Distance Computation in Massive Graphs

Michal Dory, University of Haifa

Computing Distances

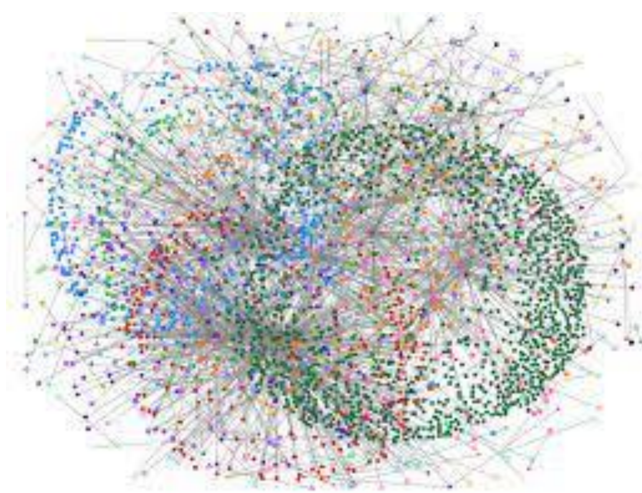
Big Data

How to do a computation when the input is too large and cannot be stored in one machine?

Big Data

How to do a computation when the input is too large and cannot be stored in one machine?

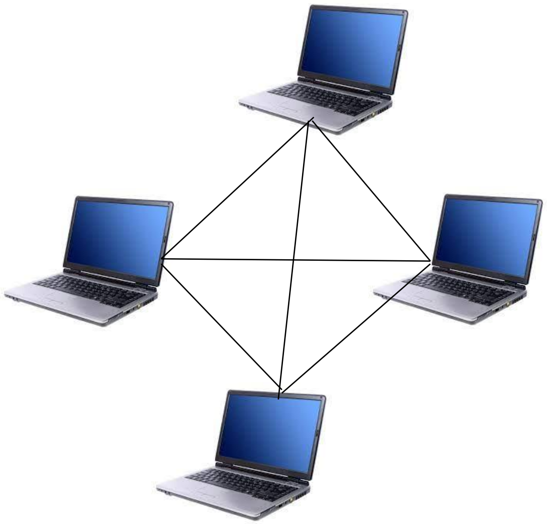
Development of practical systems for massively parallel computation: MapReduce, Hadoop, Spark, Dryad

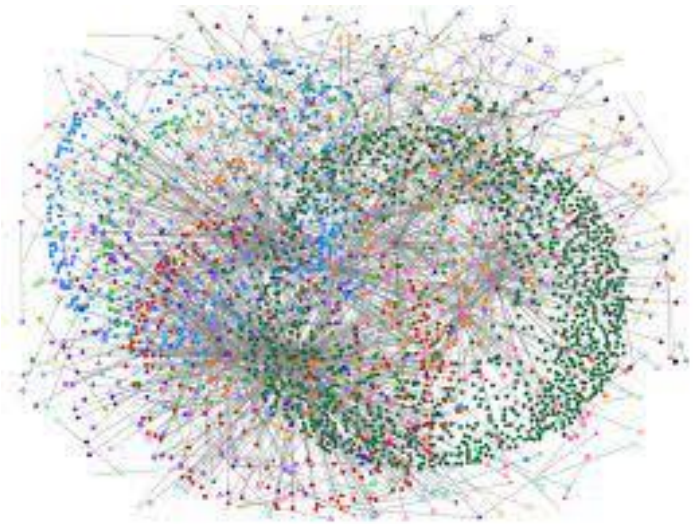


This Talk

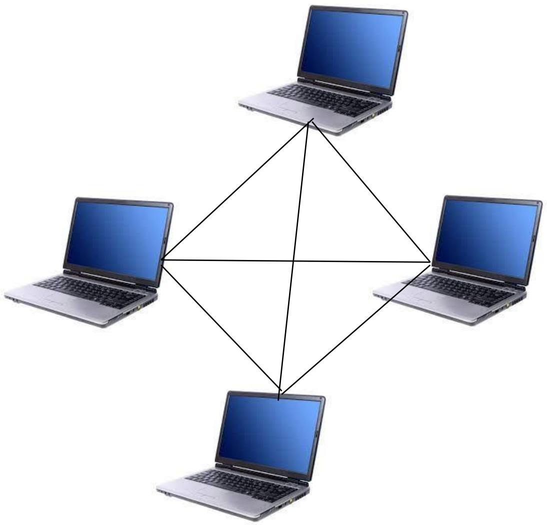
<u>Computing distances:</u>

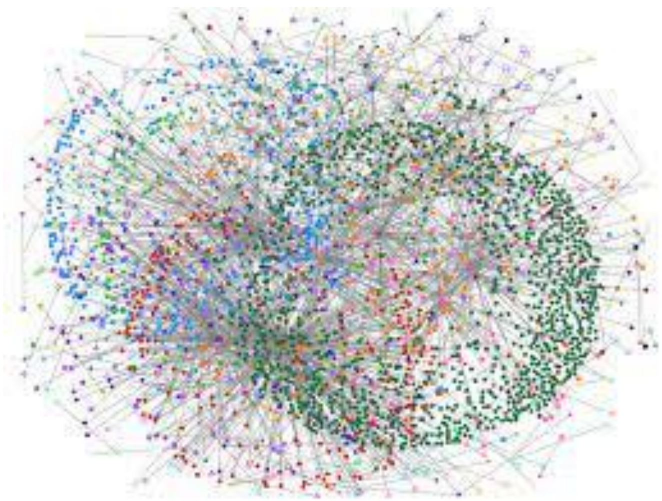
[Censor-Hillel, D, Korhonen, Leitersdorf, 2019]
[D, Parter, 2020]
[D, Fischer, Khoury, Leitersdorf, 2021]
[Biswas, D, Ghaffari, Mitrovic, Nazari, 2021]





The Model

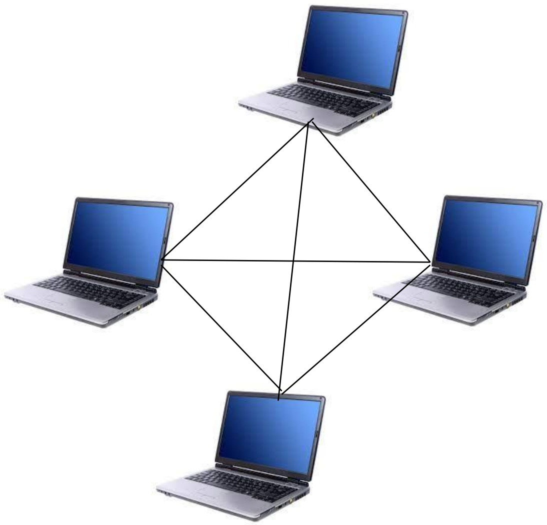


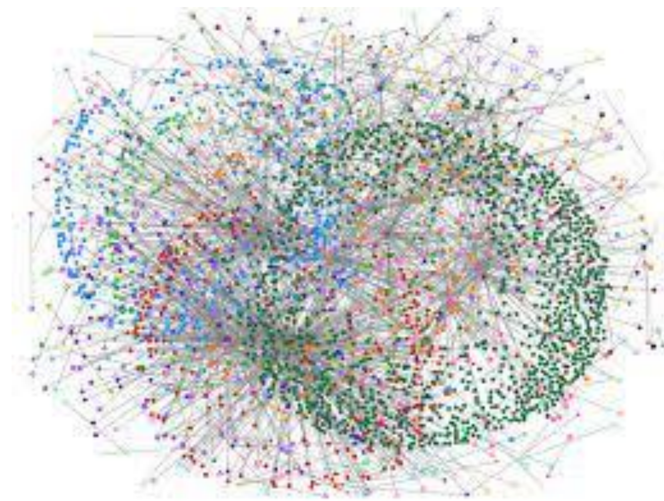


- n machines

n vertices

The Model

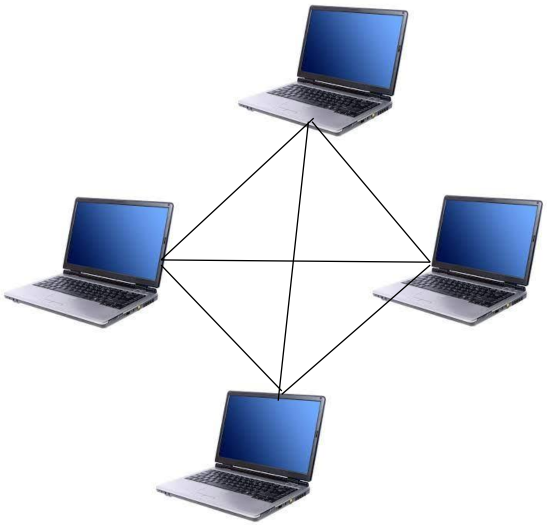


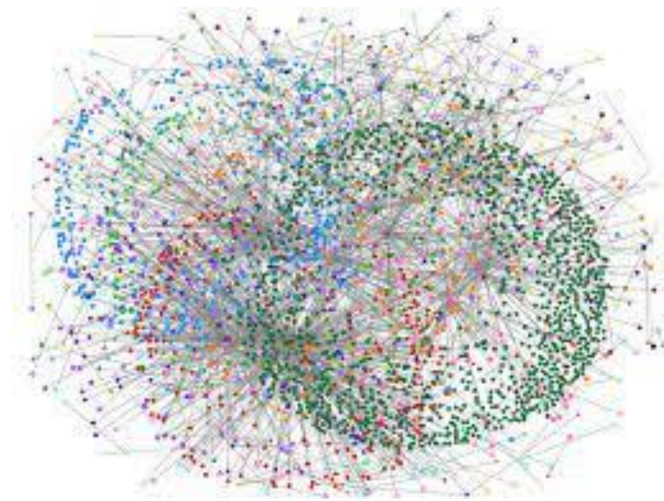


n vertices

- n machines
- Input per machine: $O(n)$ edges

The Model

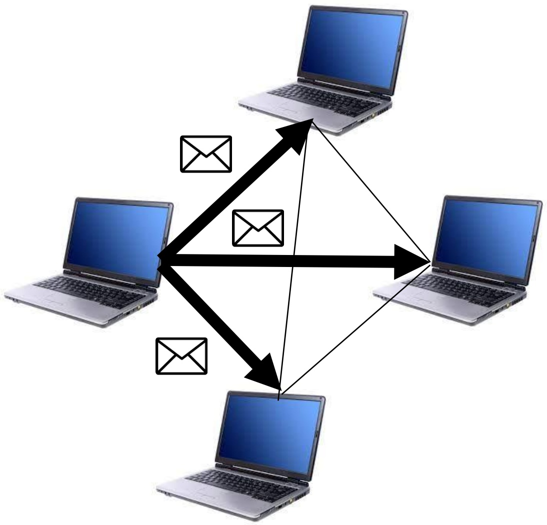


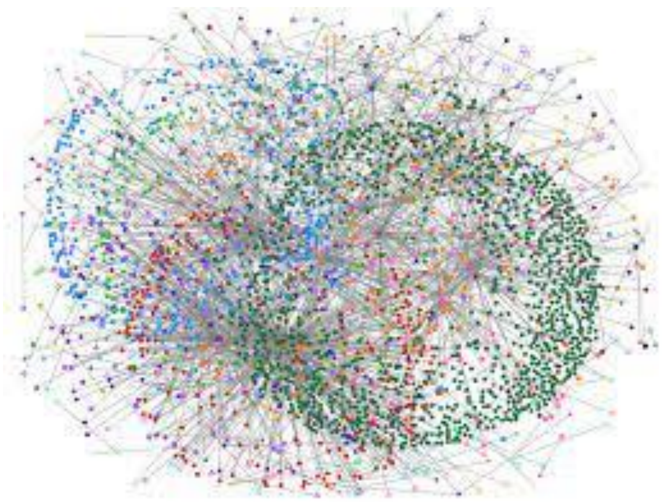


n vertices

- n machines
- Input per machine: $O(n)$ edges
- Each machine can send/receive a total of n messages of $O(\log n)$ bits per round

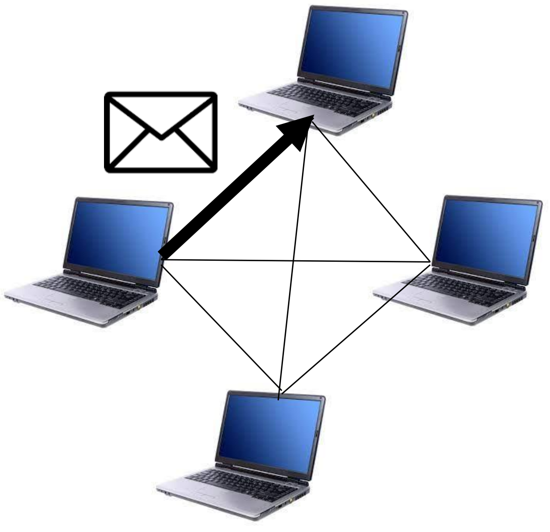
The Model

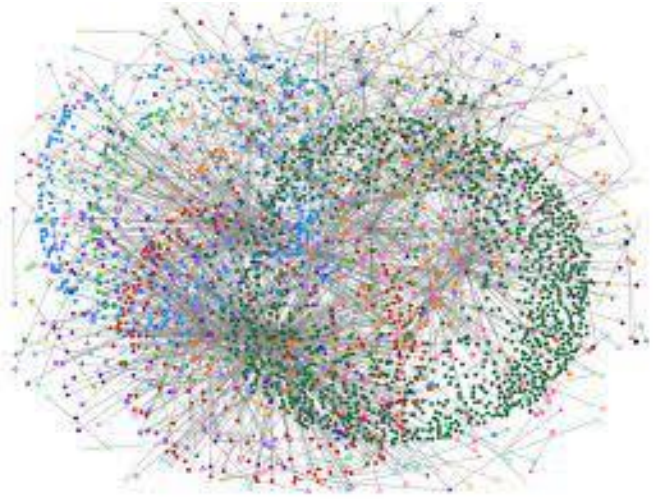




- n machines
- Input per machine: $O(n)$ edges
- Each machine can send/receive a total of n messages of $O(\log n)$ bits per round

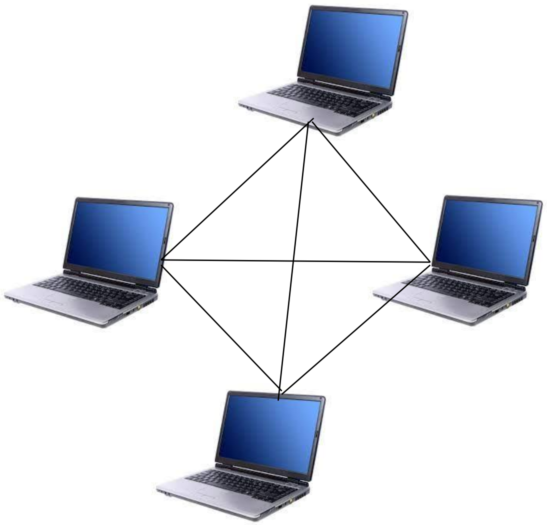
The Model

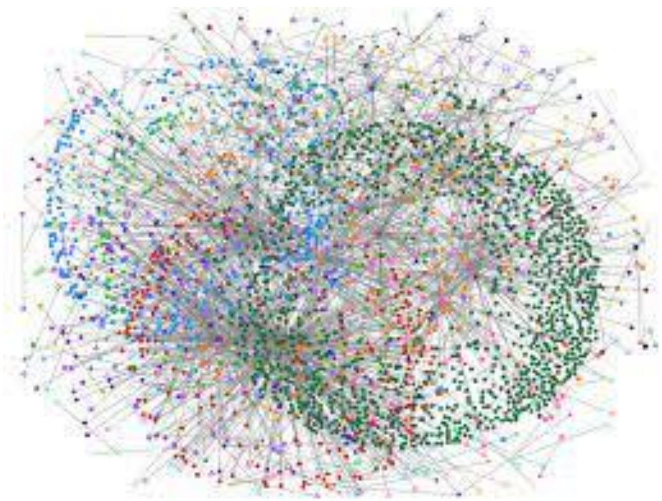




- n machines
- Input per machine: $O(n)$ edges
- Each machine can send/receive a total of n messages of $O(\log n)$ bits per round

The Model - Congested Clique



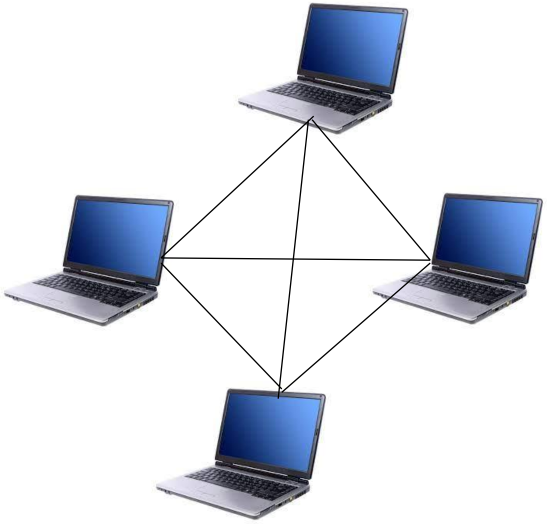


n vertices

- n machines
- Input per machine: $O(n)$ edges
- Each machine can send/receive a total of n messages of $O(\log n)$ bits per round

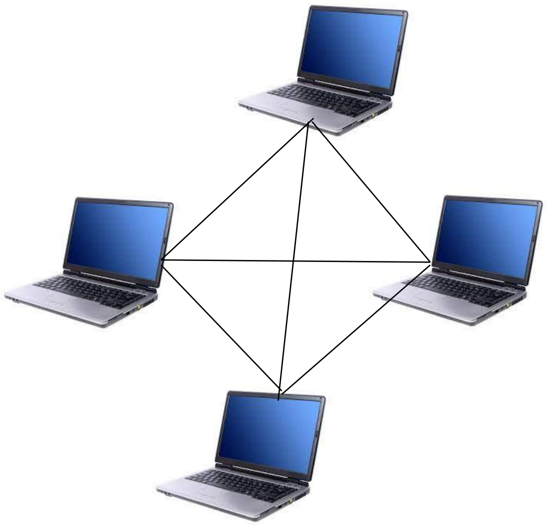
In the AMG workshop (Friday)

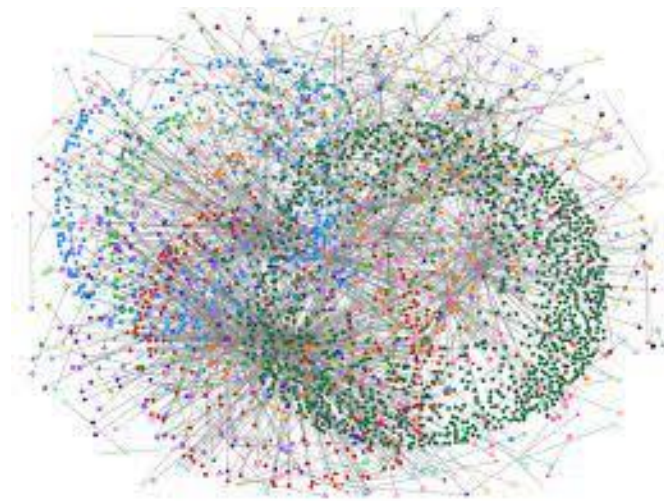
Yasamin Nazari – Distance Computation in MPC and related models



Machines have sublinear memory

The Model - Congested Clique



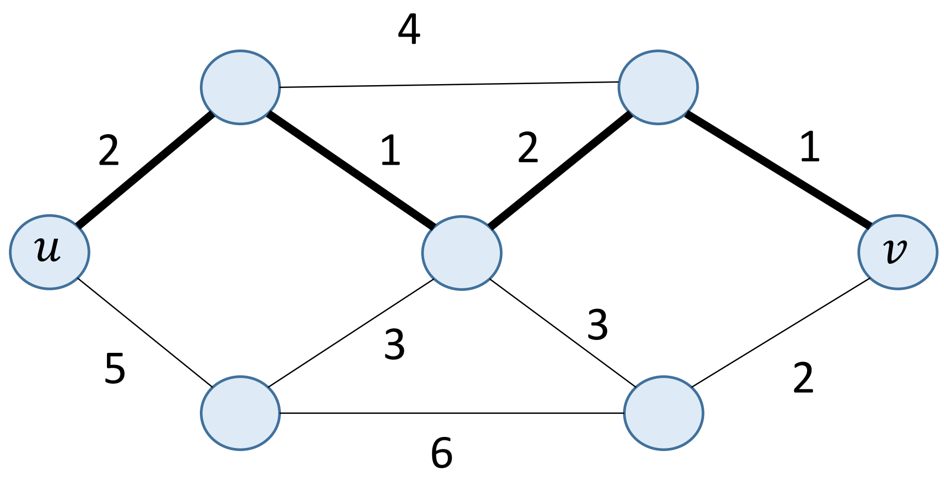


n vertices

- n machines
- Input per machine: $O(n)$ edges
- Each machine can send/receive a total of n messages of $O(\log n)$ bits per round

