithms for Graphs

- **Dense graphs** vs. sparse graphs
  - **Dense**: $\boldsymbol{S} \gg |V|$
    - Linear sketching: one round
    - "Filtering" [Karloff, Suri Vassilvitskii, SODA'10; Ene, Im, Moseley, KDD'11; Lattanzi, Moseley, Suri, Vassilvitskii, SPAA'11; Suri, Vassilvitskii, WWW'11] … [Bateni, Behnezhad, Derakshan, Hajiaghayi, Kiveris, Lattanzi, Mirrokni, NIPS'17]
  - Sparse: $\boldsymbol{S} \ll |V|$ (or $\boldsymbol{S} \ll$ solution size)

    Sparse graph problems appear hard (**Big open question**: connectivity in $\mathrm{o}(\log n)$ rounds?)



vs.



Two Conjectures

- **Conj 1: (Connectivity)** Given graph with $O(|V|)$ edges finding connected components requires Ω(log |V|) rounds
- **Conj 2: (Two cycles)** Distinguishing one cycles from two cycles requires Ω(log |V|) rounds
- [Roughgarden, Vassilvitskii, Wang SPAA‘16]
  - Give an $\Omega(\log_s n)$ lower bound
  - Some evidence of relationship to $NC^1 \subseteq P$

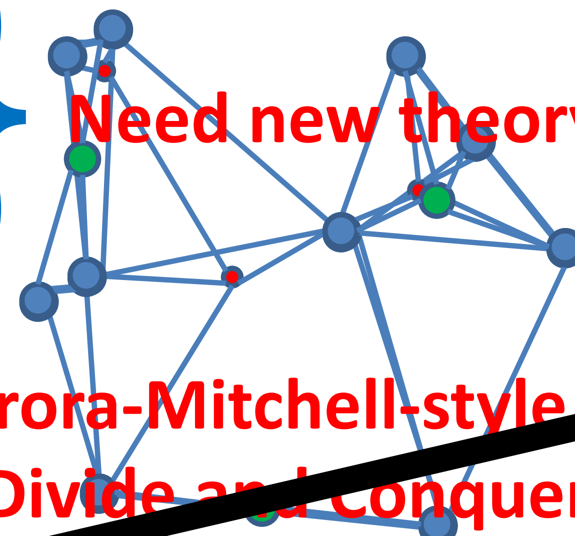
Geometric Graph Problems [ANOY'14]

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.)

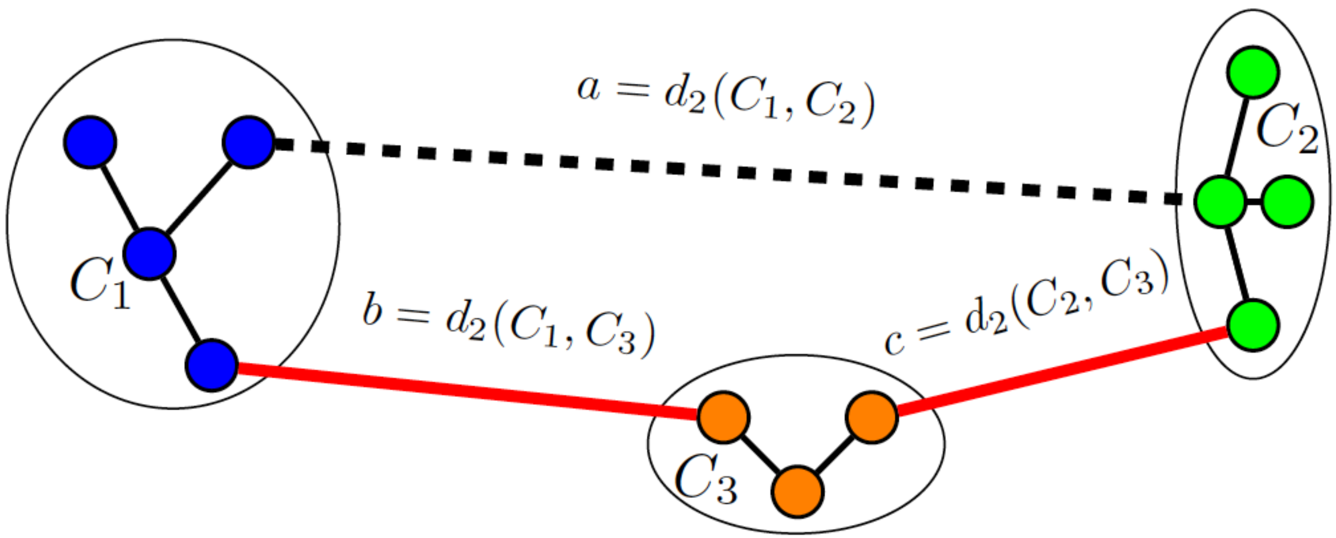
Arora-Mitchell-style "Divide and Conquer", easy to implement in Massively Parallel Computational Models, but bad running time