Distance Computation

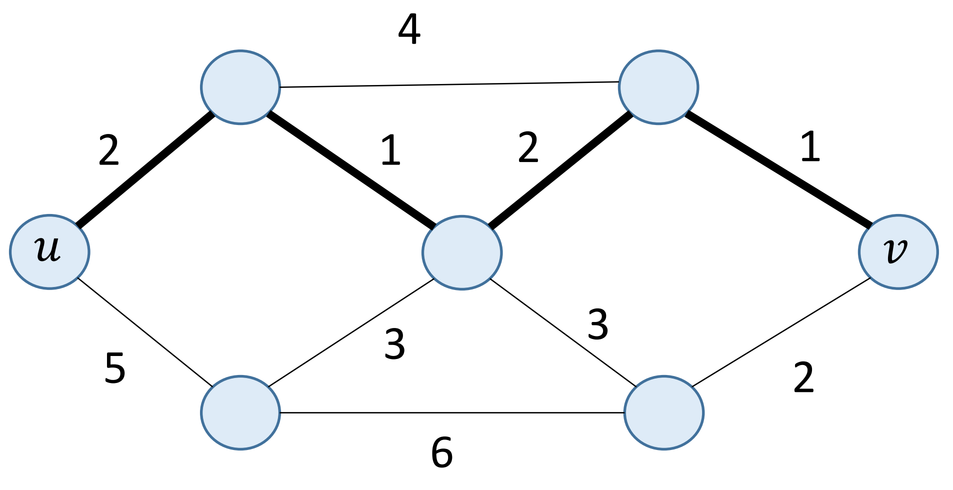
All-pairs shortest paths (APSP)



Distance Computation

All-pairs shortest paths (APSP)

<u>Focus</u>: APSP in unweighted undirected graphs

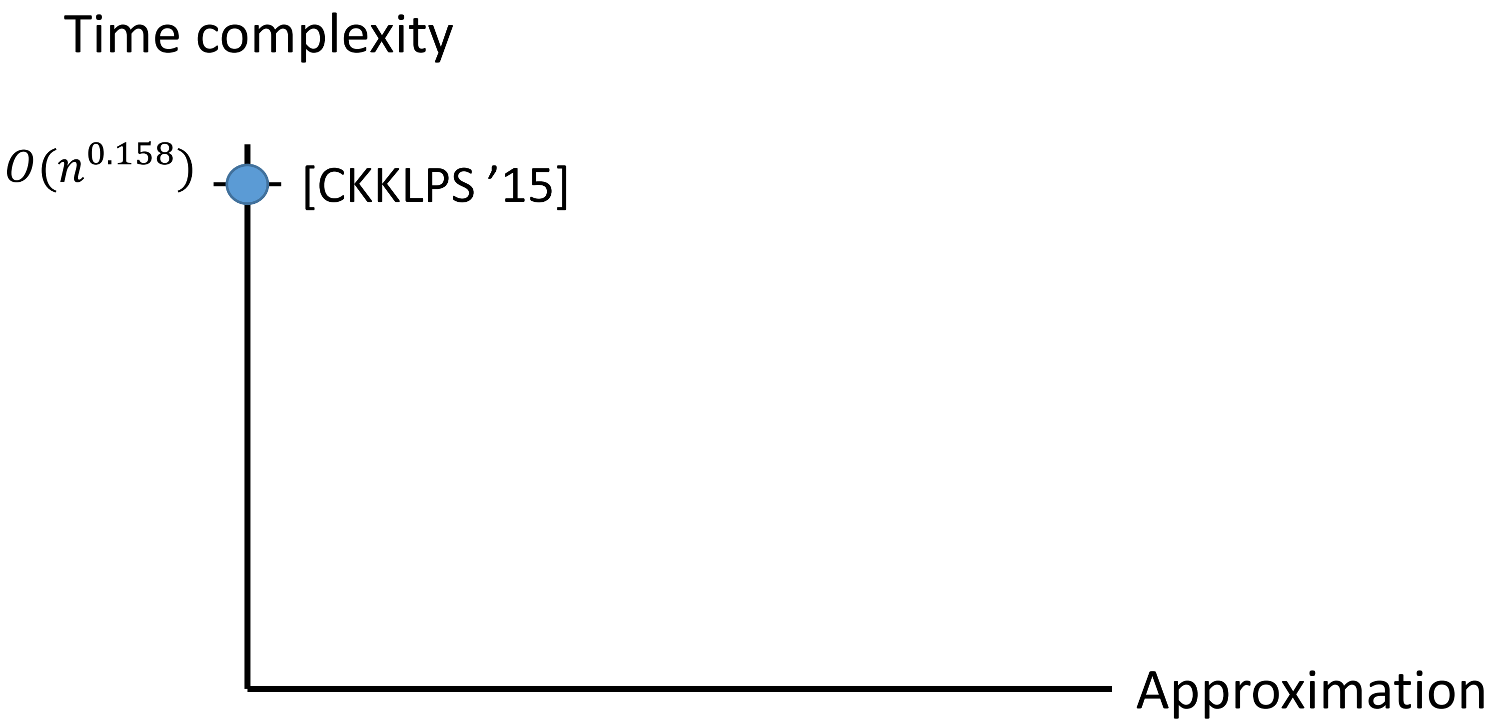


Distributed APSP

- Polynomial time algorithms for exact APSP based on matrix multiplication [Censor-Hillel et al. 15, Le Gall 16]

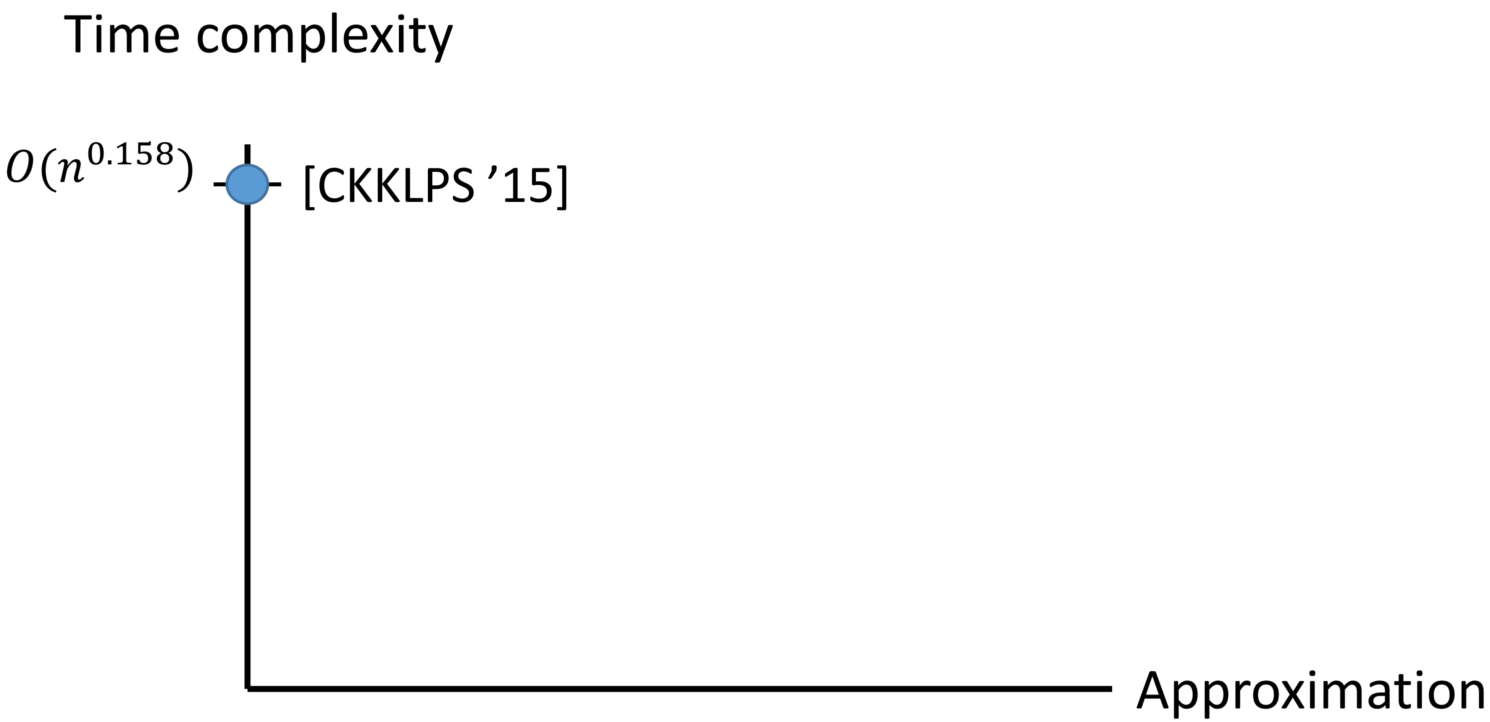
Round Complexity	Variant
$\tilde{O}\left(n^{1/3}\right)$	weighted directed
$O\left(n^{0.158}\right)$	unweighted undirected

Distributed APSP



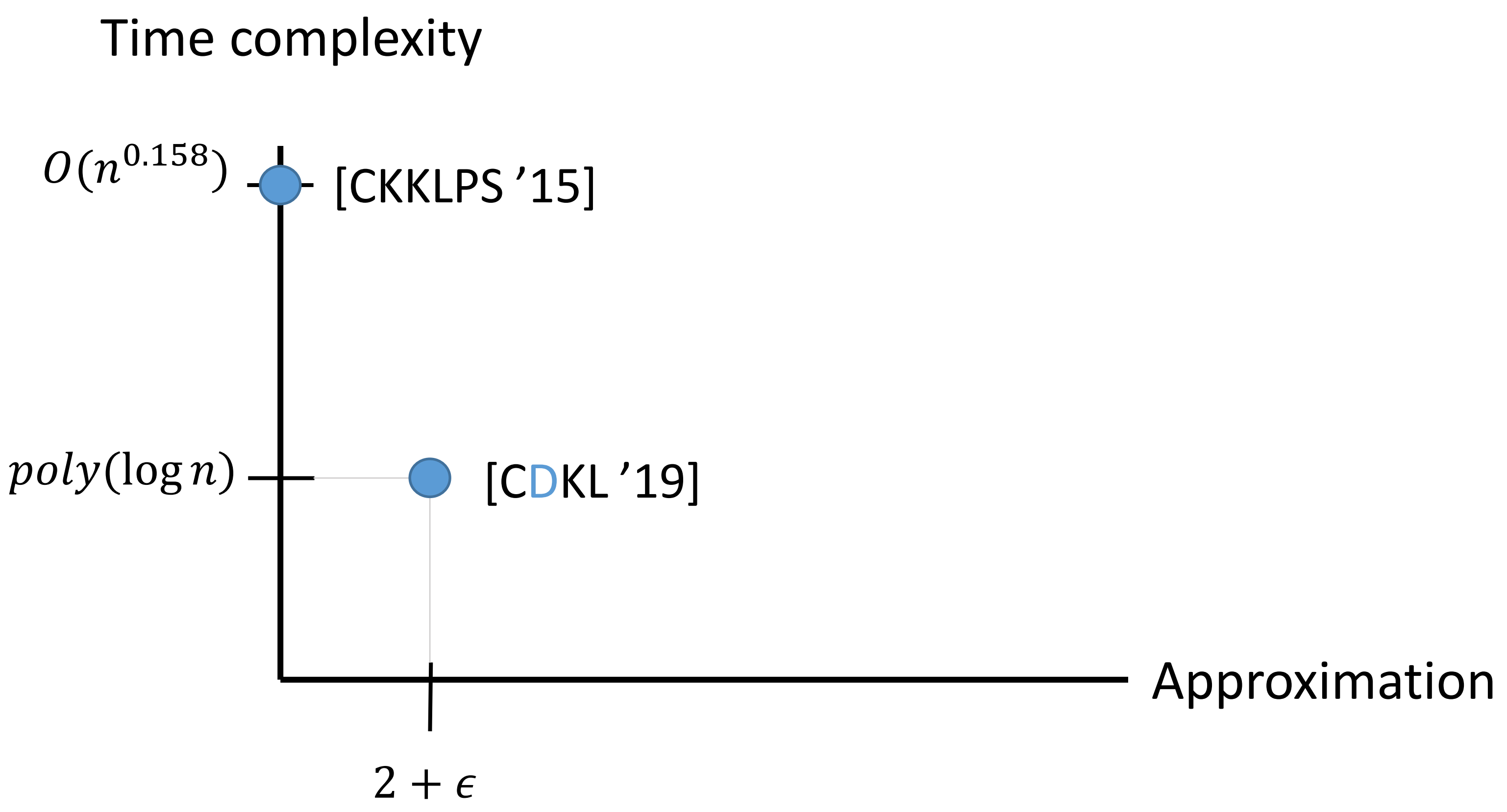
Distributed APSP

Can we get faster algorithms if we allow approximations?



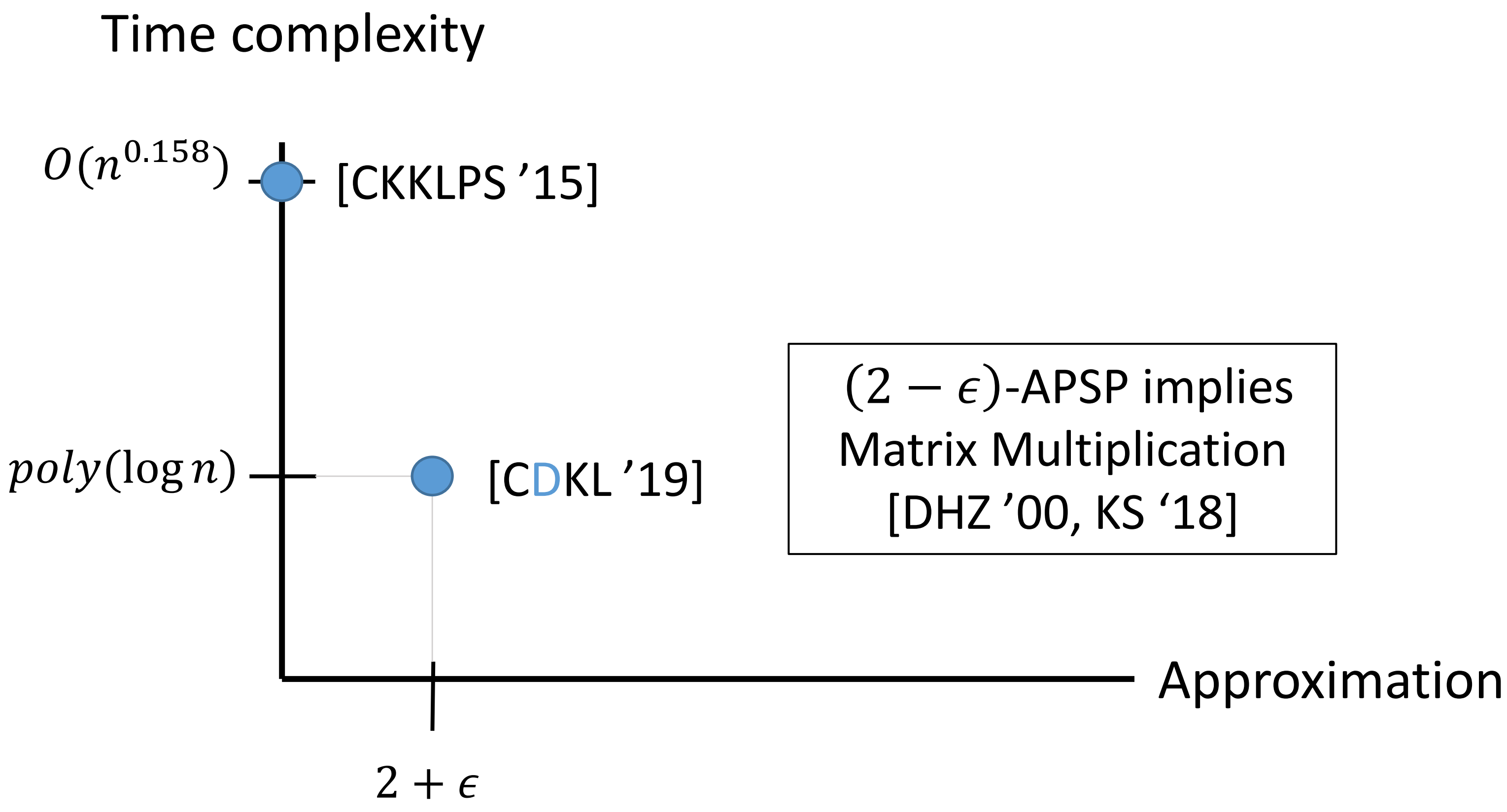
Distributed APSP

Can we get faster algorithms if we allow approximations?



Distributed APSP

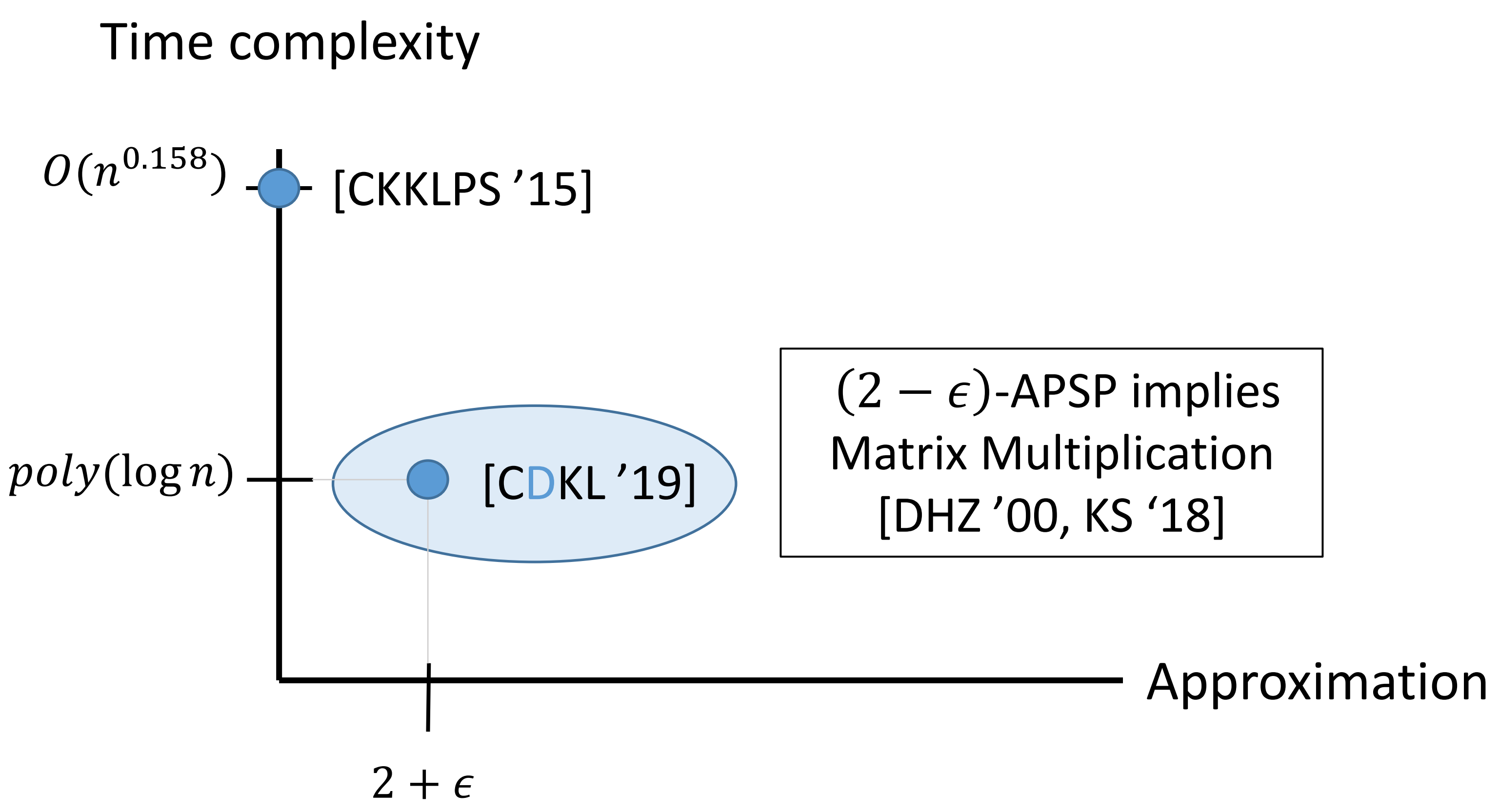
Can we get faster algorithms if we allow approximations?



$(2-\epsilon)$-APSP implies Matrix Multiplication [DHZ '00, KS '18]

Distributed APSP

Can we get faster algorithms if we allow approximations?



Distributed APSP

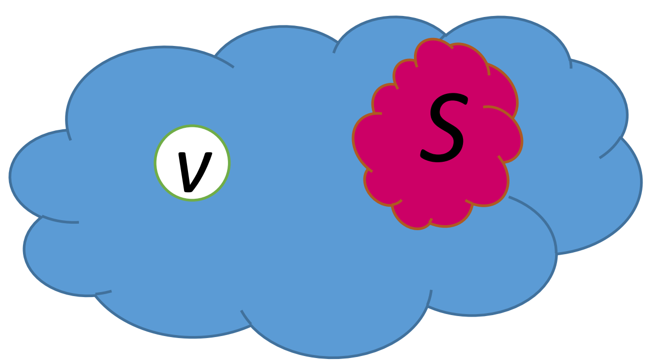
Matrix Multiplication $\Rightarrow$ Exact APSP

Sparse Matrix Multiplication $\Rightarrow$ $(2+\epsilon)$-APSP

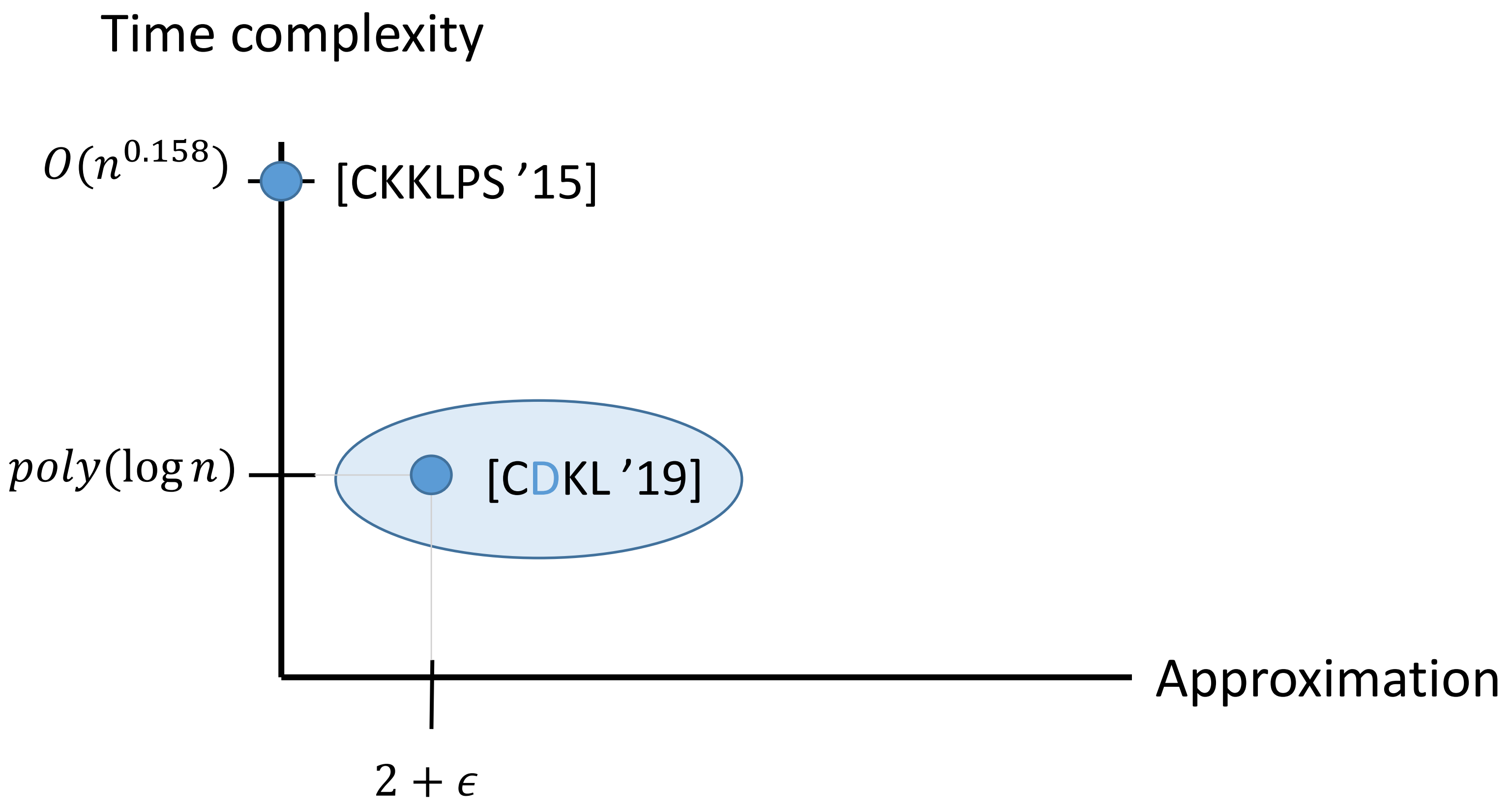
Distributed APSP

Matrix Multiplication → Exact APSP

Sparse Matrix Multiplication → Distance Tools → $(2+\epsilon)$-APSP

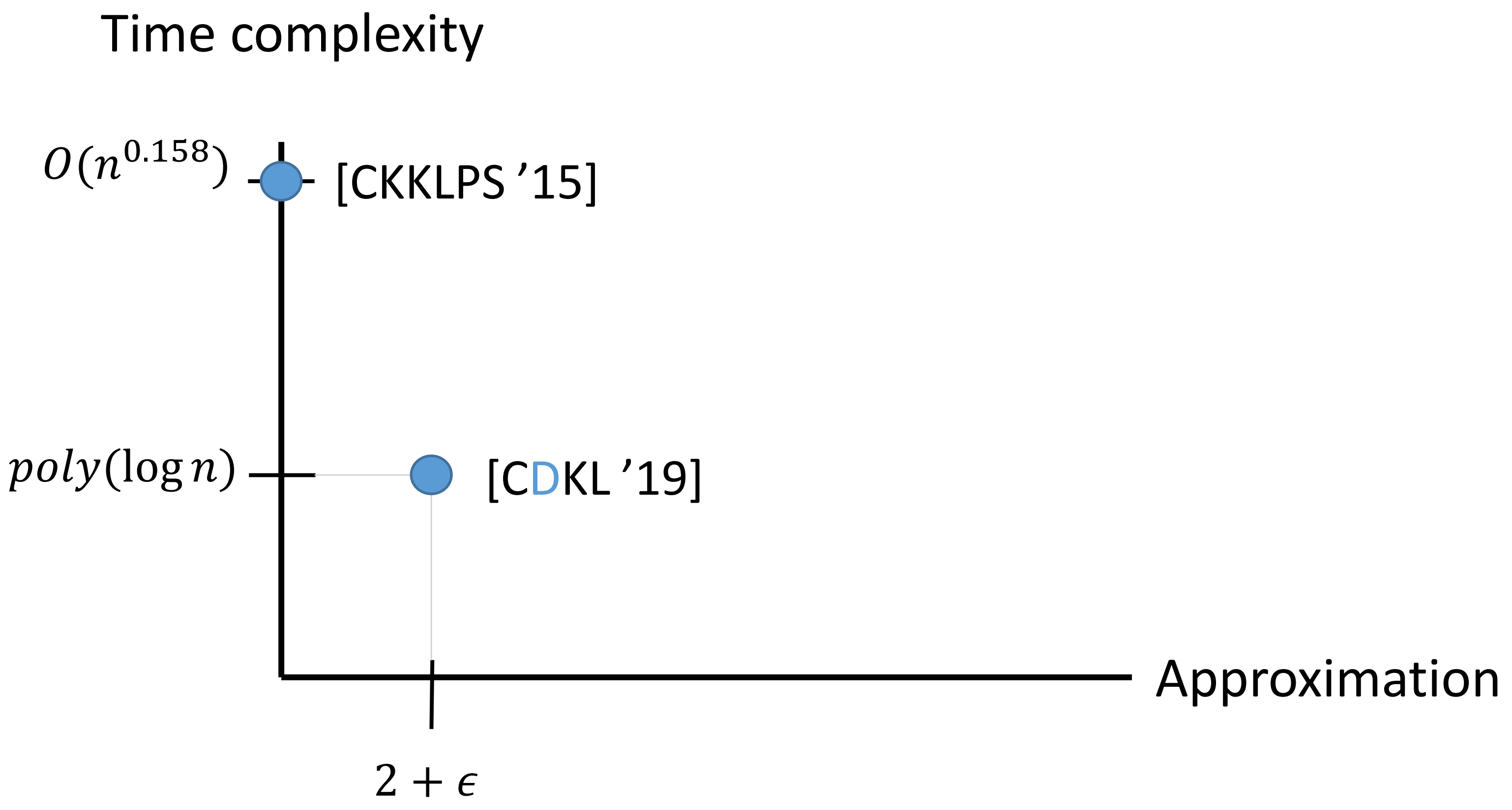


Distributed APSP



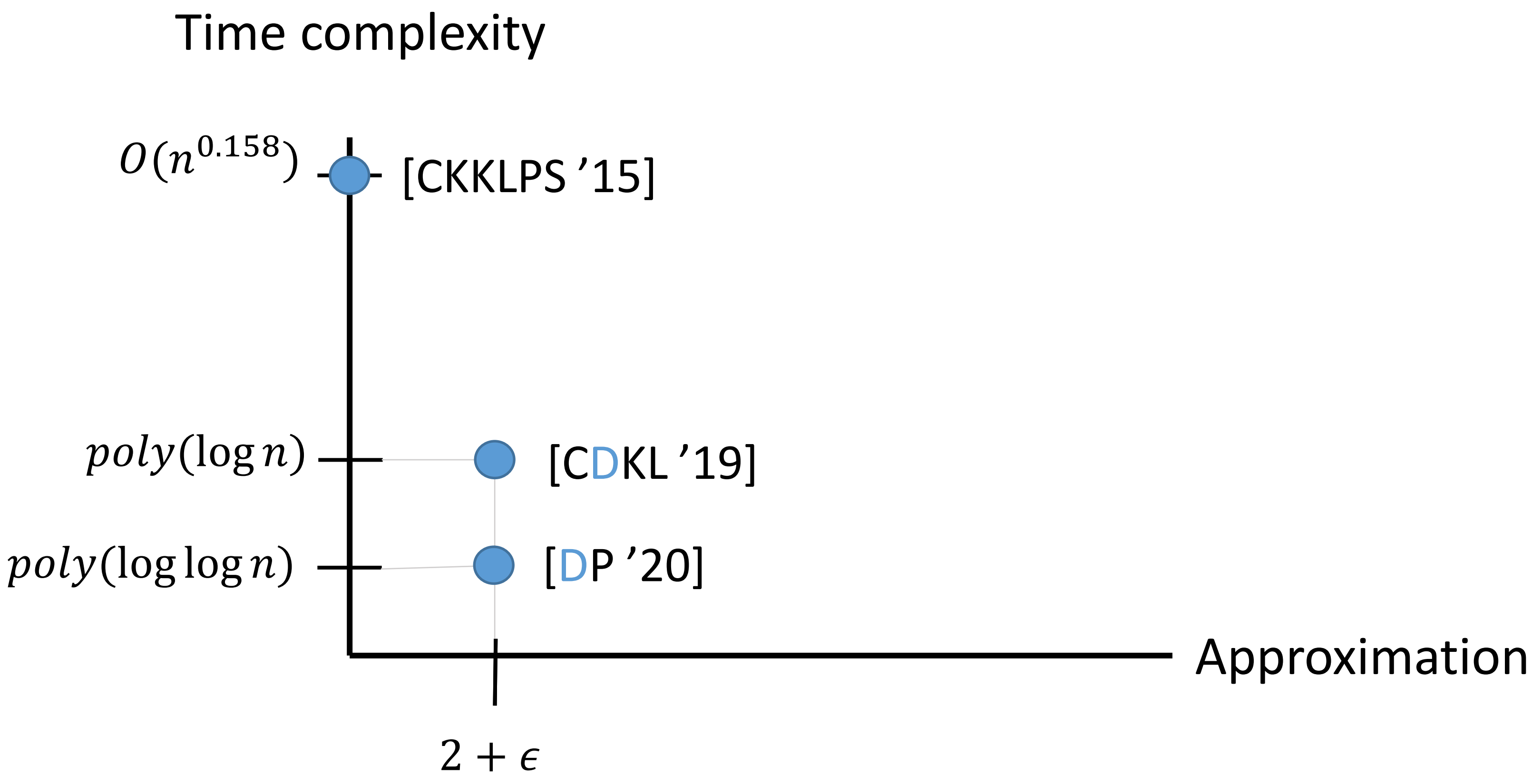
Distributed APSP

Can we get faster algorithms?



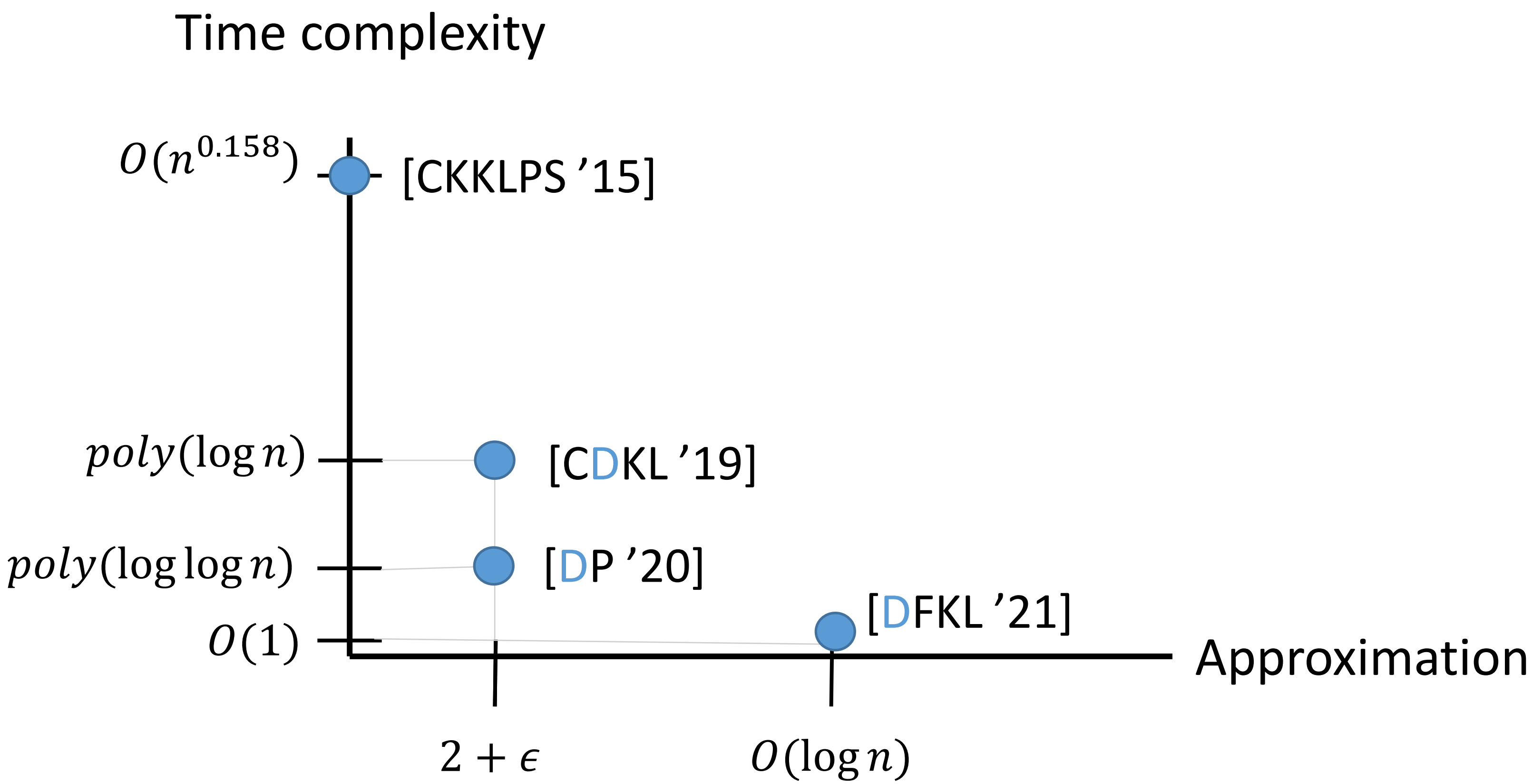
Distributed APSP

Can we get faster algorithms?



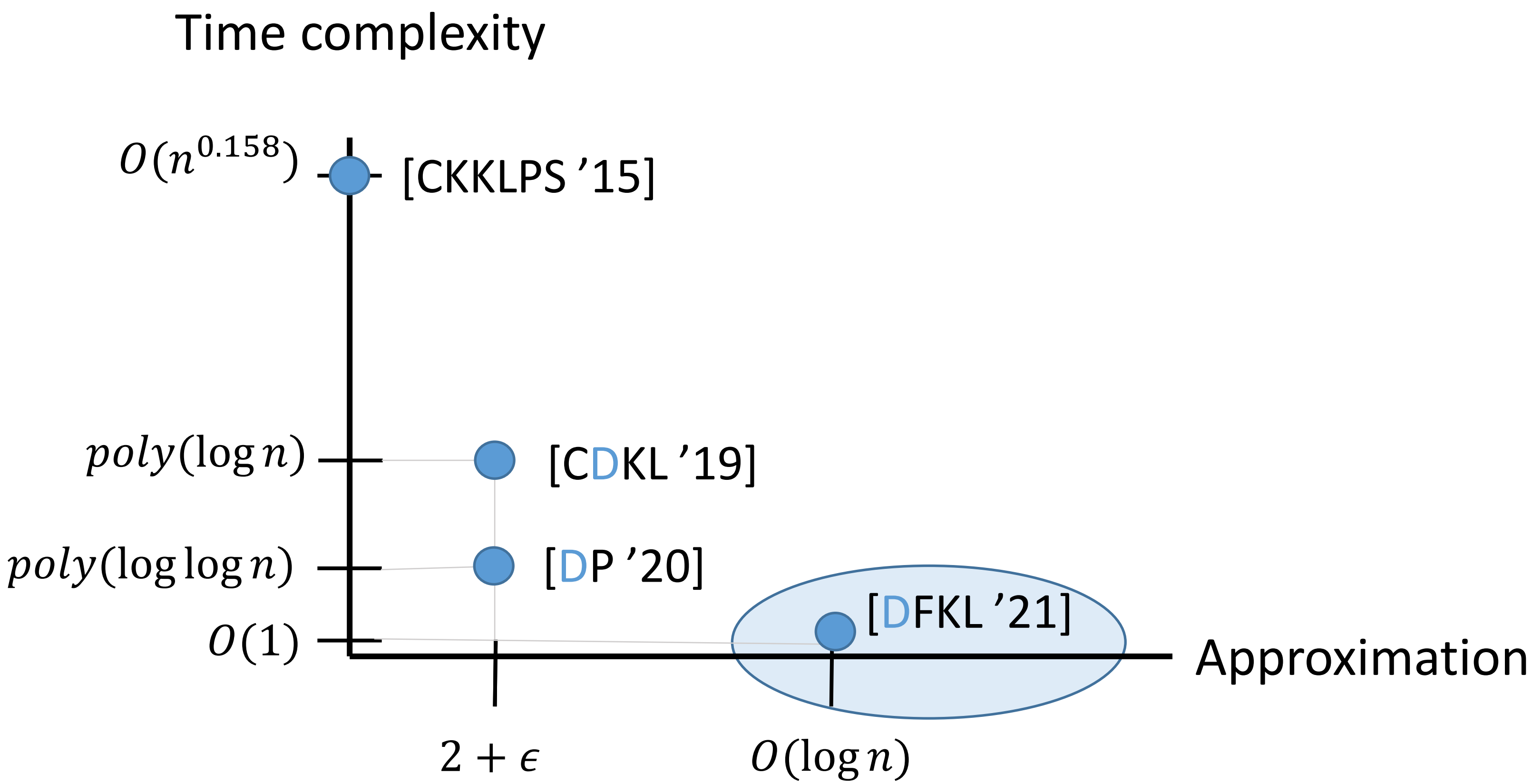
Distributed APSP

Can we get faster algorithms?