MST: Single Linkage Clustering

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

k clusters: remove $k-1$ longest edges from MST

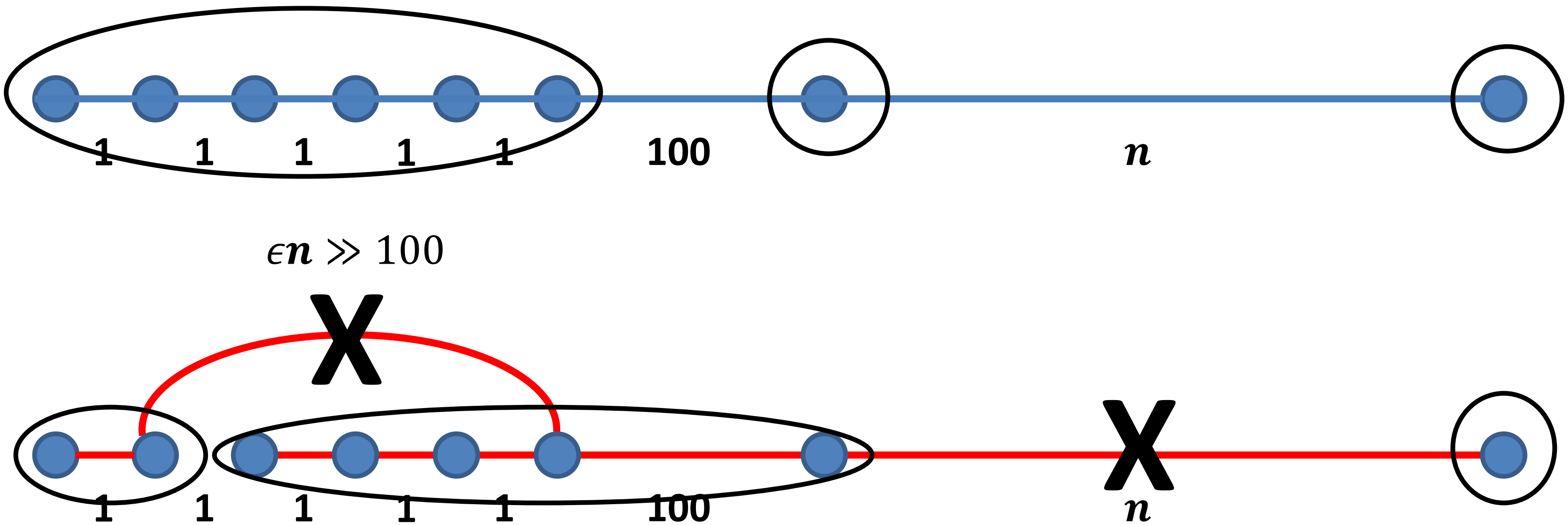
- Maximizes **minimum** intercluster distance



Objective: min(a,b,c)

MST vs. Single Linkage Clustering

- [ANOY'14]: $(1+\epsilon)$-approx MST (in expectation)
- Single-linkage clustering can be arbitrarily bad:



Our Results

	Approximation in $O(\log n)$ rounds	Hardness of approx. in $o(\log n)$ rounds
ℓ_0	Exact for $d = O(1)$	2 for $d = 2$ under Connectivity 3 for $d = \Omega(n)$ under Two Cycles
ℓ_1	$(1+\epsilon)$ for $d = O(1)$	2 for $d = \Omega(n)$ under Connectivity 3 for $d = \Omega(n)$ under Two Cycles
ℓ_2	$(1+\epsilon)$ for $d = O(1)$	$1.41 - \epsilon$ for $d = \Omega\left(\frac{\log n}{\epsilon^2}\right)$ under Connectivity $1.84 - \epsilon$ for $d = \Omega\left(\frac{\log n}{\epsilon^2}\right)$ under Two Cycles
ℓ_∞	$(1+\epsilon)$ for $d = O(1)$	2 for $d = \Omega(n)$ under Connectivity [ANOY'14]

ℓ_0 and ℓ_1 hardness results hold for O(1)-sparse vectors, i.e. for inputs of size O(n)

Hardness for ℓ_2

Reduction from "One vs. Two Cycles"

- Vector v_i for each vertex, set $\boldsymbol{v}_i = \boldsymbol{e}_i$
- For each edge (i, j) update (for $\boldsymbol{\xi} = 1/\sqrt{2}$):
  - $\boldsymbol{v}_i = \boldsymbol{v}_i + \boldsymbol{\xi}\boldsymbol{e}_j$
  - $\boldsymbol{v}_j = \boldsymbol{v}_j + \boldsymbol{\xi}\boldsymbol{e}_i$
- Apply Johnson-Lindenstrauss transform to reduce the dimension down to $\boldsymbol{d} = O\left(\frac{\log n}{\epsilon^2}\right)$

Important that reduction can be done in O(1) rounds

Hardness for ℓ_2

Reduction from "One vs. Two Cycles"

- Vector $\boldsymbol{v}_i$ for each vertex, set $\boldsymbol{v}_i = \boldsymbol{e}_i$
- For each edge (i, j) update (for $\xi = 1/\sqrt{2}$):
  - $\boldsymbol{v}_i = \boldsymbol{v}_i + \xi \boldsymbol{e}_j$
  - $\boldsymbol{v}_i = \boldsymbol{v}_j + \xi \boldsymbol{e}_i$

- $\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2 = \sqrt{2}\left(\sqrt{2 - \sqrt{2}}\right)$ if there is an edge (i, j)
- $\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2 = 2$ if there is no edge (i, j)

Ratio between these cases gives hardness of $\sqrt{2 + \sqrt{2}}$

General algorithm for $\ell_1, \ell_2, \ell_\infty$

- Input: vectors $v_1, \ldots, v_{\boldsymbol{n}} \in \mathbb{R}^{\boldsymbol{d}}$
- $E = \emptyset$
- Repeat $O(\log \boldsymbol{n})$ times sequentially:
  - E' = set of edges of a $(1 + \epsilon)$-approximate MST
  - $E = E \cup E'$
- Run Boruvka's algorithm on E
- Drop $k - 1$ longest edges to get the clustering

Large geometric graphs

- Graph algorithms: **Dense graphs** vs. sparse graphs
  - **Dense**: $\boldsymbol{S} \gg |V|$.
  - Sparse: $\boldsymbol{S} \ll |V|$.

- <u>**Our setting:**</u>
  - Dense graphs, sparsely represented: $\mathrm{O}(\mathbf{n})$ space
  - Output doesn't fit on one machine ($\boldsymbol{S} \ll \boldsymbol{n}$)
- **Today:** $(1+\epsilon)$-approximate MST [ANOY'14]
  - $\boldsymbol{d} = 2$ (easy to generalize)
  - $\boldsymbol{R} = \log_{\boldsymbol{S}} \boldsymbol{n} = \mathrm{O}(1)$ rounds ($\boldsymbol{S} = \boldsymbol{n}^{\boldsymbol{\Omega(1)}}$)