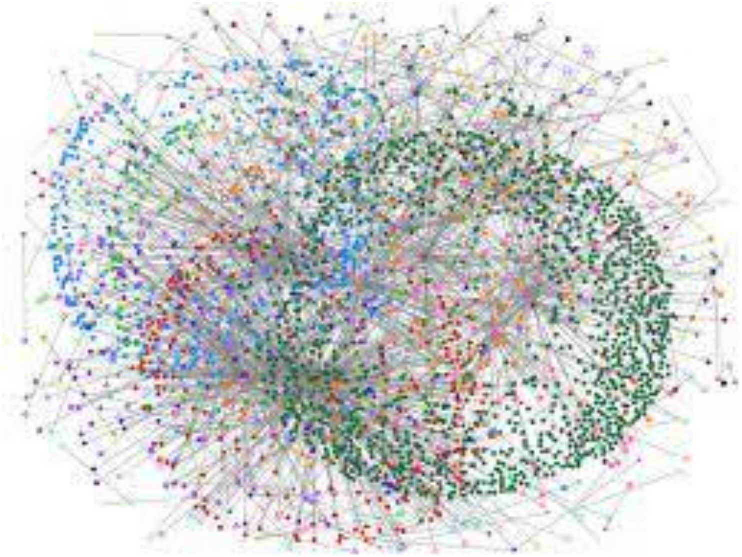
Distributed APSP

Can we get faster algorithms?



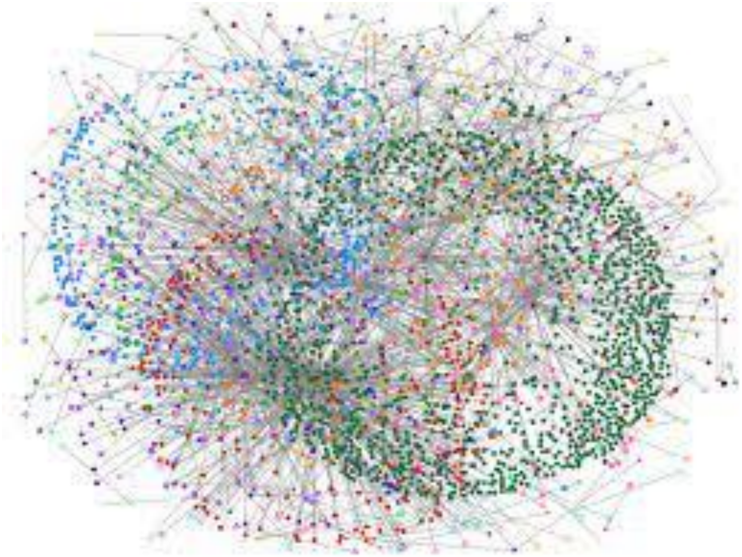
Our Techniques

Goal: compress our graph while preserving the distances



Our Techniques

Goal: compress our graph while preserving the distances

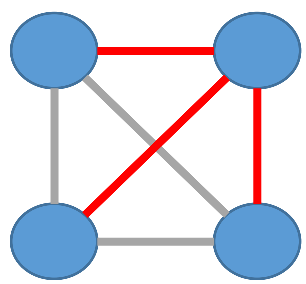


Compute a spanner, a sparse subgraph that approximately preserves the distances

Spanners

- A ***k*-spanner** of a graph G, is a subgraph H of G such that for all u, v:

$$d_G(u, v) \leq d_H(u, v) \leq k \cdot d_G(u, v)$$



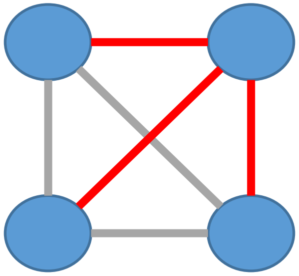
Spanners

- A ***k*-spanner** of a graph G, is a subgraph H of G such that for all u, v:

$$d_G(u,v) \leq d_H(u,v) \leq k \cdot d_G(u,v)$$

<u>Numerous applications</u>:

- Synchronization in distributed networks
- Compact routing schemes
- Approximate shortest paths
- …



Spanners

- A ***k*-spanner** of a graph G, is a subgraph H of G such that for all u, v:
$$d_G(u,v) \leq d_H(u,v) \leq k \cdot d_G(u,v)$$

Theorem [Althöfer et al.,93]: Every graph has a

$(2k-1)$-spanner of size $O(n^{1+1/k})$

Spanners

- A ***k*-spanner** of a graph G, is a subgraph H of G such that for all u, v:

$$d_G(u,v) \leq d_H(u,v) \leq k \cdot d_G(u,v)$$

Theorem [Althöfer et al.,93]: Every graph has a

$$(2k-1)\text{-spanner of size } O(n^{1+1/k})$$

For $k = \log n$: $O(\log n)$-spanner of size $O(n)$

Spanners

Theorem [Althöfer et al.,93]: Every graph has a $(2k-1)$-spanner of size $O(n^{1+1/k})$

$\boxed{\text{Goal: construct } O(\log n)\text{-spanner of size } O(n)}$

Spanners

Theorem [Althöfer et al.,93]: Every graph has a

$$(2k-1)\text{-spanner of size } O(n^{1+1/k})$$

Goal: construct $O(\log n)$-spanner of size $O(n)$

$\Downarrow$

$O(\log n)$-approximation for APSP in $O(1)$ rounds

Spanners

Theorem [Althöfer et al.,93]: Every graph has a

$(2k-1)$-spanner of size $O(n^{1+1/k})$

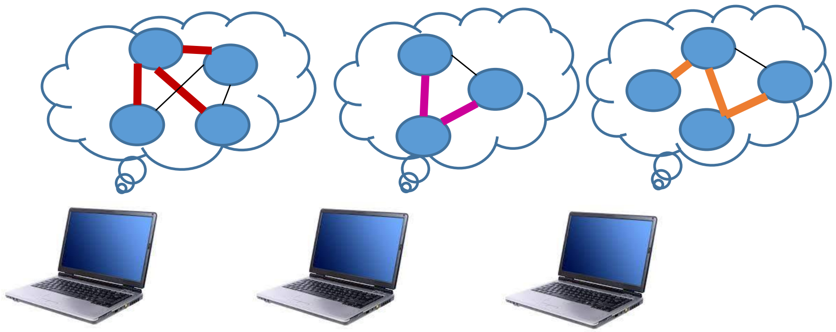
Goal: construct $O(\log n)$-spanner of size $O(n)$

$\Downarrow$

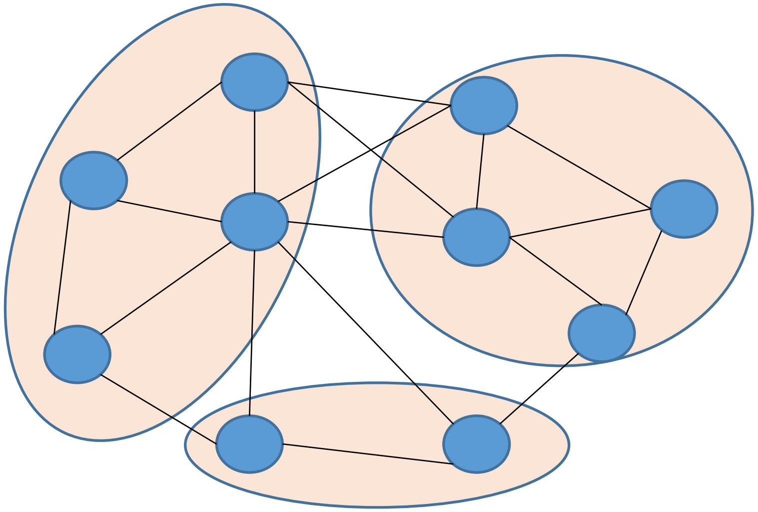
$O(\log n)$-approximation for APSP in $O(1)$ rounds

The Spanner Algorithm

Edge Sparsification

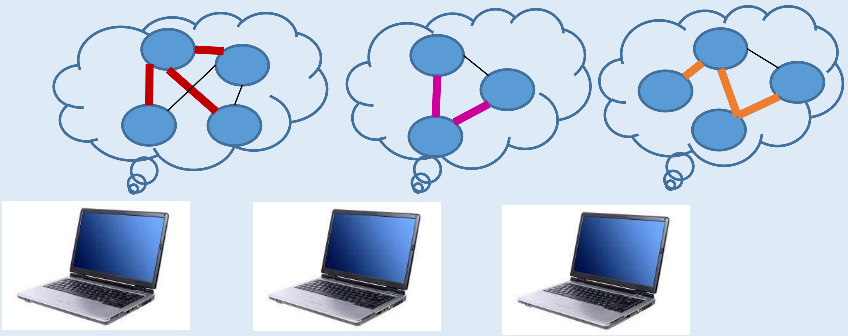


Vertex Sparsification

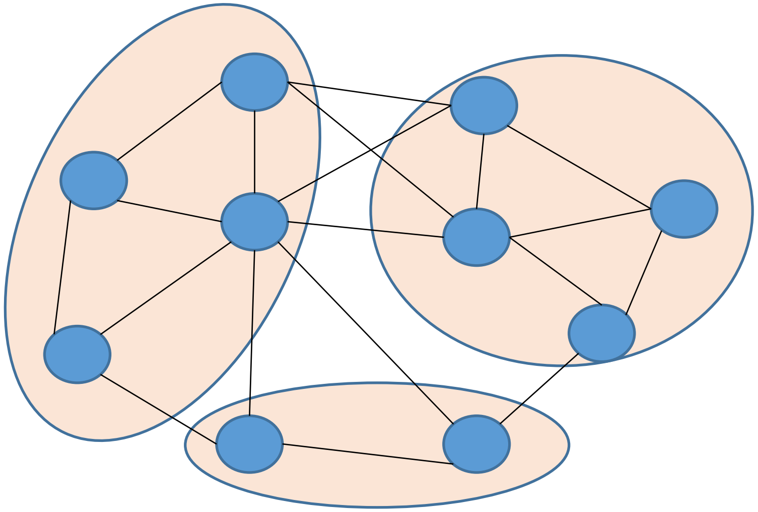


The Spanner Algorithm

Edge Sparsification

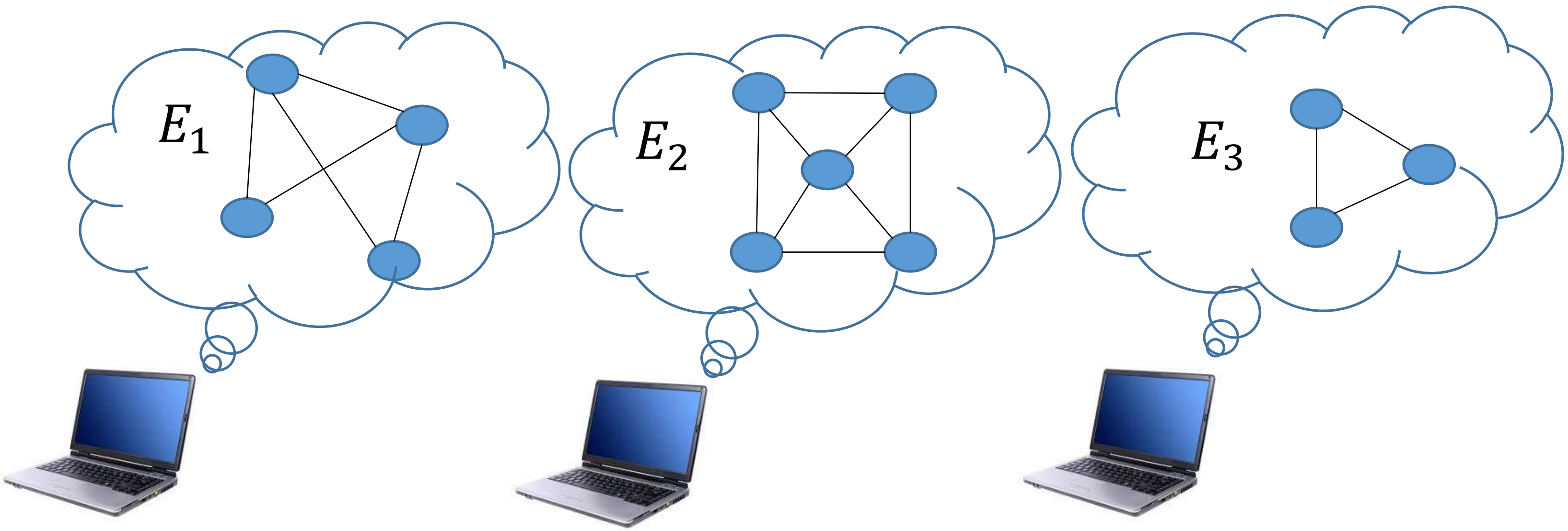


Vertex Sparsification



Edge Sparsification

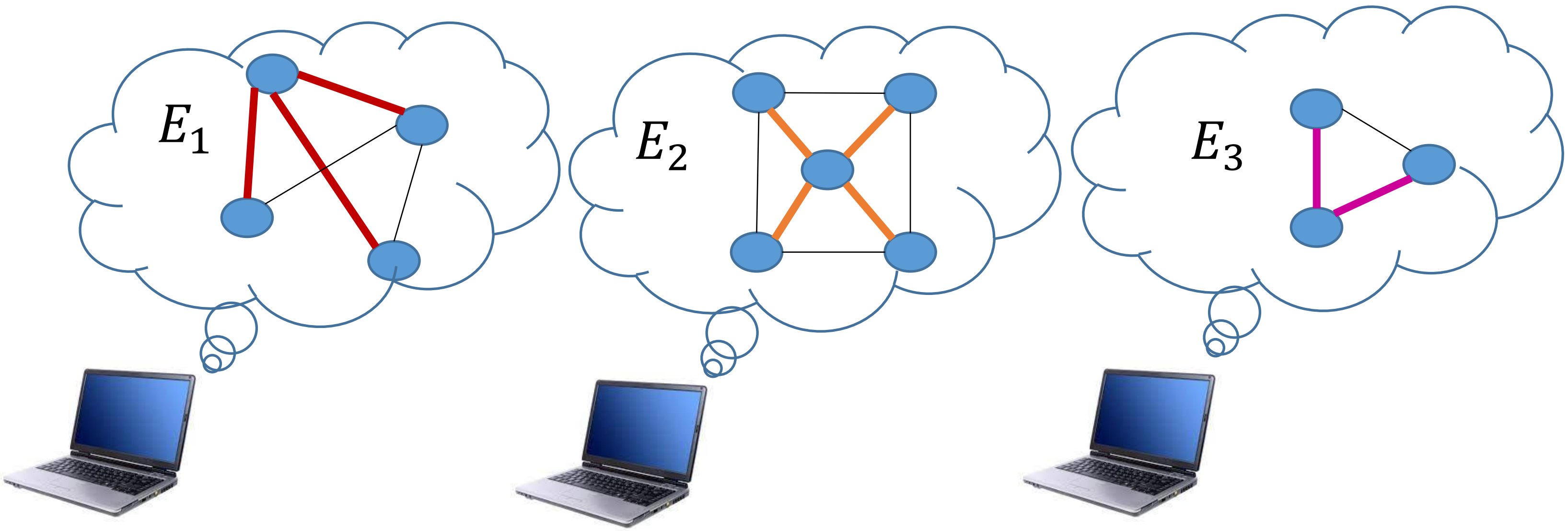
Partition the edges: $E = \cup E_i$



Compute a spanner on each part separately

Edge Sparsification

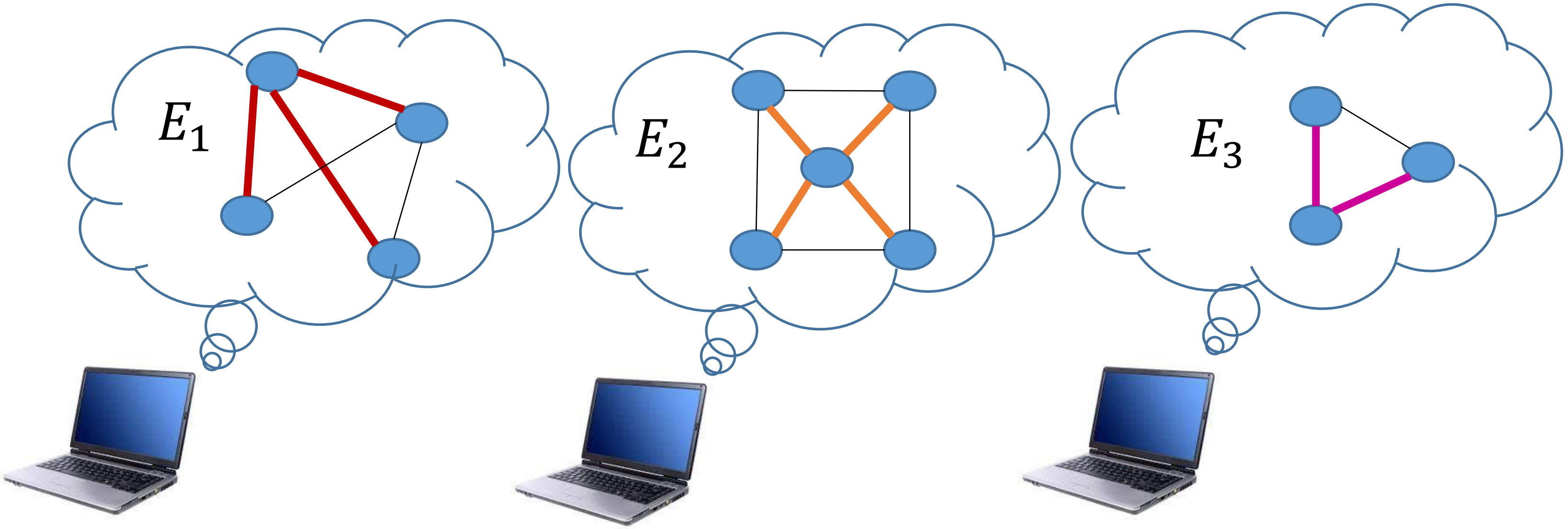
Partition the edges: $E = \cup E_i$



Compute a spanner on each part separately

Edge Sparsification

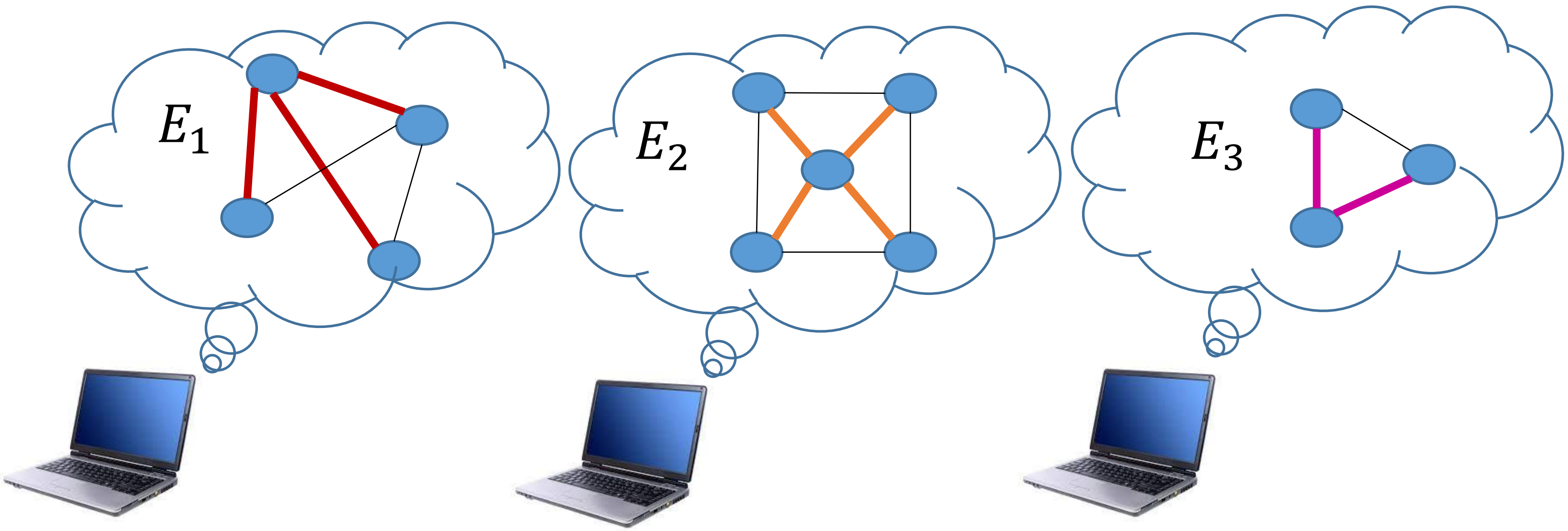
Partition the edges: $E = \cup E_i$



<u>Claim:</u> the union of the spanners is a spanner of the original graph

Edge Sparsification

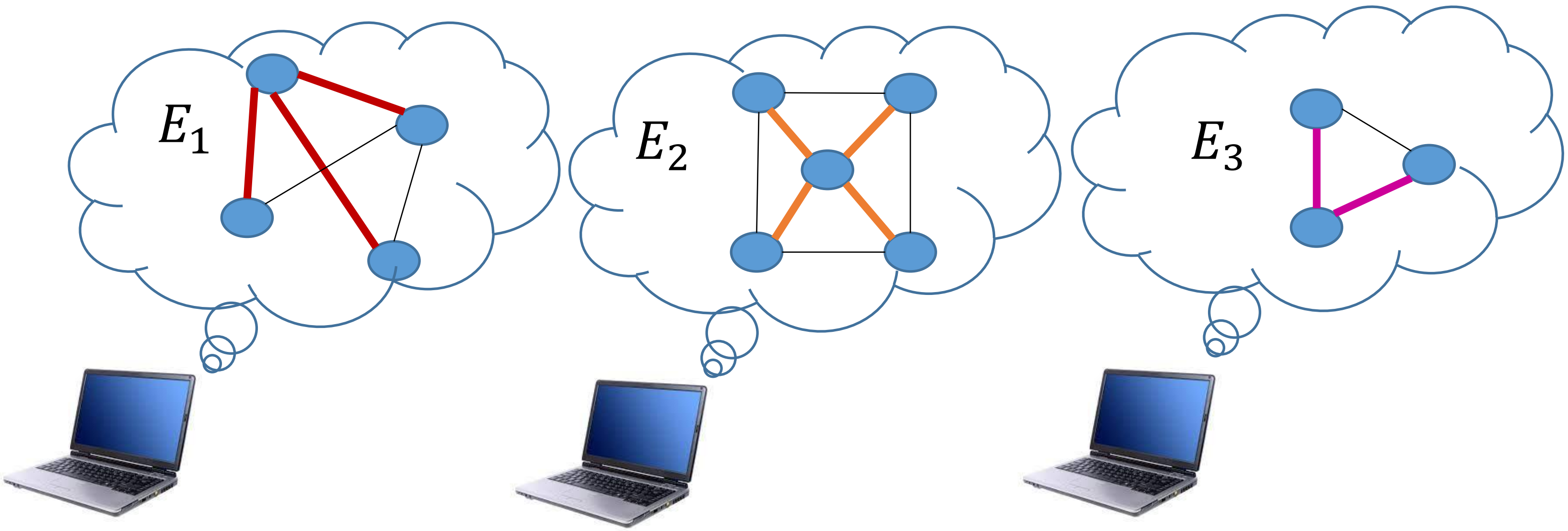
Partition the edges: $E = \cup E_i$



Problem: what is the size of the spanner?

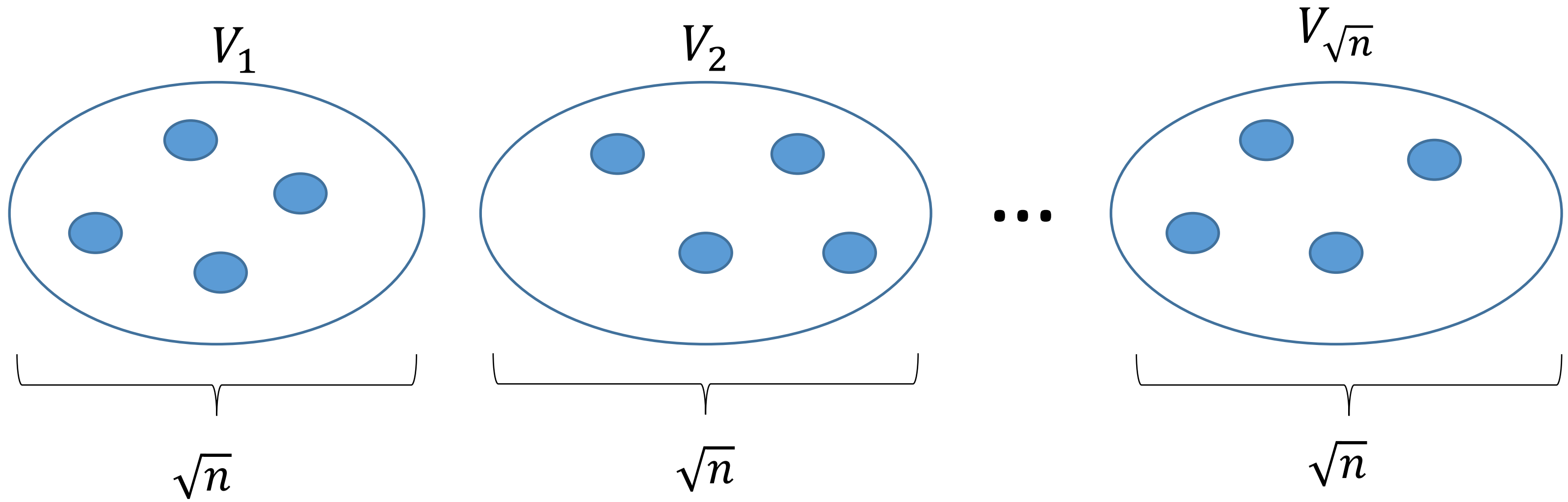
Edge Sparsification

Partition the edges: $E = \cup E_i$



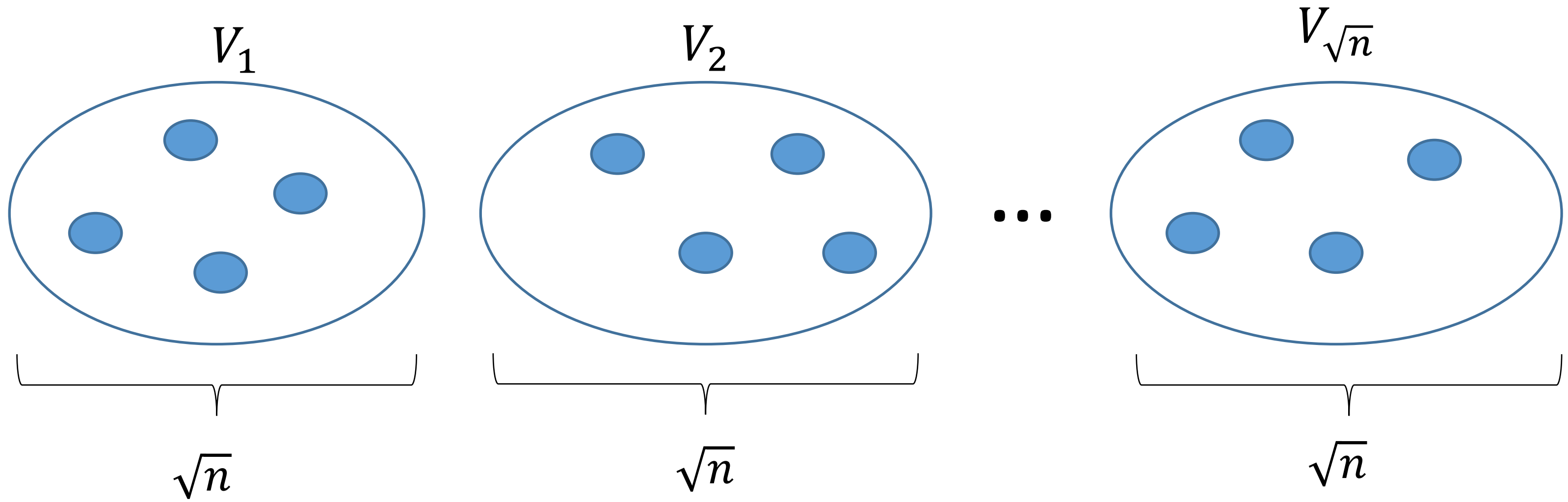
Challenge: find a smart partitioning

Edge Partitioning: Try 1



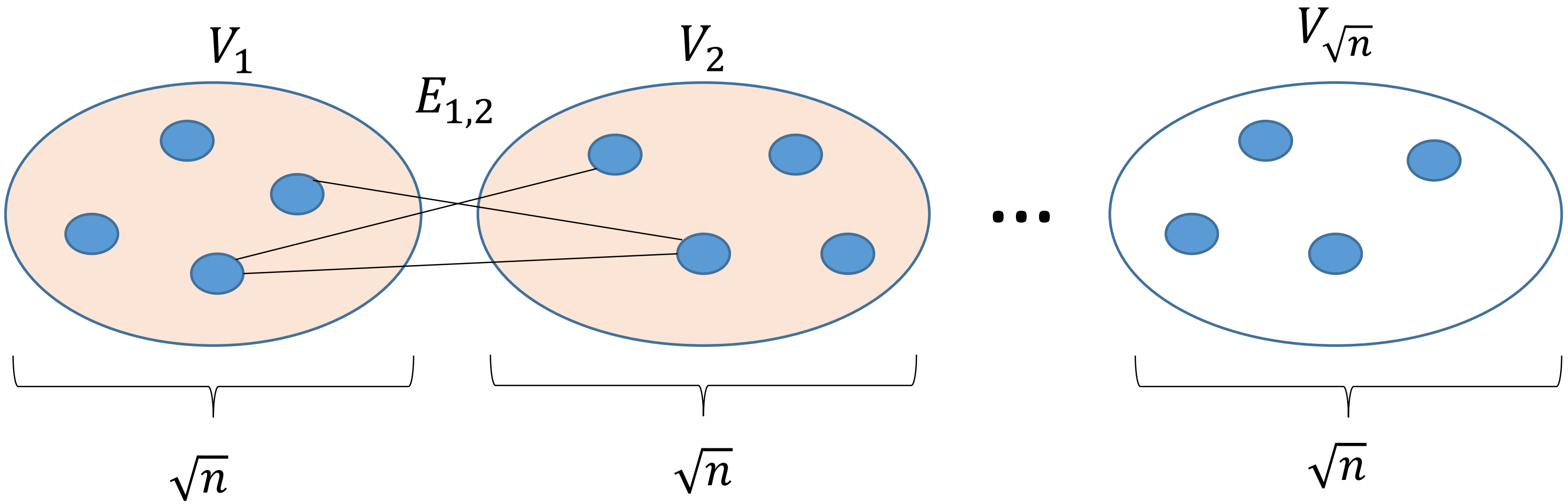
Partition the vertices into $\sqrt{n}$ sets of size $\sqrt{n}$

Edge Partitioning: Try 1



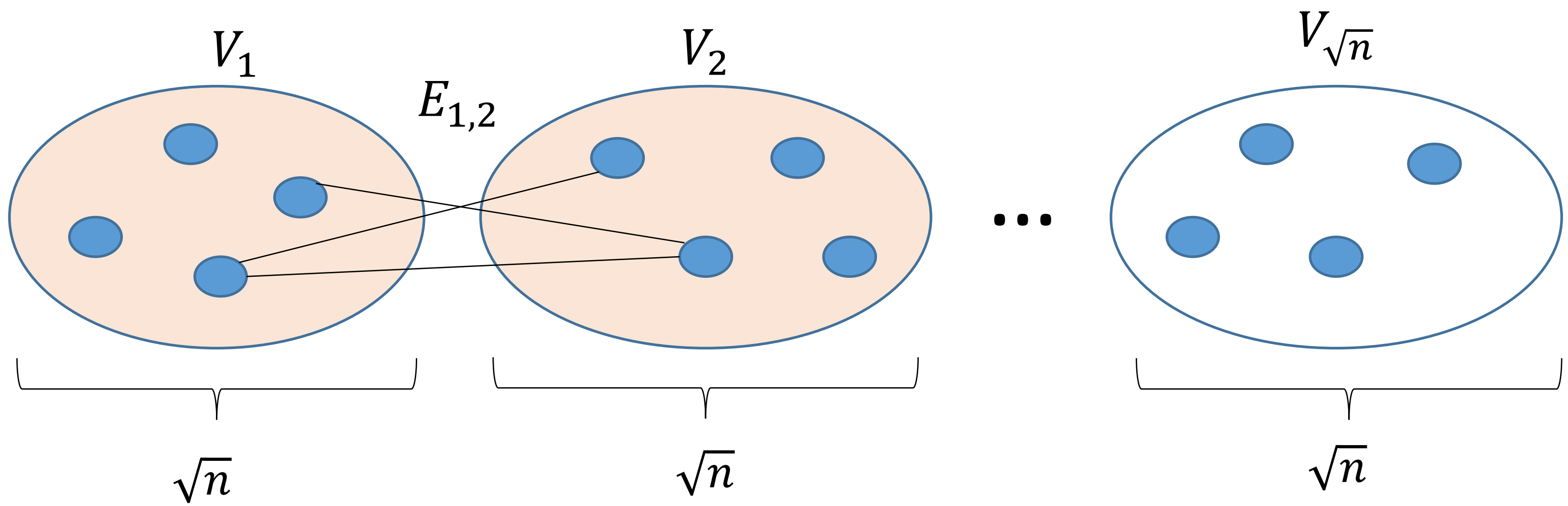
$$E_{i,j} = \left\{\{u, v\} \in E \mid u \in V_i, v \in V_j\right\}$$

Edge Partitioning: Try 1



$$E_{i,j} = \left\{ \{u, v\} \in E \mid u \in V_i, v \in V_j \right\}$$

Edge Partitioning: Try 1

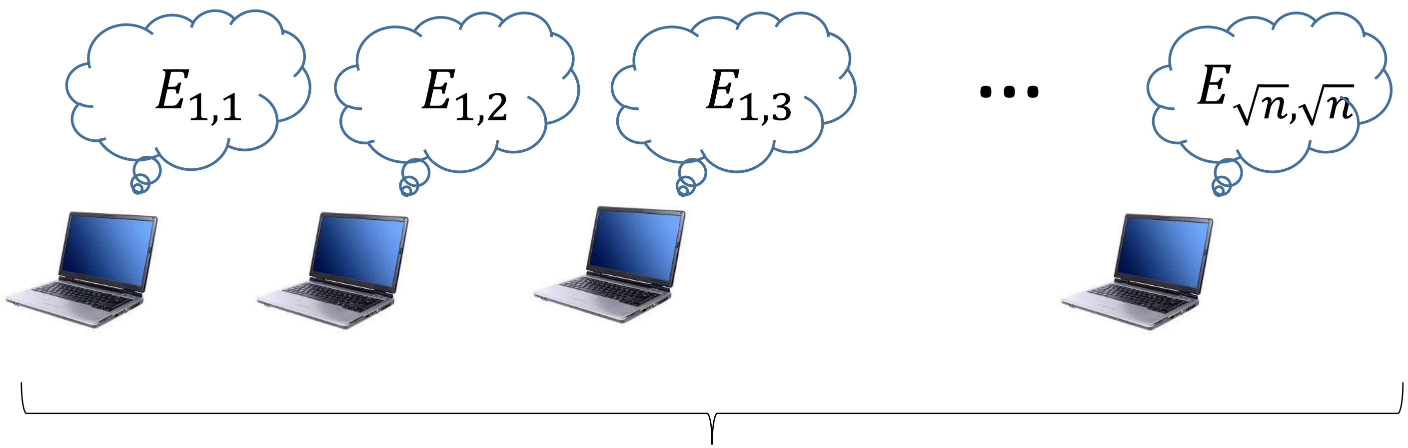


$$E_{i,j} = \left\{\{u, v\} \in E \mid u \in V_i, v \in V_j\right\}$$

Properties: 1. n different subsets $E_{i,j}$

2. $\left|E_{i,j}\right| = O(n)$

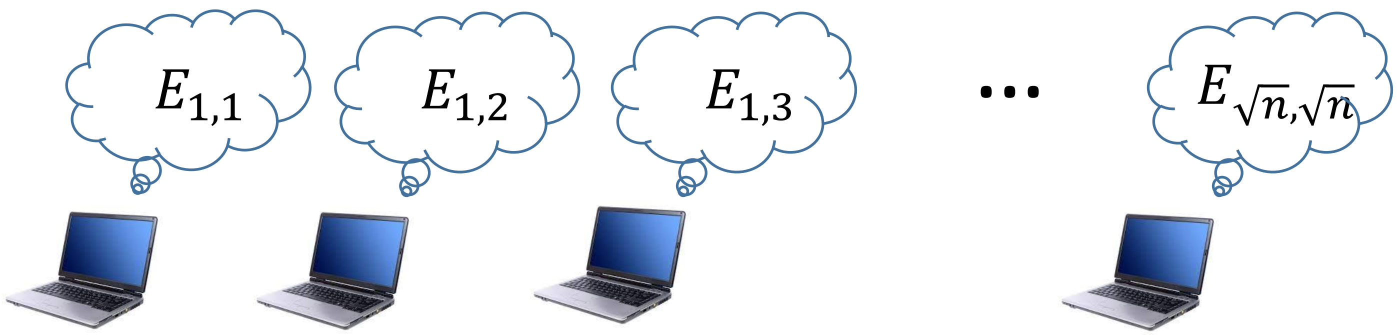
Edge Partitioning: Try 1



n sets of size $O(n)$

Each machine computes a spanner for one set $E_{i,j}$

Edge Partitioning: Try 1



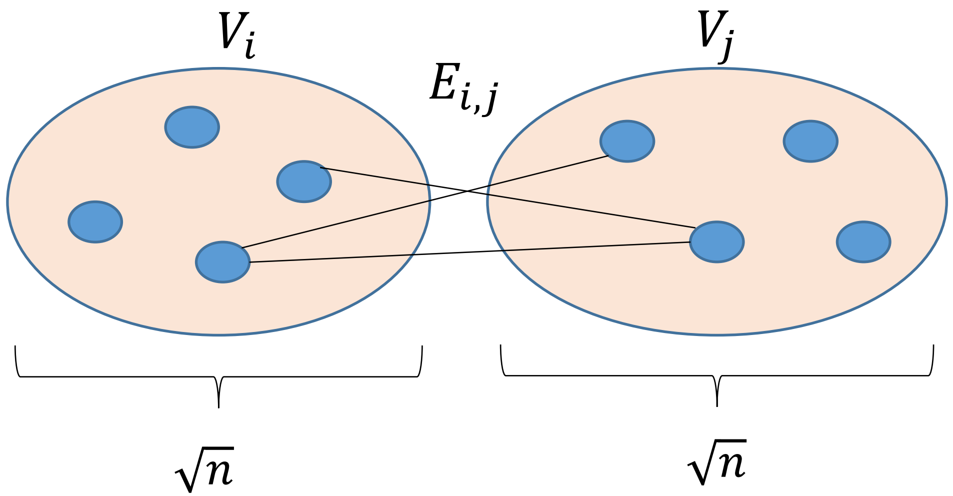
n sets of size $O(n)$

Each machine computes a spanner for one set $E_{i,j}$

What is the size of the spanner?