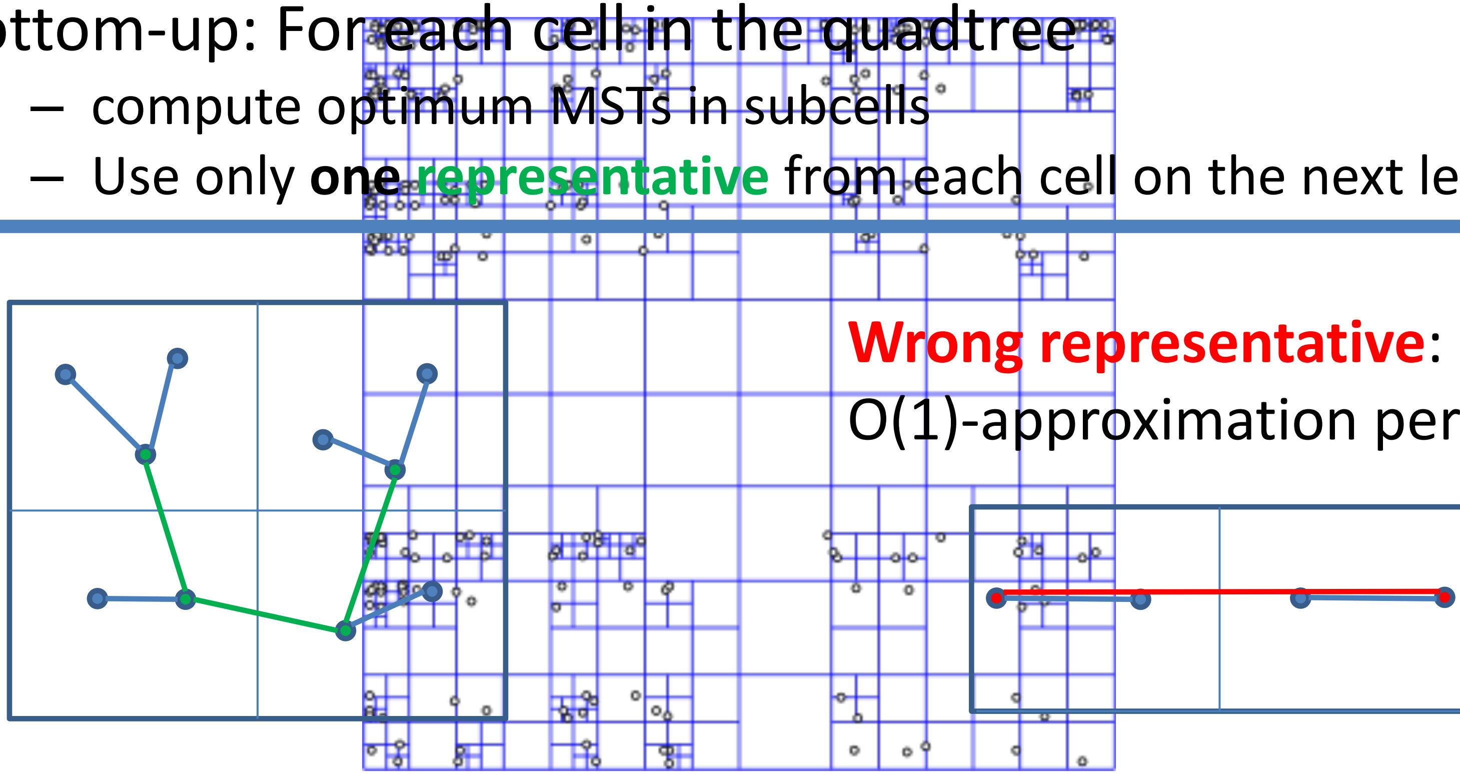
$O(\log n)$-MST in $R = O(\log n)$ rounds

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

Impose an $O(\log \boldsymbol{n})$-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:

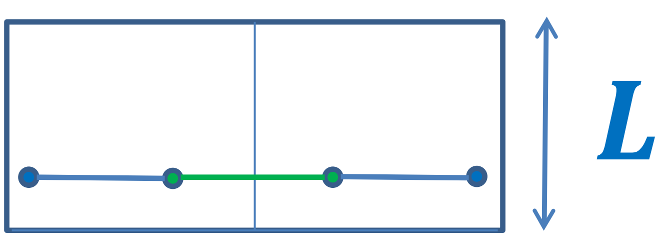
  Collection **S** of vertices in C, every vertex is at distance <= ϵL from some vertex in **S**. (Fact: Can efficiently compute ϵ-net of size $O\left(\frac{1}{\epsilon^2}\right)$)

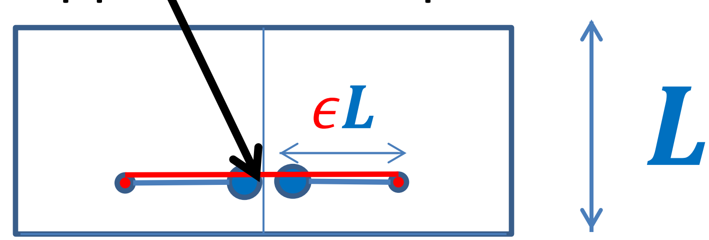
Bottom-up: For each cell in the quadtree
– Compute optimum MSTs in subcells
– Use ϵL-net from each cell on the next level

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

Wrong representative

O(1)-approximation per level





Randomly shifted quadtree

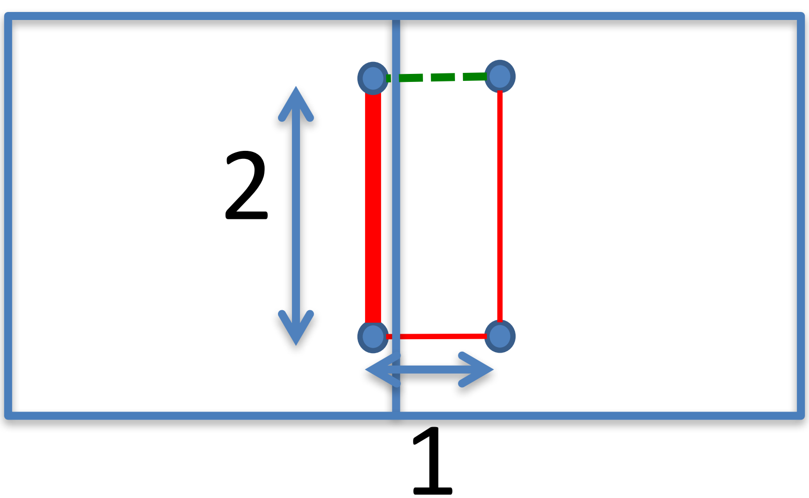
- Top cell shifted by a random vector in $[0, \boldsymbol{L}]^2$

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

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] = $\boldsymbol{\Omega(1)}$

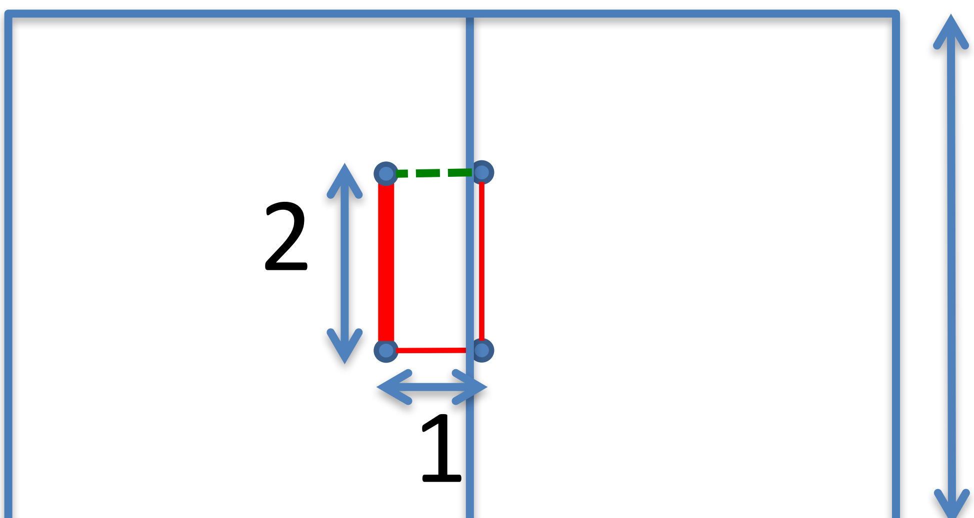
$(1+\epsilon)$-MST in $\mathbf{R} = O(\log\ n)$ rounds