Edge Partitioning: Try 1

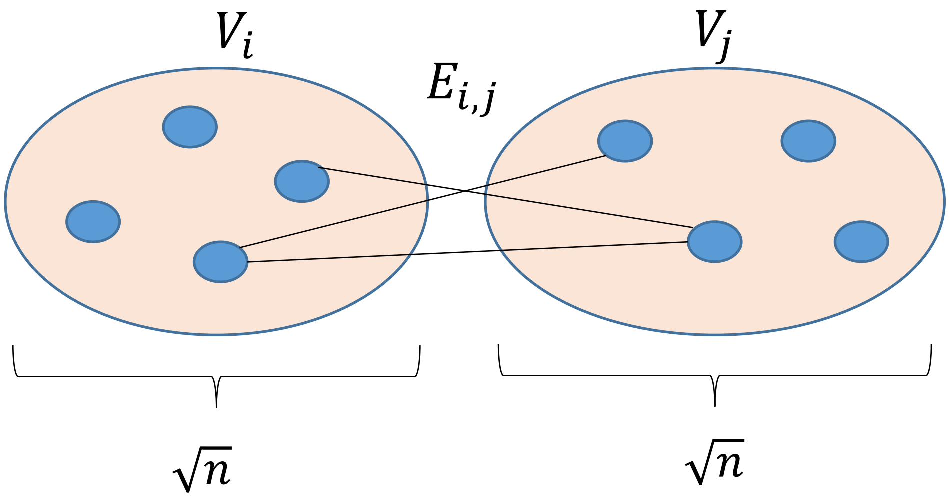
For one set $E_{i,j}$ construct a spanner of size $O(\sqrt{n})$



<u>Overall:</u> $O(n \cdot \sqrt{n}) = O(n^{3/2})$

Edge Partitioning: Try 1

For one set $E_{i,j}$ construct a spanner of size $O(\sqrt{n})$



<u>Overall:</u> $O(n \cdot \sqrt{n}) = O(n^{3/2})$ ➡ Too expensive

<u>Goal</u>: $O(n)$ size

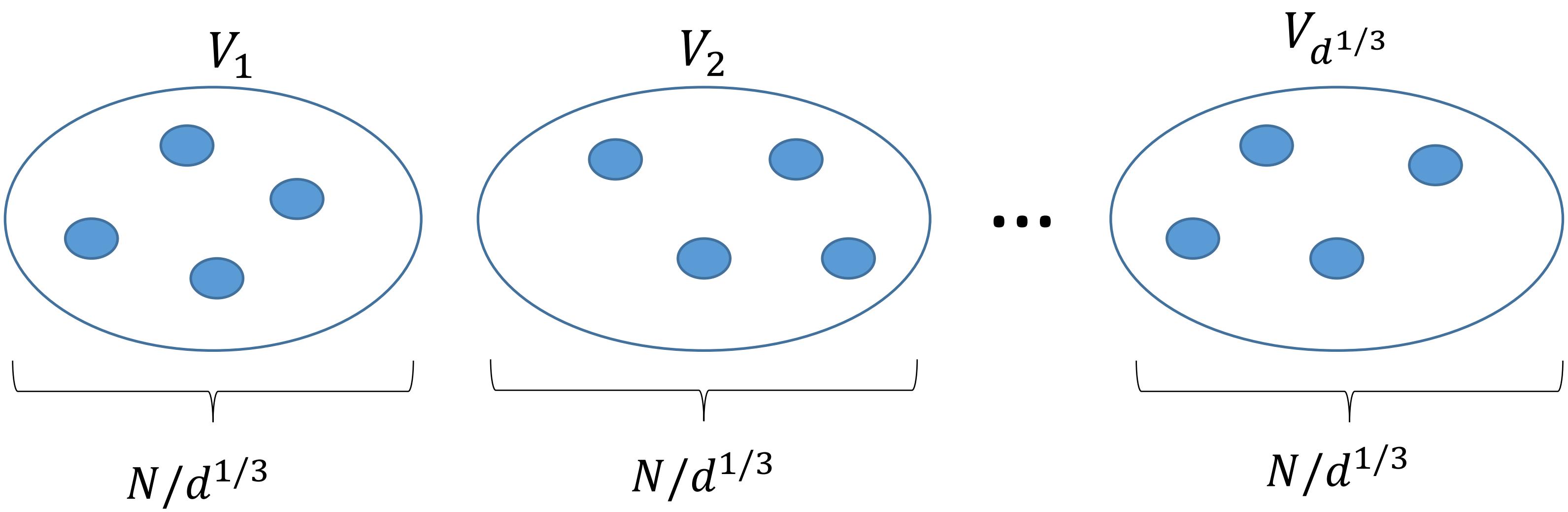
Edge Sparsification

<u>Sparsification theorem:</u> Let G be a graph with N vertices and M edges, we can construct $O(\log N)$-spanner for G with $O(N^{2/3}M^{1/3})$ edges

Edge Sparsification

Sparsification theorem: Let G be a graph with N vertices and M edges, we can construct $O(\log N)$-spanner for G with $O(N^{2/3}M^{1/3})$ edges

$d = M/N$



Edge Sparsification

Sparsification theorem: Let G be a graph with N vertices and M edges, we can construct $O(\log N)$-spanner for G with $O(N^{2/3}M^{1/3})$ edges

This is still larger than $O(n)$ if $m = \omega(n)$

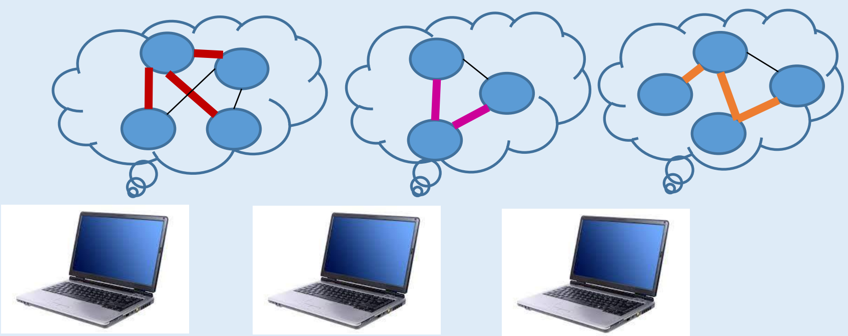
Edge Sparsification

Sparsification theorem: Let G be a graph with N vertices and M edges, we can construct $O(\log N)$-spanner for G with $O(N^{2/3}M^{1/3})$ edges

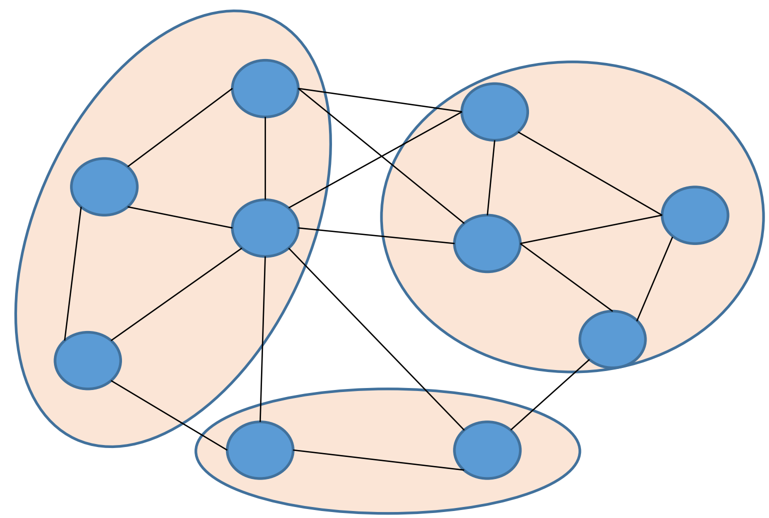
Goal: apply it on a graph where $O(N^{2/3}M^{1/3}) = O(n)$

The Spanner Algorithm

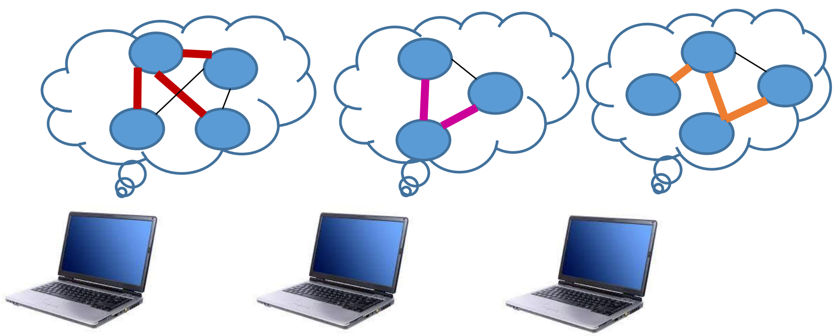
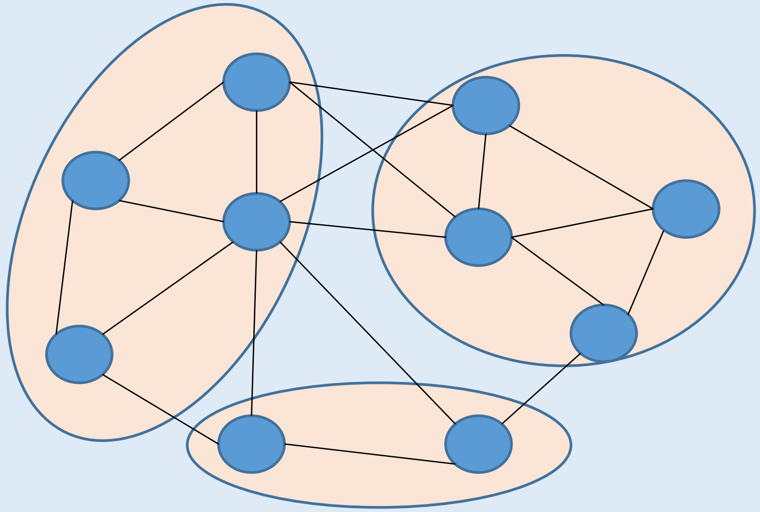
Edge Sparsification



Vertex Sparsification

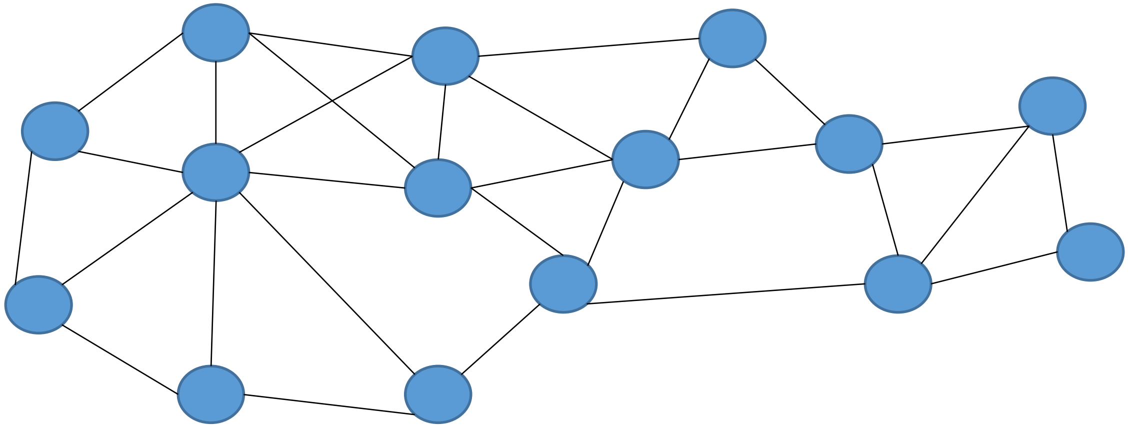


The Spanner Algorithm

Edge Sparsification	Vertex Sparsification
	

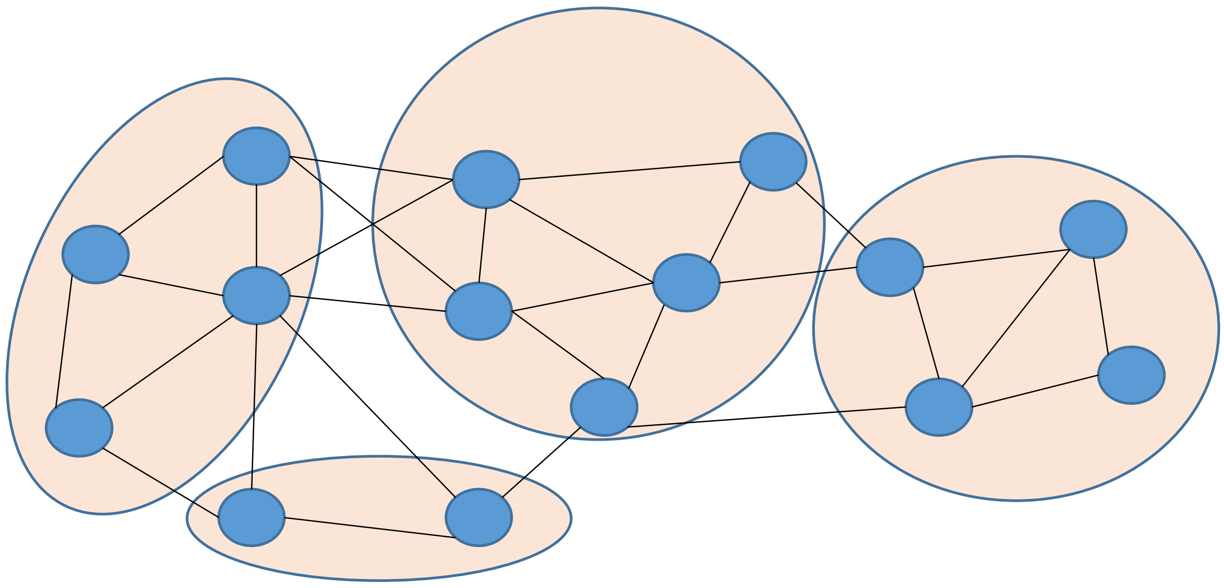
Vertex Sparsification

<u>Idea:</u> divide the vertices into groups of close-by vertices, and treat each group as one vertex



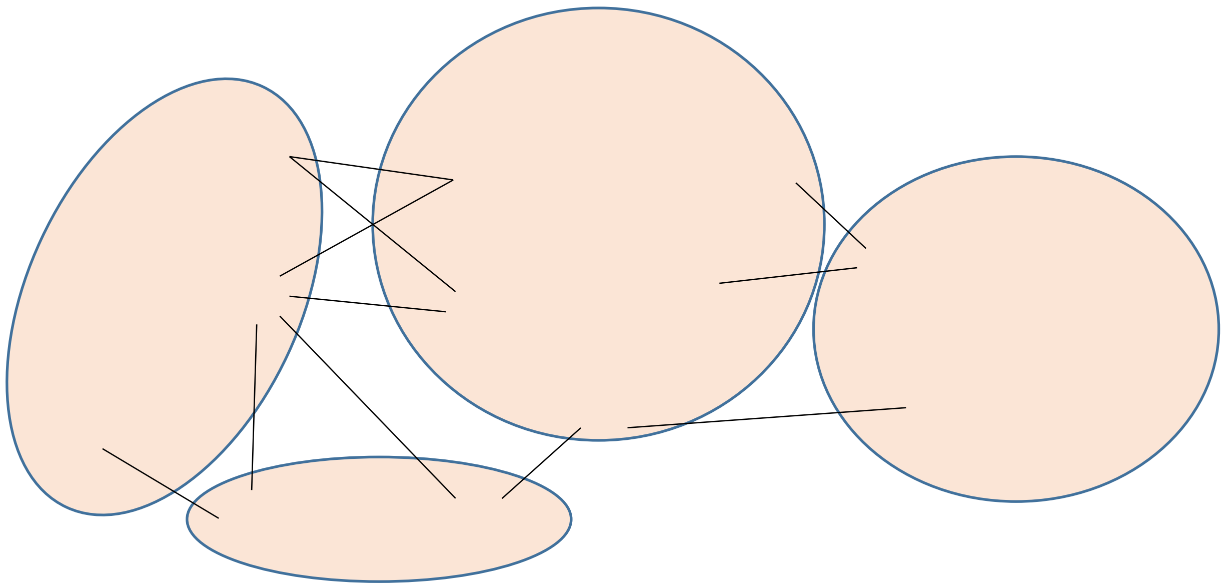
Vertex Sparsification

Idea: divide the vertices into groups of close-by vertices, and treat each group as one vertex



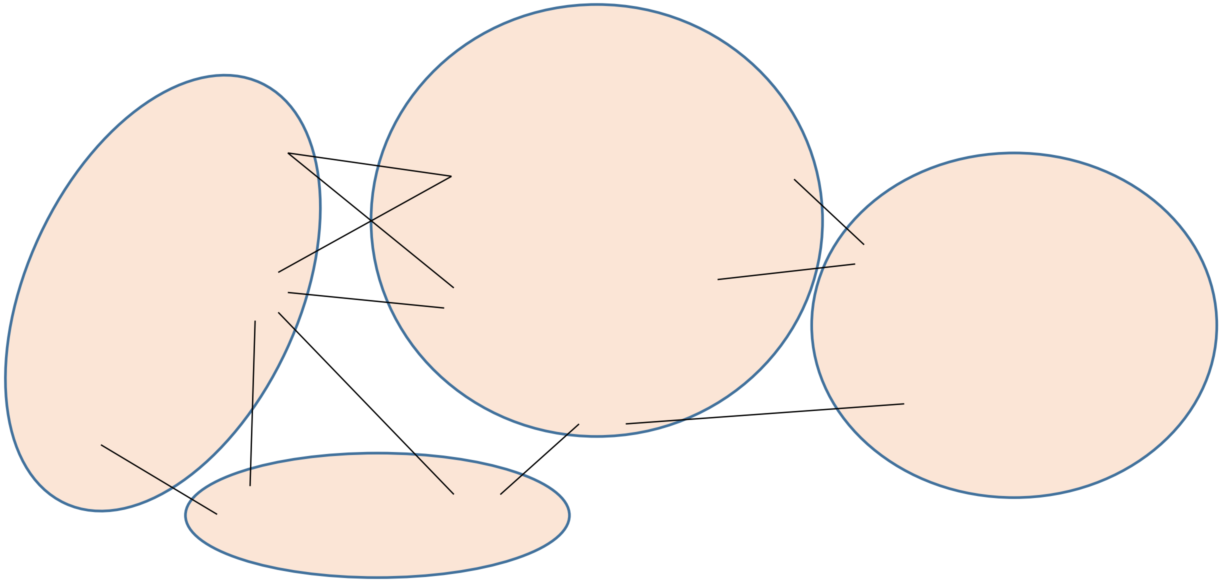
Vertex Sparsification

<u>Idea:</u> divide the vertices into groups of close-by vertices, and treat each group as one vertex



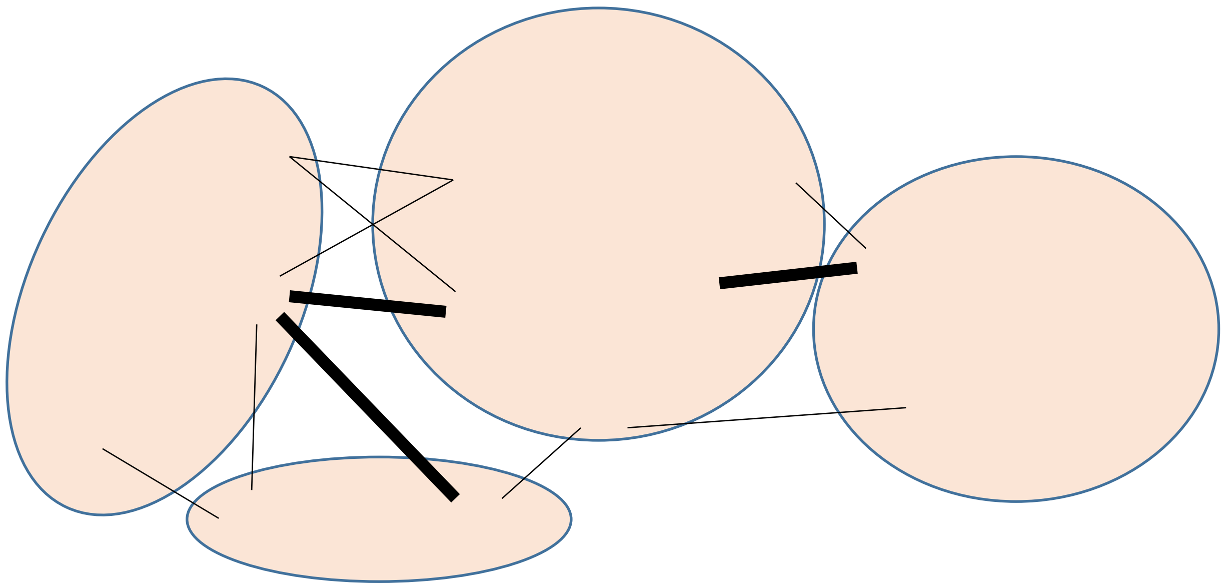
Vertex Sparsification

Using the sparsification theorem: construct $O(n)$ size spanner for the contracted graph



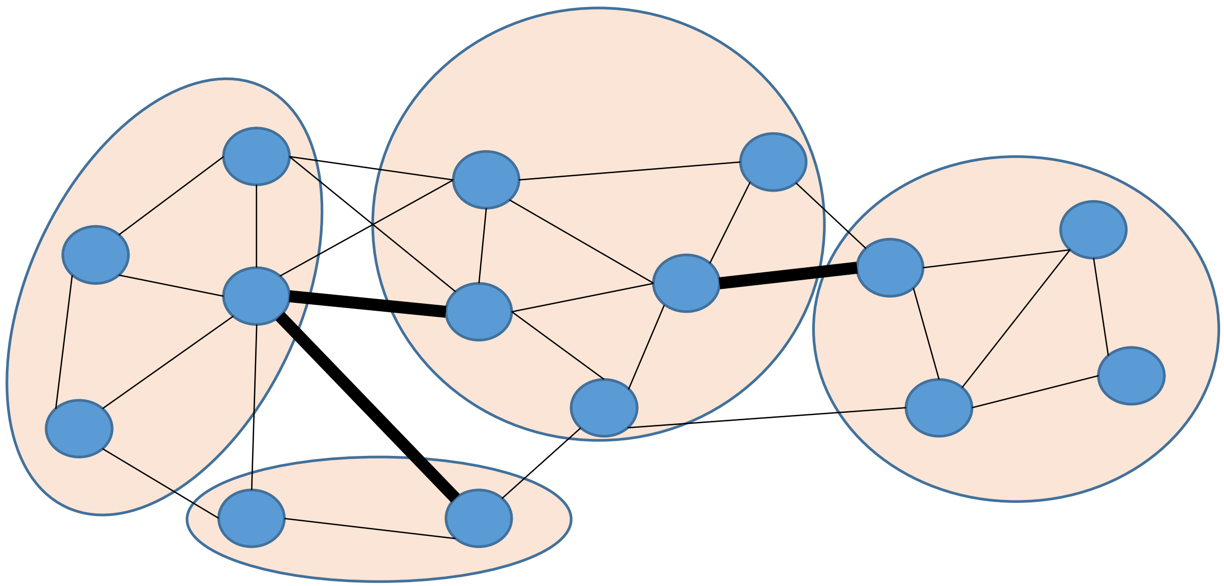
Vertex Sparsification

Using the sparsification theorem: construct $O(n)$ size spanner for the contracted graph



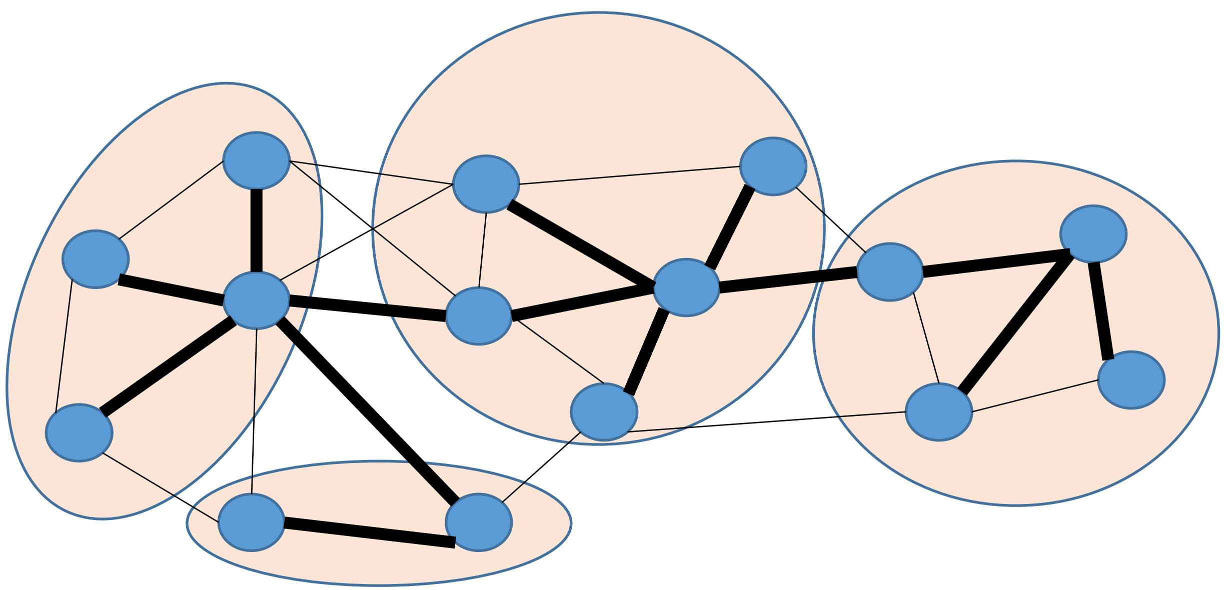
Vertex Sparsification

Can be converted to a spanner for the original graph



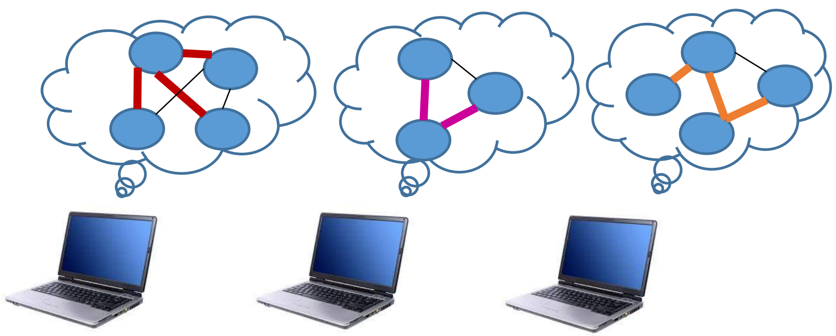
Vertex Sparsification

Can be converted to a spanner for the original graph

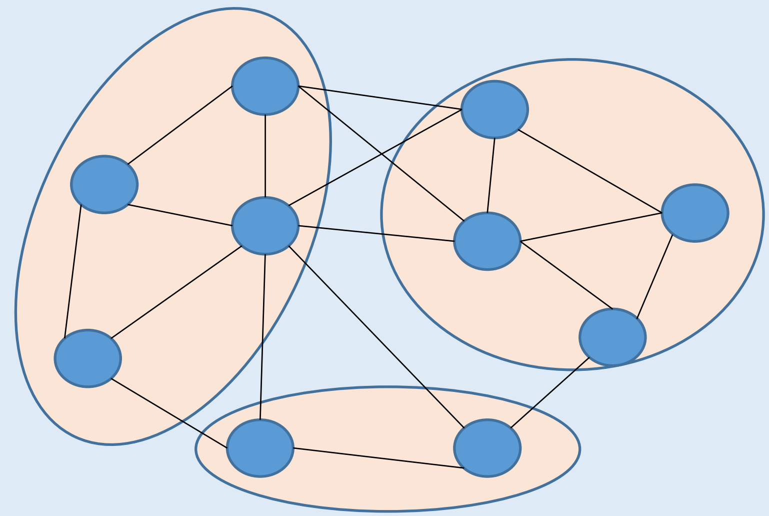


The Spanner Algorithm

Edge Sparsification



Vertex Sparsification



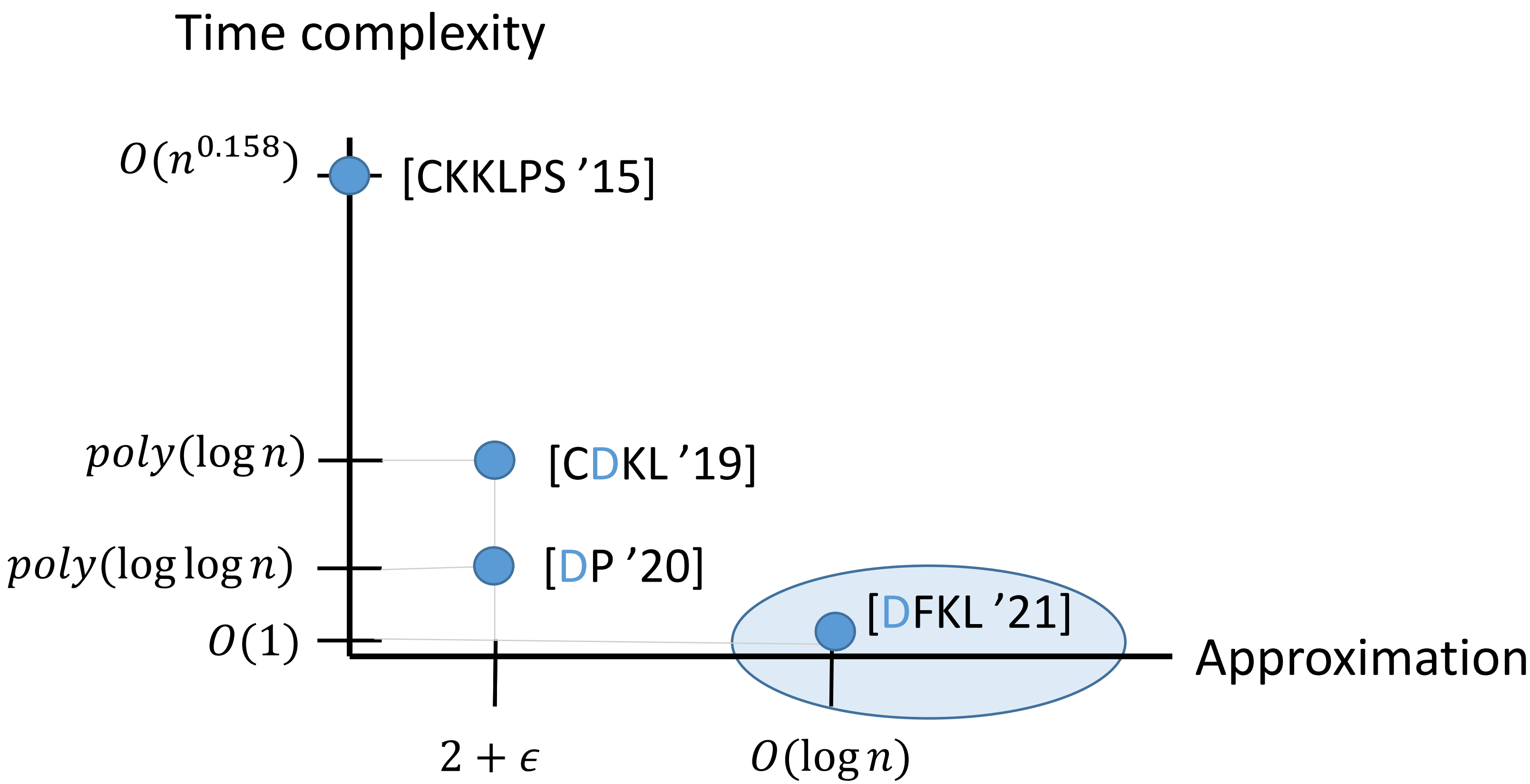
Conclusion

$O(\log n)$-spanners of size $O(n)$

$\Downarrow$

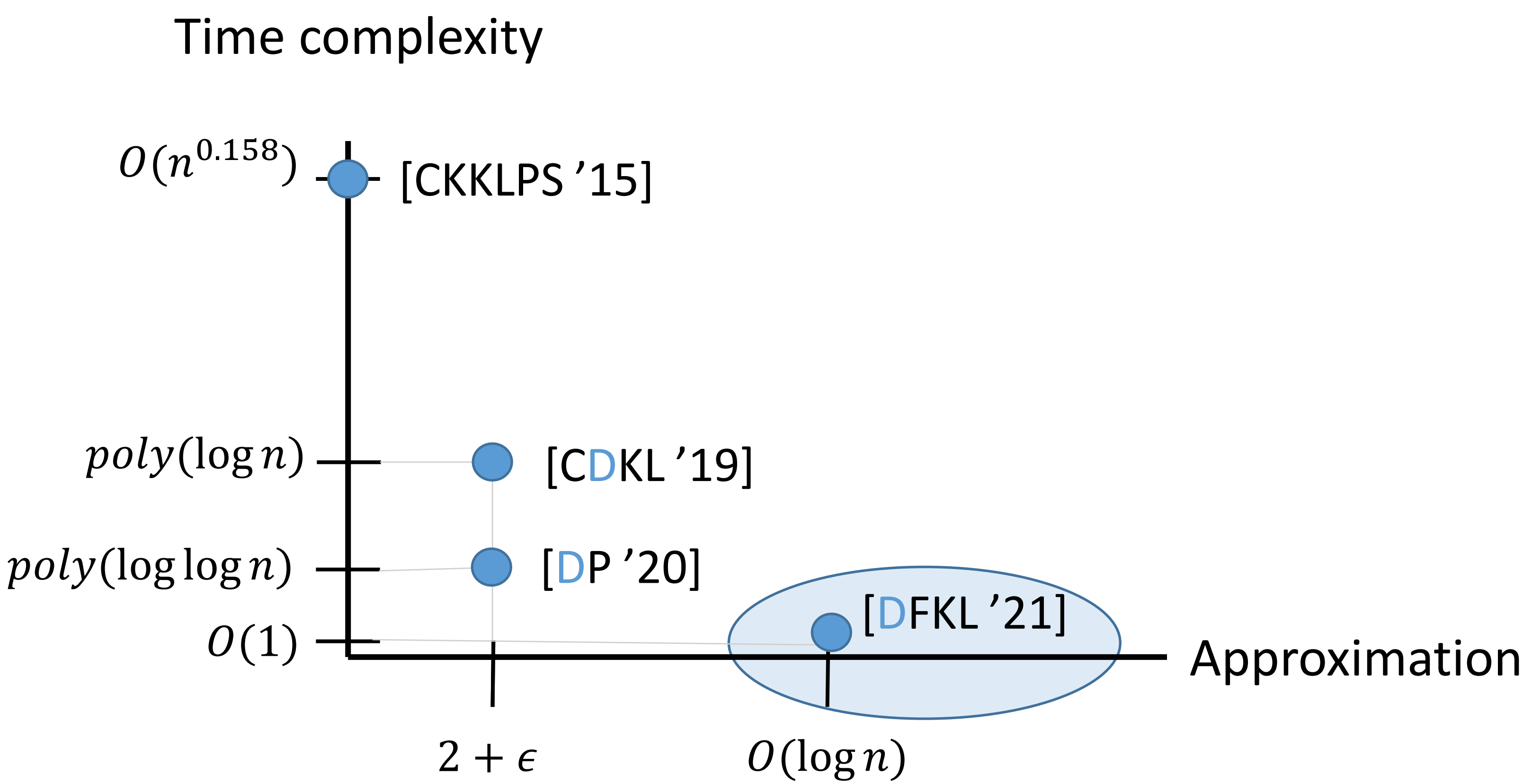
$O(\log n)$-approximation for APSP in $O(1)$ rounds

Distributed APSP



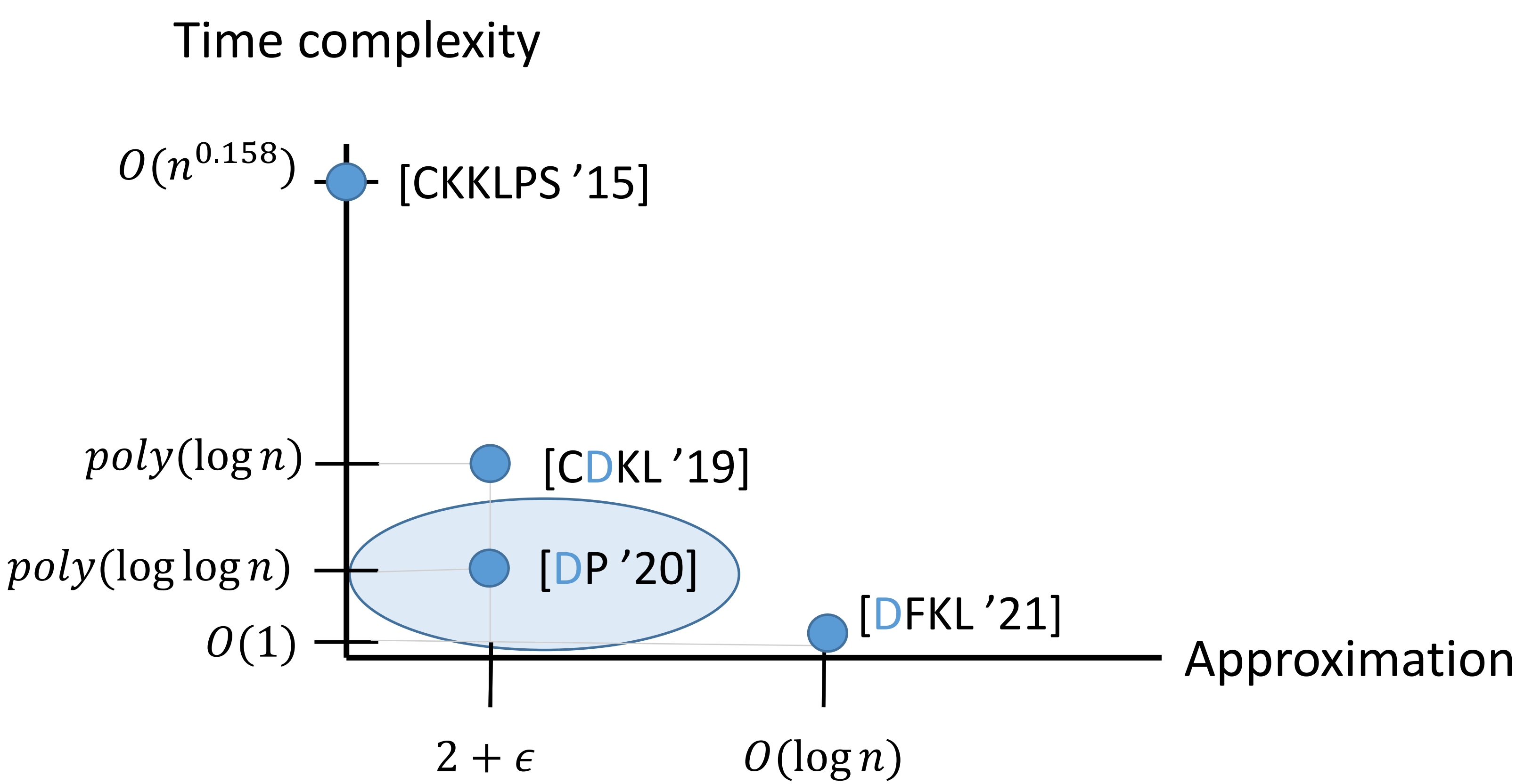
Distributed APSP

How to get a better approximation?