- **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

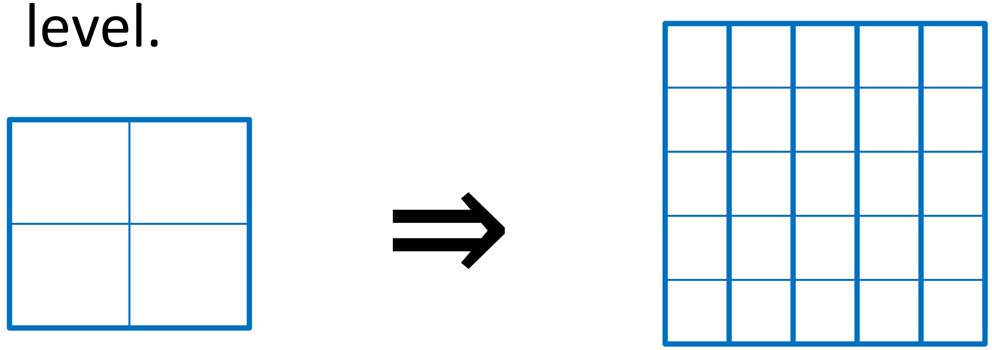


$$L = \Omega\left(\frac{1}{\epsilon}\right)$$

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: $(\sqrt{M}\times\sqrt{M})$-grids instead of (2x2) grids at each level.



$\sqrt{M} = n^{\Omega(1)}$

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

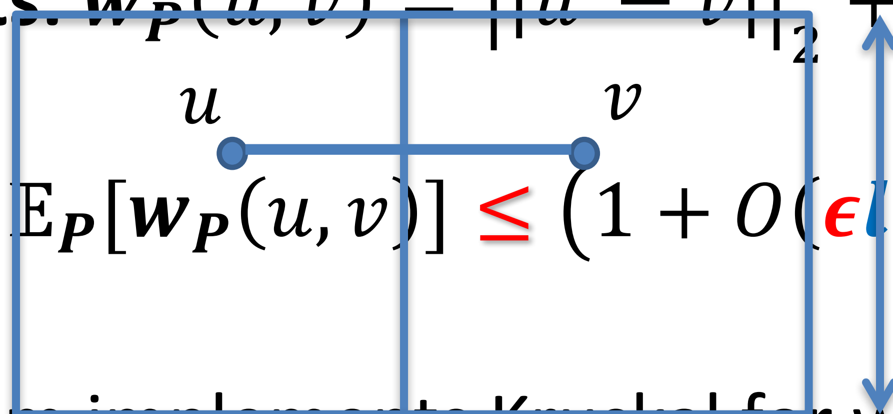
$(1+\epsilon)$-MST in $\mathbf{R} = O(1)$ rounds

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

$$\mathbb{E}_{\boldsymbol{P}}[\mathbf{ALG}] \le \left(1 + O(\boldsymbol{\epsilon l d})\right)\mathbf{OPT}$$

Proof (sketch):

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



$$||u - v||_2 \le \mathbb{E}_{\boldsymbol{P}}[\boldsymbol{w_P}(u, v)] \le \left(1 + O(\boldsymbol{\epsilon l d})\right)||u - v||_2$$

- Our algorithm implements Kruskal for weights $\boldsymbol{w_P}$

Thank you!

- Experiments in Apache Spark on largest vector datasets from UCI ML repository
  - $\approx$11M vectors => 960 TB for adjacency matrix
  - SIFT & HIGGS datasets preprocessed with PCA
- More on my blog http://grigory.us/blog/
- CAML: http://caml.indiana.edu