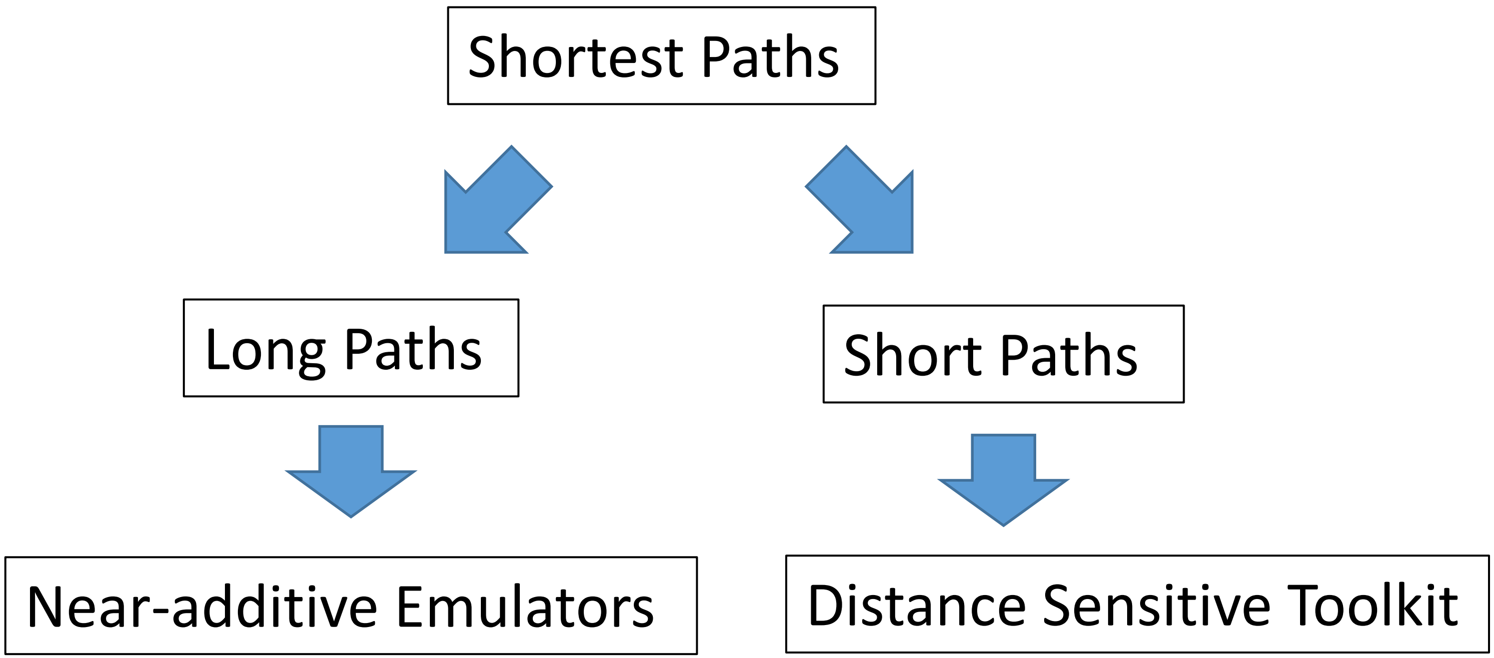


Distributed APSP

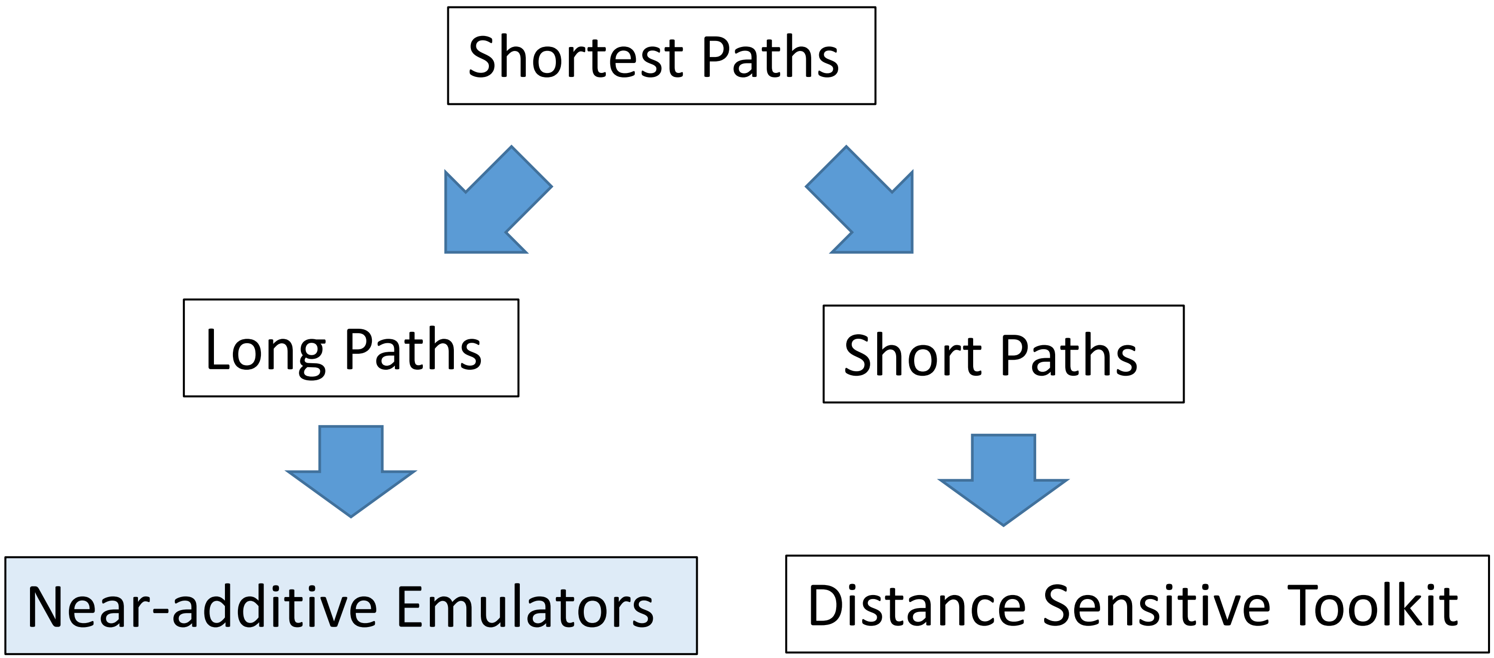
How to get a better approximation?



Our Techniques



Our Techniques



Near-additive Emulator

A sparse graph H such that for all u, v:

$$d(u, v) \leq d_H(u, v) \leq (1 + \epsilon)d(u, v) + \beta$$

Near-additive Emulator

A sparse graph H such that for all u, v:

$$d(u,v) \leq d_H(u,v) \leq (1+\epsilon)d(u,v) + \beta$$

$(1 + \Theta(\epsilon))$-approximation for long distances!

Near-additive Emulator

A sparse graph H such that for all u, v:

$$d(u,v) \leq d_H(u,v) \leq (1+\epsilon)d(u,v) + \beta$$

Build a sparse near-additive emulator

$(1 + \Theta(\epsilon))$-approximation for long distances!

Shortest Paths via Emulators

Left with short paths of length $t = O(\beta/\epsilon)$

$\boxed{\text{Requires } poly(\log t) = poly(\log\log n) \text{ time!}}$

Shortest Paths via Emulators

Shortest Paths

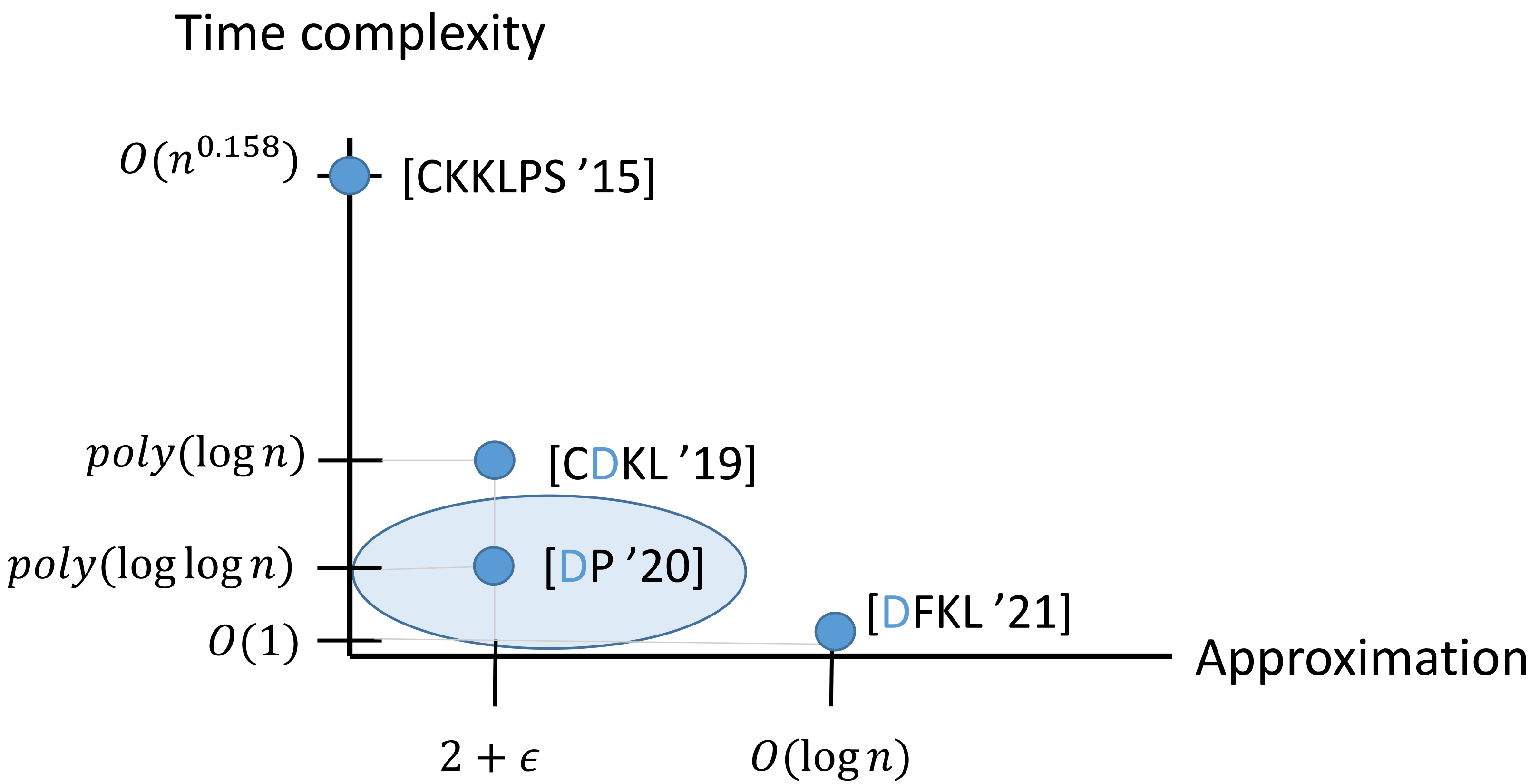
Long Paths: $\Omega(\beta/\epsilon)$

Short Paths: $O(\beta/\epsilon)$

Near-additive Emulators:
$(1+\Theta(\epsilon))$-approximation

Distance Sensitive Toolkit:
$poly\left(\log\frac{\beta}{\epsilon}\right)$ time

Distributed APSP



Conclusion

We saw:

Round Complexity	**Approximation**
$poly(\log\log n)$	$2+\epsilon$
$O(1)$	$O(\log n)$

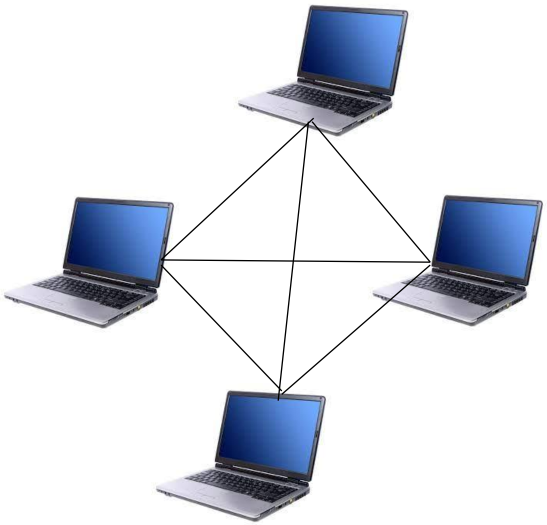
Conclusion

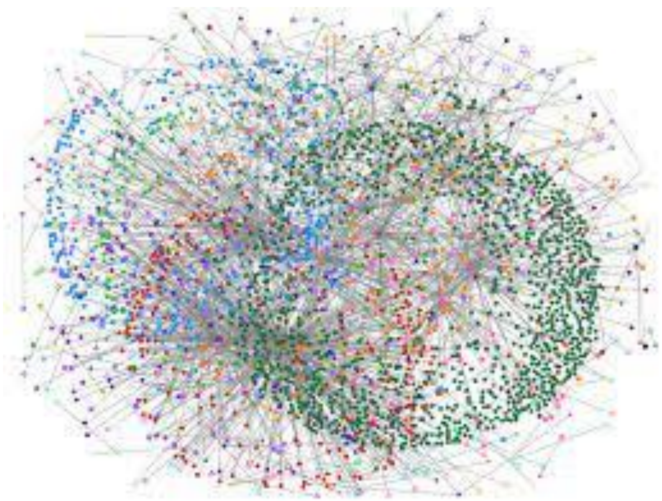
We saw:

Round Complexity	Approximation
$poly(\log \log n)$	$2 + \epsilon$
$O(1)$	$O(\log n)$

Can we get $O(1)$-approximation in $O(1)$ rounds?

The Model - Congested Clique

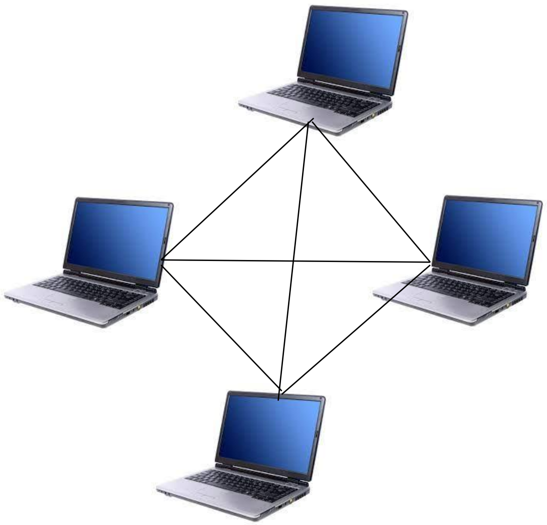




n vertices

- n machines
- Input per machine: $O(n)$ edges
- Each machine can send/receive a total of n messages of $O(\log n)$ bits per round

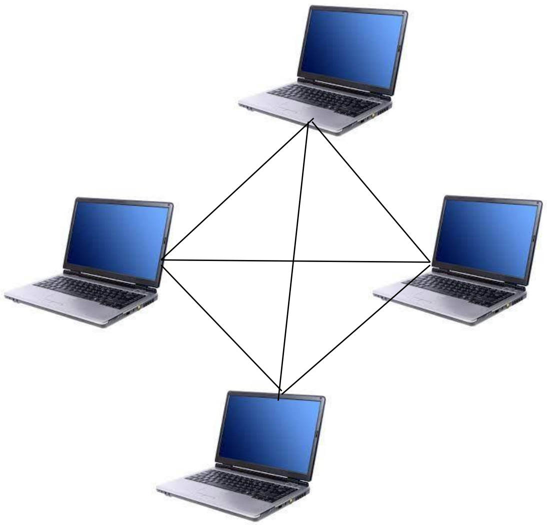
The Model - Congested Clique



What happens if we bound the memory?

- n machines
- Input per machine: $O(n)$ edges
- Each machine can send/receive a total of n messages of $O(\log n)$ bits per round

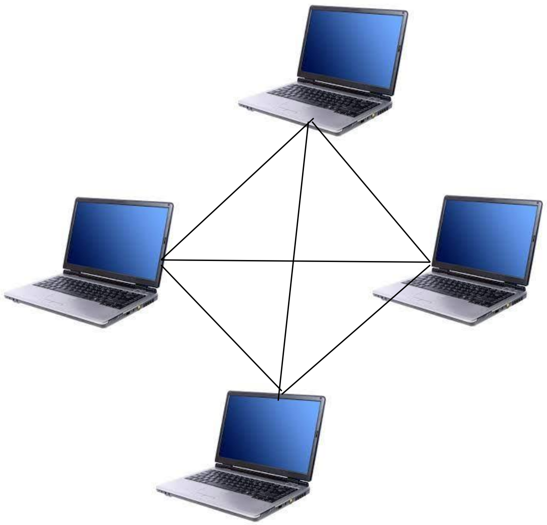
The Model – Linear Memory MPC



What happens if we bound the memory?

- $O(m/n)$ machines – total memory of $\tilde{O}(m)$
- $\tilde{O}(n)$ memory per machine

The Model – Linear Memory MPC

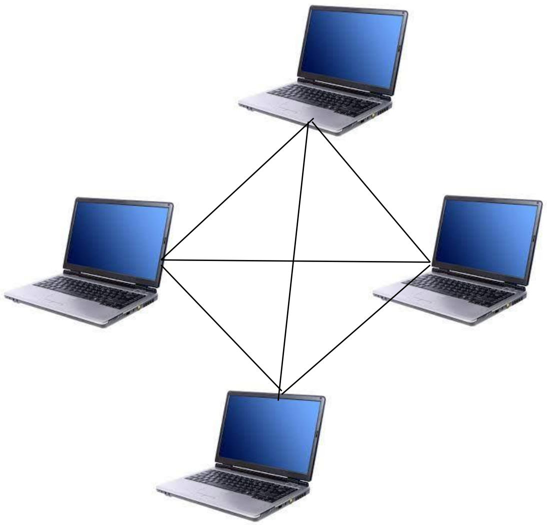


☑ $O(\log n)$-approximation

☒ $O(1)$-approximation

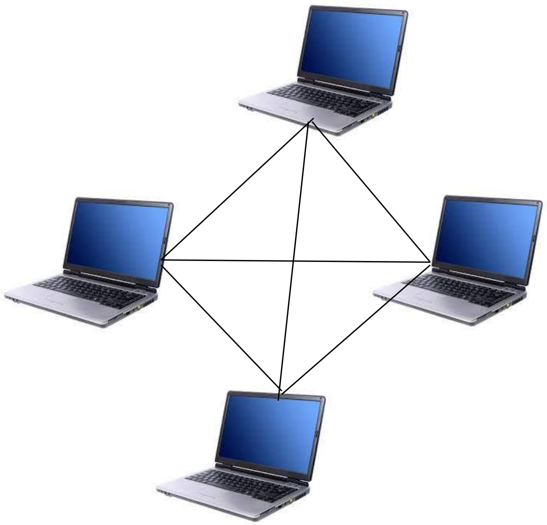
- $O(m/n)$ machines – total memory of $\tilde{O}(m)$
- $\tilde{O}(n)$ memory per machine

The Model – Sublinear Memory MPC



- $O(m/n^{\gamma})$ machines – total memory of $\tilde{O}(m)$
- $\tilde{O}(n^{\gamma})$ memory per machine, $\gamma < 1$

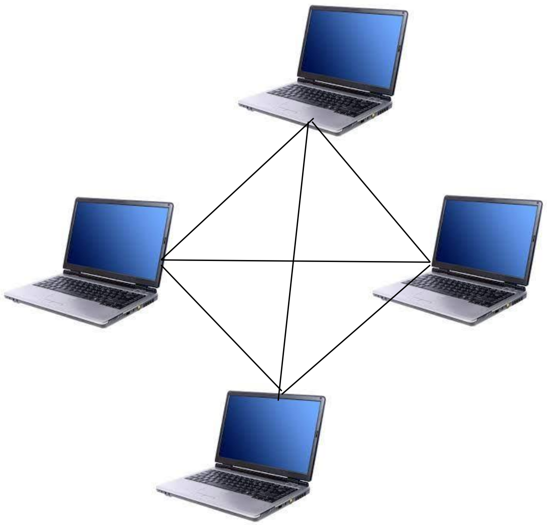
The Model – Sublinear Memory MPC



- Spanners in $poly(\log\log n)$ time [BDGMN, 2021]
- Conditional $\Omega(\log n)$ lower bounds for shortest paths computations

- $O(m/n^{\gamma})$ machines – total memory of $\tilde{O}(m)$
- $\tilde{O}(n^{\gamma})$ memory per machine, $\gamma < 1$

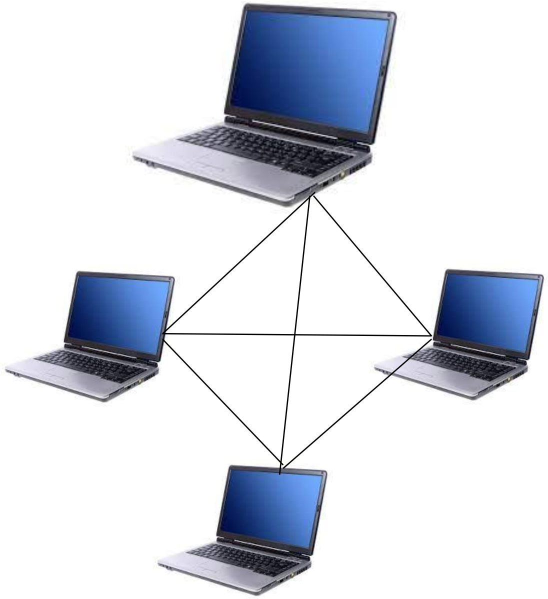
The Model – Sublinear Memory MPC



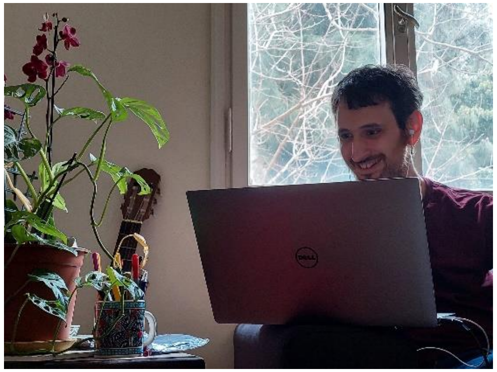
<u>See Friday's AMG talk</u>:
Yasamin Nazari – Distance Computation in MPC and related models

- $O(m/n^{\gamma})$ machines – total memory of $\tilde{O}(m)$
- $\tilde{O}(n^{\gamma})$ memory per machine, $\gamma < 1$

The Model –Heterogeneous MPC



Next ADGA talk:
Orr Fischer - Massively Parallel Computation in a Heterogeneous Regime: One Strong Machine Makes a Big Difference



One linear machine,
The rest – sublinear machines

Open Questions

- Can we get $O(1)$-approximation in $O(1)$ rounds?
- Directed/Exact shortest paths
- What other problems can be solved in $O(1)$ rounds?

Open Questions

<u>$O(1)$-round algorithms</u>:
- Approximate APSP
- Spanners
- Minimum Spanning Tree
- $(\Delta + 1)$-Coloring

<u>$O(\log \log n)$-round algorithms</u>:
- Approximate Maximum Matching
- Maximal Independent Set
- Approximate Vertex Cover